https://doi.org/10.15388/vu.thesis.57
https://orcid.org/0000-0001-6111-3277

VILNIUS UNIVERSITY

Konstantinas

KOROVKINAS

Hybrid Method for Textual Data Sentiment Analysis

DOCTORAL DISSERTATION

Natural Sciences
Informatics N 009

VILNIUS 2020

This dissertation was written between 2015–2020 at Vilnius University.

Scientific supervisor:
Prof. Dr. Gintautas Garšva (Vilnius University, Natural Sciences, Informatics – N 009).

https://doi.org/10.15388/vu.thesis.57
https://orcid.org/0000-0001-6111-3277

VILNIAUS UNIVERSITETAS

Konstantinas

KOROVKINAS

Hibridinis tekstinių duomenų metodas nuomonių analizei

DAKTARO DISERTACIJA

Gamtos mokslai
Informatika N 009

VILNIUS 2020

Disertacija rengta 2015–2020 metais Vilniaus universitete.

Mokslinis vadovas:
prof. dr. Gintautas Garšva (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

SUMMARY

Textual data sentiment analysis became very popular when people started using the Internet. Nowadays if you want to get an opinion about surveys, social, economic and others events, you can find all information you need on the Internet. The main goal of research related to sentiment analysis is to obtain authors' feelings expressed in positive or negative comments. This dissertation focuses on machine learning methods for textual data sentiment analysis in large scale datasets. The goal of the research is to propose a hybrid sentiment analysis method with a recommended set of parameters for large textual data with a better execution time and with a similar classification accuracy compared with classical methods. The proposed hybrid method is a combination of four methods: classical machine learning algorithm, k-Means clustering, particle swarm optimization metaheuristic and ensemble, which are integrated into acceleration method – SpeedUP. The SpeedUP method is the main method, which automatically performs all parts of the proposed hybrid method, based on specified parameters, which are recommended in this research and are set as default in the SpeedUP method. The proposed hybrid method is tested with well-known classical machine learning algorithms: multinomial naïve Bayes, logistic regression, linear support vector machine, decision tree and random forest. The results obtained are compared with a focus on evaluating classification accuracy and their significance is tested with Welch's t-test, which is used in statistics to test the hypothesis. The hybrid method is employed and experimental research is conducted with Python programming language (v3.6.2) and scikit-learn[1] (v0.19.2) – library for machine learning.

The dissertation consists of four chapters, general conclusions, and a list of references and appendices. The scope of the dissertation is 170 pages including 55 tables and 33 figures. The list of references contains 212 various sources, including books, scientific papers, patents, technical reports and Internet sources.

The results of this research were presented at three international and two national scientific conferences. Three papers have been published in reviewed journals (one journal is included in ISI Web of Science database), four in periodical conference proceedings and one abstract in conference abstracts proceedings.

[1] `https://scikit-learn.org/`

SANTRAUKA

Nuomonių analizė tapo populiari tada, kai žmonės pradėjo naudotis internetu. Šiais laikais, jei norima susidaryti nuomonę apie apklausas, socialinius, ekonominius ir kt. įvykius, jos ieškoma internete. Pagrindinis tyrimų, susijusių su nuomonių analize, tikslas – gauti nuomonę, išreikštą teigiamais ar neigiamais komentarais. Disertacijoje nagrinėjami mašininio mokymosi metodai, skirti tekstinių duomenų nuomonių analizei didelės apimties duomenyse. Tyrimo tikslas – pasiūlyti hibridinį nuomonių analizės metodą su rekomenduojamų parametrų rinkiniu dideliems tekstiniams duomenims, kurio vykdymo greitis būtų spartesnis, o klasifikavimo tikslumas – panašus, lyginant su klasikiniais metodais. Pasiūlytą hibridinį metodą sudaro keturios dalys: klasikinis mašininio mokymosi algoritmas, k-Means klasterizavimas, dalelių spiečiaus optimizavimo metaeuristika ir ansamblis. Šios dalys yra integruotos į spartinimo metodą SpeedUP, kuris automatiškai vykdo visas pasiūlyto hibridinio metodo dalis, priklausomai nuo nurodytų parametrų, kurie rekomenduojami šiame tyrime bei nustatyti kaip numatytieji SpeedUP metode. Pasiūlytas hibridinis metodas yra testuojamas su gerai žinomais klasikiniais mašininio mokymosi algoritmais: daugialypiu naiviuoju Bajesu, logistine regresija, tiesine atraminių vektorių mašina, atsitiktiniu mišku ir sprendimu medžiu. Gauti rezultatai lyginami vertinant tekstinių duomenų klasifikavimo tikslumą, o gauto tikslumo reikšmingumas yra įvertinamas atliekant Welcho t-testą, kuris statistikoje naudojamas hipotezėms tikrinti. Hibridiniui metodui kurti ir eksperimentiniam tyrimui naudojami Python programavimo kalba (v3.6.2) ir scikit-learn[1] (v0.19.2): mašininio mokymosi biblioteka.

Disertaciją sudaro 4 skyriai, bendrosios išvados, literatūros sąrašas ir priedai. Disertacijos apimtis - 170 puslapių, 55 lentelės ir 33 paveikslai. Literatūros sąrašą sudaro 212 įvairių šaltinių, įskaitant knygas, mokslinius straipsnius, patentus, technines ataskaitas ir interneto šaltinius.

Tyrimo rezultatai pristatyti trijose tarptautinėse konferencijose ir dviejose konferencijose Lietuvoje. Disertacijos tema paskelbti trys straipsniai recenzuojamuose žurnaluose (1 žurnalas yra įtrauktas į ISI Web of Science duomenų bazę), keturi straipsniai – periodiniuose konferencijų leidiniuose, 1 santrauka konferencijų santraukų leidinyje.

ACKNOWLEDGEMENTS

I would like to express my very great appreciation to my scientific supervisor Prof. Dr. Gintautas Garšva for his patience, motivation and for the guidance throughout the research and writing of this dissertation.

I would also like to thank the reviewers Prof. Dr. Gintautas Dzemyda and Dr. Virginijus Marcinkevičius for their valuable and constructive commentaries.

I wish to acknowledge the contributions to my papers' co-author Dr. Paulius Danėnas for advice and discussions which led to this research.

I would like to thank my family for their support, patience and understanding throughout my study.

And finally, I wish to thank all the people who were directly or indirectly involved in the preparation of this dissertation.

Konstantinas Korovkinas

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACC	Accuracy
AmazonProduct	Amazon product data dataset
AmazonTest	Amazon customer reviews dataset
AUC	Area under the receiver operating characteristics
Books	Books dataset
Bopt	Bayesian optimization
CFM	Contextual factorization machine
CNN	Convolutional neural network
CV	Cross-validation
DNN	Deep neural network
DT	Decision tree
Electronics	Electronics dataset
Event	Event dataset
Expr	Expression
F_1score	Harmonic mean of PPV and TPR
FM	Factorization machine
GridSearchCV	Cross-validated grid search
k-Means	K-means clustering
KindleStore	Kindle store dataset
LabeledData	Manually selected and labeled data
LR	Logistic regression
LR-BoW	Logistic regression with bag-of-words
LR-WE	Logistic regression with word embeddings
LSVM	Linear support vector machine
MaxEnt	Maximum entropy
ML	Machine learning
MNB	Multinomial naïve Bayes
NB	Naïve Bayes
NLP	Natural language processing
NN	Neural network
NPV	Precision. Negative predictive value
Person	Public person dataset
PFM	Position-aware factorization machine
Phones&Accessories	Cell phones and accessories dataset
POS	Part of speech tagging
PPV	Precision. Positive predictive value
PSO	Particle swarm optimization

RF	Random forest
RS	Random search
SA	Sentiment analysis
sentiment140	Stanford Twitter sentiment corpus dataset
SGD	Stochastic gradient descent
Source1	Vilmorus ltd
Source2	Baltic Surveys
SVM	Support vector machine
SVM-BoW	Support vector machine with bag-of-words
SVM-Poly	Support vector machine with the poly kernel
SVM-WE	Support vector machine with word embeddings
TF-IDF	Term frequency-inverse document frequency
TNR	Recall. True negative rate
TPR	Recall. True positive rate

NOTATION

α	Significance level ($\alpha = 0.05$)
μ	Mean
$\overline{d}$	Mean difference between two groups
C	Penalty (cost) parameter of the error term
SE	Estimated standard error of the mean
$Split_{ratio}$	Variable calculated dependently on training and testing data split ratio expressed in percentage – ($Training\_data(\%)/Testing\_data(\%)$)
SQ	Subsets quantity
Ss	Subset size
S	Standard deviation
t_{table}	Value from the t-distribution table
TDs	Testing data size

1. INTRODUCTION

1.1 Research context

Textual data sentiment analysis (SA) became very popular when people started using the Internet, to be more concrete when e-shops and social networks, blogs and other platforms appeared where people could write their comments. Nowadays if you want to get an opinion about surveys, social, economic and others events, you can find all information you need on the Internet. The main goal of research related to sentiment analysis is to obtain authors' feelings expressed in positive or negative comments. This analysis is performed at multiple levels: document, sentence, and aspect. According to Pang and Lee [1], the term "sentiment" appeared in papers [2, 3] in 2001 and subsequently in papers [4, 5] in 2002. Opinion mining is another term in certain respects parallel to sentiment analysis that appeared in the 2003 paper by Dave et al. [6]. They described an ideal opinion-mining tool that "would process set of search results for the given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions about each of them (poor, mixed, good)" [6]. Liu [7] gives a definition of sentiment analysis. He described it as "the field of study that analyzes people's opinions, sentiments, appraisals, attitudes, and emotions toward entities and their attributes expressed in written text". According to him, this field represents a huge problem space due to "many related names and slightly different tasks". "Sentiment analysis, opinion mining, opinion analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, and review mining", according to Liu, are "all under the umbrella of sentiment analysis". Basically, sentiment analysis is divided into lexicon-based methods and machine learning (ML) methods [8]. The authors also mixed aforementioned methods to achieve better results. The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document [4, 9]. At the present time the popularity of sentiment analysis has greatly increased and is still growing, because of a huge amount of text data, which is accessible on the Internet and could be very useful for companies, users etc. The usage of lexicon-based methods for a huge amount of data extracted from social media websites is not very effective, because of their unstructured format and the data can contain textual peculiarities, informal and dynamic nature of language, new slang, abbreviations, and new expressions [10]. Also, it could significantly increase computational costs. However, this dissertation investigates sentiment classification using machine learning methods.

1.2 Research problem

Sentiment analysis of text is considered as a very challenging area: although a lot of work has been done in this field, accuracy (ACC) is still rather average due to comments, slang, smiles etc. We need to understand the whole context of the sentence, because even a single word can change the polarity of a sentence, and this might have a significant impact especially in currently sensitive domains such as medicine, business etc. Another problem which is faced in this area is a huge data amount. If a customer needs to get an opinion about a product, hotel, flight etc. – it becomes a very hard task. In the case of a large volume of data the performance of machine learning algorithms decreases depending on the dataset size – the higher the number of features is, the longer computation time it requires. In order to solve this problem, researchers use various techniques: parallelism, implementation on graphics processor unit (comparison between graphics processor unit and central processing unit presented in [A6]), using cloud computing technology, selecting only representative data for training etc. This leads to another problem – the need for special hardware or a more powerful computer, cloud provider with required software etc. This dissertation investigates the problem of big computational costs achieved by machine learning algorithms on large scale textual datasets. Therefore, the main focus is on increasing execution speed without or with a slight loss of accuracy and without the need of special software or hardware.

1.3 Object of the research

The main object of this research is the textual data classification methods, their execution speed and accuracy in large scale datasets, sentiment analysis.

1.4 Goal and objectives of the research

The goal of the research is to propose a hybrid sentiment analysis method with a recommended set of parameters for large textual data with a better execution time and with a similar classification accuracy compared with classical methods.

The objectives of the dissertation are:

1. To propose a hybrid method that increases the classification speed of the selected classical machine learning algorithms, which are commonly used for textual data sentiment analysis with a similar classification accuracy.

2. To perform the experimental evaluation of the method proposed and select the recommended set of parameters for it, as well as to improve classification accuracy.
3. To perform a comparison of the proposed hybrid method with other authors' works on large scale textual datasets and evaluate obtained results.

1.5 Research methodology and tools

The following methods were used:

1. Bibliographic research on the sentiment analysis field helped to formulate the research tasks and goals.
2. Analysis of related works helped to select machine learning algorithms for the proposed hybrid method.
3. Proposed hybrid method for textual data sentiment analysis is described in Chapter 3.
4. Experimental research methodology and experimental research for comparative analysis of the proposed method are described in Chapter 4.
5. Formulation of conclusions after each chapter and general conclusions at the end of research.

For the proposed hybrid method development and performing experimental research the following were used: Python programming language and scikit-learn [11] – library for machine learning. For the dissertation LaTeX[2], a document preparation system was used; for the presentation of graphical results and diagrams latex Tikz[3] package was employed.

1.6 Scientific novelty

In this dissertation a hybrid method for textual data sentiment analysis with a recommended set of parameters suitable for large scale datasets is proposed. The method consists of the following:

1. SpeedUP method – this is the main part of the proposed hybrid method, whose aim is to increase the classification speed of classical machine learning algorithms.
2. k-Means clustering method – this method is responsible for training data selection.
3. PSO tuning method – this method performs hyperparameters tuning for linear support vector machine (LSVM).

[2] `https://www.latex-project.org/`

[3] `https://www.overleaf.com/learn/latex/TikZ_package`

4. Ensemble method – this is the last part of the proposed hybrid method, which performs combination and voting of the machine learning algorithms.

In this dissertation, machine learning algorithms for textual data sentiment analysis were reviewed and five most common were selected. Effectiveness metrics, average ranking and statistical significance were performed with selected machine learning algorithms.

Based on experimental results the recommended set of parameters for the proposed hybrid method was selected and presented. The results showed that the proposed method increased the classification speed of classical machine learning algorithms on large scale textual datasets with slight loses in terms of accuracy; it is also competitive with state-of-the-art methods.

Unlike the techniques proposed previously, SpeedUP automatically performs all parts of the proposed hybrid method, based on specified parameters, which are recommended in this research and are set as default in the SpeedUP method. Depending on the determined size of a subset, the size of training data is calculated; depending on it the k-Means method automatically selects the appropriate amount of training data (in the case of an ensemble method it selects as many training datasets as the selected number of classifiers) and passes it to LSVM input; if the k-Means method is switched off datasets will be selected randomly. PSO automatically selects a C parameter for LSVM and its training is performed with this parameter; in the case of ensemble the same number of C parameters and classifiers is also selected. All calculations, training data selection, testing dataset dividing into subsets, tuning of the hyperparameters, joining ML algorithms to ensembles, joining of results and voting are performed automatically by the SpeedUP method.

The proposed method can be applied to classify textual data sentiments and is suitable to work with large scale datasets without using supercomputers.

1.7 Practical significance

Since sentiment analysis of text is still a very challenging area and at the same time it is very widely applicable in practice for product reviews, customer churn prediction, fraud detection, election etc., the proposed hybrid method could be:

1. Successfully applied in these areas for the development of new models or for improving the existing ones.
2. Be suitable for work with large scale textual datasets and allows classifying sentiment without using high-performance computers.

3. Be useful with surveys, because opinions are taken from social networks, articles comments etc., which allows forming different opinions on the current topic.

1.8 Defended statements

1. The proposed hybrid method for textual data sentiment analysis in large scale datasets can increase the classification speed of classical machine learning algorithms such as linear support vector machine, logistic regression (LR), decision tree (DT) and random forest (RF), whereas losses in terms of accuracy are not very great.
2. The best results are achieved when the proposed method with the recommended set of parameters is used with LSVM, compared with those when it is employed with multinomial naïve Bayes (MNB), logistic regression, decision tree and random forest.
3. The proposed hybrid method could be used as an alternative method on large scale textual datasets for classical and state-of-the-art methods which are used by other authors. Moreover, PSO tuning method could be competitive with such popular methods as random search (RS) and Bayesian optimization (Bopt).

1.9 Presentation and approbation of the results

International conferences

1. The Symposium for Young Scientists in Technology, Engineering and Mathematics (SYSTEM 2018), The 23th Conference for Master and PhD students, Gliwice, Poland, 2018.
2. The 24th International Conference on Information and Software Technologies (ICIST 2018), Kaunas, Lithuania, 2018.
3. The Baltic DB&IS 2020 Doctoral Consortium, Tallinn, Estonia, 2020.

National conferences

1. Information Technologies (IT 2018), The 23th Conference for Master and PhD students, Kaunas, Lithuania, 2018.
2. 10th International Workshop Data Analysis Methods for Software Systems (DAMSS 2018), Druskininkai, Lithuania, 2018.

Publications

Papers in periodic scientific journals:

[A1] Korovkinas, K., Danėnas, P., Garšva, G., 2017. SVM and Naïve Bayes Classification Ensemble Method for Sentiment Analysis. Baltic Journal of Modern Computing, 5(4), pp. 398–409.

[A2] Korovkinas, K., Danėnas, P., Garšva, G., 2019. SVM and k-Means Hybrid Method for Textual Data Sentiment Analysis. Baltic Journal of Modern Computing, 7(1), pp. 47–60.

[A3] Korovkinas, K., Danėnas, P., Garšva, G., 2020. Support Vector Machine Parameter Tuning Based on Particle Swarm Optimization Metaheuristic. Nonlinear Analysis: Modelling and Control, 25(2), pp. 266–281.

Papers in peer-reviewed scientific conference proceedings:

[A4] Korovkinas, K., Danėnas, P., Garšva, G., 2018. SVM Accuracy and Training Speed Trade-Off in Sentiment Analysis Tasks. In International Conference on Information and Software Technologies, Springer, Cham, pp. 227–239

[A5] Korovkinas, K., Garšva, G., 2018. Selection of Intelligent Algorithms for Sentiment Classification Method Creation. Proceedings of the International Conference on Information Technologies, Vol–2145, Kaunas, Lithuania, pp. 152–157, ISSN 1613-0073, CEUR. Available: `http://ceur-ws.org/Vol-2145/p26.pdf`

[A6] Vaitonis, M., Masteika, S., Korovkinas, K. 2018. Algorithmic Trading and Machine Learning Based on GPU. Proceedings of the Symposium for Young Scientists in Technology, Engineering and Mathematics, Vol–2147, Gliwice, Poland, pp. 49–54, ISSN 1613-0073, CEUR. Available: `http://ceur-ws.org/Vol-2147/p09.pdf`

[A7] Korovkinas, K. 2020. A Hybrid Method for Textual Data Classification Based on Support Vector Machine with Particle Swarm Optimization Metaheuristic and k-Means Clustering. Proceedings of Baltic DB&IS 2020 Doctoral Consortium, Vol–2620, Tallinn, Estonia, pp. 81–88, ISSN 1613-0073, CEUR. Available: `http://ceur-ws.org/Vol-2620/paper11.pdf`

Abstracts in conference proceedings:

1. Korovkinas, K., Garšva, G., 2018. Large Scale Sentiment Analysis Using NLP Based Feature Extraction Technique and PSOLinearSVM. Data analysis methods for software systems: 10th international workshop, Druskininkai, 2018. ISBN 978-609-07-0043-3. Available: `https://www.journals.vu.lt/proceedings/article/view/12634/11171`

1.10 Structure of the dissertation

The dissertation consists of four chapters, general conclusions, a list of references and appendices. The scope of dissertation is 170 pages including 55 tables and 33 figures. The list of references contains 212 various sources, including books, scientific papers, patents, technical reports and Internet sources.

2. TEXTUAL DATA SENTIMENT ANALYSIS USING MACHINE LEARNING

In this chapter, machine learning algorithms, which are most commonly used for textual data sentiment classification are reviewed. According to reviews, five machine learning algorithms will be selected for experiments. Later the problems related to the selected machine learning algorithms, which other authors often attempt to solve will be defined, as well as training data reduction, hyperparameters optimization and ensemble methods. Finally, natural language processing, which is also very important in textual data sentiment analysis is presented. Parts of this chapter are published in [A1],[A2],[A3],[A4],[A5],[A6],[A7].

2.1 Sentiment analysis

Traditionally, sentiment classification can be regarded as a binary-classification task [5, 6]. According to Pang et al. [5], sentiment analysis is the extraction of positive or negative opinions from text. The main goal of sentiment analysis is to extract sentiments from the text. Sentiment analysis is performed at multiple levels: document, sentence, and aspect. The task at document level is to classify whether a whole opinion document expresses a positive or negative sentiment [4, 5]. At sentence level the task goes to the sentences and determines whether each sentence expressed a positive, negative, or neutral opinion. Neutral opinion usually means no opinion [7] at all. Instead of looking at language constructs (documents, paragraphs, sentences, clauses or phrases) aspect level examines the opinion itself. It is based on the idea that an opinion consists of a sentiment (positive or negative) and a target [7]. According to the survey performed by the authors [12], the most popular is document level, then comes aspect level and sentence level is the last. SA tasks mainly have two approaches: feature extraction and sentiment classification.

Dave et al. [6] use structured reviews for testing and training, identifying appropriate features and scoring methods from information retrieval for determining whether reviews are positive or negative. The results achieved are as good as those when traditional machine learning is employed where the classifier identifies and classifies review sentences from the web, but classification is more difficult [13]. The authors [1, 7] presented an overview in sentiment analysis in which the strong points and the weak points of sentiment analysis are examined and they gave many research ways of sentiment analysis. They overview problems, challenges and tasks, like sentiment extraction, classification, summarization and polarity determina-

tion. There are also many surveys which present challenges and difficulties of SA [14, 15, 16]. Feldman [17] focused on five specific problems in SA field: document-level sentiment analysis, sentence-level sentiment analysis, aspect-based sentiment analysis, comparative sentiment analysis and sentiment lexicon acquisition. He presented problems related to SA and some of the techniques used to solve them. Additionally, the author reviewed some of the major application areas where sentiment analysis is being used and open research problems. Qazi et al. [18] performed a systematic review of literature on the practices and challenges of SA. The review shows that various techniques have been developed for opinion summarization, drawing their methods from the fields of artificial intelligence, statistics and linguistics, where each technique possesses a certain focus as well as having its particular strengths and weaknesses.

Lately interest in sentiment analysis has increased. The reason is a huge amount of data available on the Internet, which grows dramatically every day. According to the International Data Corporation forecast, about 175 ZB of data until 2025 will be generated, while in 2011 there was approximately 2 ZB of data. The reason is the increased usage of technology devices. Dhaoui et al. [19] performed comparison of lexicon versus machine learning social media sentiment analysis. They stated that in the case of big data manual approaches to sentiment analysis are impractical and raise the need to develop automated tools to analyze consumer sentiment expressed in text format. They also established that the results of lexicon based and machine learning approaches are similar in accuracy, while their combination significantly improved accuracy. This led to the conclusion that machine learning is an inseparable part of sentiment analysis. The primary role of machine learning in sentiment analysis is to improve and automate the low-level text analytical functions that sentiment analysis relies on, including part of speech tagging (POS).

2.2 Machine learning in sentiment analysis

Machine learning algorithms are one part of sentiment analysis. A lot of work has been done and many classifiers have been compared for SA tasks. Pang et al. [5] evaluated the performance of naïve Bayes (NB), maximum entropy (MaxEnt), and support vector machines (SVM) in the specific domain of movie reviews, obtaining accuracy slightly above 80%. Go et al. [20] later obtained similar results with unigrams by introducing a more novel approach to automatically classify the sentiment of Twitter messages as either positive or negative with respect to a query term. The same techniques were also used by Kharde and Sonawane [21] to perform sentiment analysis on Twitter data, yet resulting in lower accuracy; again, SVM proved to perform best. Davidov et al. [22] also stated that SVM and NB are the best

techniques to classify the data and can be regarded as the baseline learning methods by applying them for analysis based on the Twitter user defined hashtag in tweets. Kapočiūtė-Dzikienė et al. [23] used knowledge-based and machine learning approaches for sentiment classification into positive, negative and neutral on Lithuanian internet comments. Support vector machine and multinomial naïve Bayes significantly outperform their knowledge-based method. Le and Nguyen [24] proposed a sentiment analysis model based on NB and SVM; for feature extraction they applied information gain, bigram and the object-oriented extraction method in order to analyze sentiment more effectively. Gautam and Yadav [25] applied naïve Bayes, maximum entropy and support vector machine along with the semantic analysis to classify the sentence and product reviews based on Twitter data into positive and negative. NB showed higher accuracy of 88.2% than SVM (85.5%) and MaxEnt (83.8%). After semantic analysis was applied, they improved accuracy from 88.2% to 89.9%. Kolchyna et al. [26] presented a new ensemble method that uses a lexicon based sentiment score as input feature for the machine learning approach and applied it on the benchmark Twitter dataset using three machine learning algorithms: SVM, DT and NB. The results showed that when only n-grams were used as the features, SVM achieved 86.62%, NB – 81.5% and DT – 80.57% accuracy. After the new method was applied the results increased as follows: SVM – to 91.17%, DT – to 89.9% and NB – to 88.54%. Kanakaraj et al. [27] also presented the semantics-based feature vector with ensemble classifier for Twitter data sentiment analysis. They reported that their method outperformed by 3%-5% the traditional bag-of-words approach with single machine learning algorithms like naïve Bayes, maximum entropy, support vector machine, decision tree, random forest, extremely randomized trees and decision tree regression with adaBoost. Wan and Gao [28] used naïve Bayes, support vector machine, Bayesian network, C4.5 decision tree and random forest algorithms to create an ensemble method based on the majority vote principle of multiple classification methods and applied it for sentiment classification on Twitter data for airline services. Amolik et al. [29] achieved 75% accuracy with SVM and 65% with NB on Twitter sentiment analysis of movie reviews classifying tweets as positive, negative and neutral. Tripathy et al. [30] applied naïve Bayes, maximum entropy, stochastic gradient descent (SGD) and support vector machine on movie review dataset, using n-gram approach and its various combinations for sentiment classification into positive or negative. The best results were obtained when the following were used: "unigram+bigram+trigram" with SVM 88.94% or NB 86.23%; "unigram+bigram" with SVM 88.88% or MaxEnt 88.42%; "unigram" with MaxEnt 88.48% or SVM 86.97%. The authors [31, 32, 33] reviewed machine learning techniques for sentiment analysis. The well-known techniques like MaxEnt, SailAil sentiment analyzer, multilayer perceptron, naïve Bayes,

multinomial naïve Bayes, support vector machine and random forest were discussed and accuracy was compared on different datasets. Pranckevičius and Marcinkevičius [34] investigated naïve Bayes, random forest, decision tree, support vector machines, and logistic regression classifiers implemented in Apache Spark and identified the optimal number of n-grams to get the highest accuracy. The technique applied on Amazon customers' product-review data for Android apps. Rathor et al. [35] also applied support vector machines, naïve Bayes and maximum entropy for classification Amazon reviews into positive, neutral and negative. NB produced the bests results with unigrams – 66.84%; however, SVM showed better results with weighted unigrams – 81.20%. Manikandan and Sivakumar [36] reviewed the principles, advantages and applications of document classification, document clustering and text mining, focusing on the existing literature. According to approaches on machine learning, the common algorithms for text classification are: naïve Bayes classifier, support vector machine, decision tree, Rocchio algorithm, k-nearest neighbor, decision rules classification, artificial neural network, fuzzy correlation and genetic algorithm. Manikandan and Sivakumar concluded that support vector machine, naïve Bayes, k-nearest neighbor and their hybrid system with the combination of different other algorithms turned out the most appropriate. Ogutu et al. [37] explored the use of a detailed pre-processing technique with the implementation of NB and SVM classifiers on product reviews. Finally, the results were compared between both classifiers. According to them, NB (98.40%) results were better with up to 2000 reviews compared with SVM (97.50%), while SVM (97.60%) outperformed NB (93.80%) in the case of 3000 reviews. Kandhro et al. [38] proposed a model implemented on multinomial naïve Bayes, stochastic gradient decent, linear support vector machine, random forest and multilayer perceptron classifier approaches for student feedback sentiment analysis. They reported that the best choice for their research is MNB (83%) and multilayer perceptron classifier (83%). The results of others classifiers are as follows: stochastic gradient decent (79%), support vector machine (80%) and random forest (72%). Kapočiūtė-Dzikienė et al. [39] performed a comparative analysis of the traditional machine learning (SVM and MNB) and deep learning methods (long short-term memory and convolutional neural network (CNN)) to solve the sentiment analysis task for the Lithuanian language. They reported the superiority of the traditional machine learning methods with MNB accuracy of 73.5% and SVM – 72.4%, compared to CNN – 70.6% and long short-term memory – 61.7%. Shamantha et al. [40] performed sentiment analysis of the tweets or reviews published in Twitter. Performance was evaluated with random forest, naïve Bayes and support vector machine. SVM and NB resulted in higher accuracy, while NB – in the faster execution time. Gupta et al. [41] performed sentiment analysis of Twitter posts by using DT, LR, SVM and neural net-

work (NN). They reported that the highest accuracy is achieved by neural network – 79.6%. According to the results presented by the authors in [16], the machine learning techniques for solving SA of scientific citation issues are used more often than other techniques as lexicon based, deep learning and keyword based. The SVM and NB classifiers have been the most frequently used methods for performing SA. Kumar et al. [42] applied maximum entropy, support vector machine, convolutional neural network, and long short-term memory to study the impact of age and gender on user reviews. They demonstrated that in comparison on the basis of age SVM and CNN performed with higher accuracy of 78%, while in comparison on the basis of gender the best accuracy was achieved by CNN – 80%. Dang et al. [43] performed a comparative study in sentiment analysis based on deep learning. After the analysis of 32 papers they concluded that deep neural networks (DNN), CNN and hybrid approaches are most widely used. The experiments were performed on eight textual datasets. Average accuracy of the DNN model is 79.54% and of CNN – 78.88%. Kumar and Jaiswal [44] performed a systematic literature review of sentiment analysis on Twitter. Based on their approach the mostly used sentiment classification techniques are: SVM, NB, LR, DT, k-nearest neighbor, NN, CNN, fuzzy logic etc.

Also, a numbers of inventions have been devised in the sentiment analysis field. Huang et al. [45] presented a system, which applies both full text and complex feature analyses to sentences of a product review. Each analysis is weighted prior to linear combination into a final sentiment prediction. A conditional random field framework is used to enhance sentiment prediction. Xia [46] presented a method for classifying social media text data by establishing a more accurate classifier by selecting it from eight machine learning methods: SVM, MaxEnt, tree, bagging tree, boosting tree, RF, neural network and NB. Li et al. [47] invented a text sentiment classification method to improve text sentiment classification accuracy. A Bayes classifier model is used to classify the text according to the probability of each word appearing in the text of each polarity. Wang et al. [48] provided a method for classifying text messages in accordance with sentiment and/or emotion expressed by the text messages and a method for handling ambivalence or hidden sarcasm in text messages. Chatterjee et al. [49] presented a hybrid human machine learning system with tagging and scoring techniques for sentiment magnitude scoring of textual passages. They concluded that "the combination of machine learning systems with data from human pooled language extraction techniques enable the present system to achieve high accuracy of human sentiment measurement and textual categorization of raw text, blog posts, and social media streams".

Table 2.1 shows the summarized results of related work reviews. The most common used machine learning algorithms are selected based on aforementioned reviews and according to the authors [12, 16, 18, 44, 50].

The averaged accuracy presented in the table is calculated by the author according to related work reviews.

Table 2.1. Averaged accuracy of machine learning algorithms, based on related work reviews

Method	SVM	NB	LR (MaxEnt)	RF	DT	DNN	CNN	NN	Fuzzy logic
Accuracy (%)	**86.94**	**82.22**	**86.46**	**86.24**	**87.96**	79.54	78.60	79.60	64.28

Based on the accuracy results presented in Table 2.1, five machine learning algorithms were selected: SVM, NB, LR, RF and DT for testing with the proposed hybrid method.

2.3 Relevant methods and reviews

In this section multinomial naïve Bayes, logistic regression, linear support vector machines, random forest (decision tree), term frequency-inverse document frequency (TF-IDF), k-Means clustering (k-Means) and particle swarm optimization (PSO) are briefly described. It was decided to use classical methods with the intention of easier implementation of parameter tuning and altering model designs because they are quite easy to interpret and understand. In the case of state-of-the-art methods, they are very often "black box" and even researchers do not fully understand their "inside" due to the lack of a theoretical foundation.

2.3.1 Multinomial naïve Bayes

Multinomial naïve Bayes is one of the widely used methods in text classification. Its popularity is based on easier implementation and high computational speed. Authors, who work in this field, have proposed various techniques. Rennie et al. [51] reported that NB has systematic errors as well as problems that arise because text is not actually generated according to a multinomial model. They proposed a fast and easy way to solve these problems. Modified naïve Bayes contains corrected deficiencies in the application, performance improvement, the problem of uneven training data solution and handling of word occurrence dependencies. Kibriya et al. [52] presented an empirical comparison of several MNBs on text categorization problem, comparing them to LSVM. They found out that MNB can be improved by applying TF-IDF transformation and normalization. Su et al. [53] proposed a simple semi-supervised extension of MNB, called semi-supervised frequency estimate for large scale text classification and showed

that their method improved MNB with additional data (labeled or unlabeled) in terms of area under the receiver operating characteristics (AUC) and accuracy. Jiang et al. [54] presented a novel method, which alleviates the attribute independence assumption by averaging all of the weighted one-dependence multinomial estimators and reported the superiority to MNB. Mowafy et al. [55] presented the MNB method with TF-IDF and resulted in better accuracy for text document classification. Abbas et al. [56] presented a text classification framework based on MNB and TF-IDF. They succeeded in improving NB performance with standardized categorization and management of the dependence of word occurrence. Chen et al. [57] improved classical NB by adding a correlation factor to it that incorporates overall correlation among the different classes and resulted in better accuracy compared with the classical NB. Ruan et al. [58] proposed a new class-specific deep feature weighting method for MNB. Their method, in addition to feature weighting, estimates the conditional probabilities of the text classifier by deeply computing feature weighted frequencies from training data. Balakrishnan and Kaur [59] used string-based MNB for emotion classification among the Facebook diabetes community and reported that their method outperformed Emolex, NB and MNB. Farisi et al. [60] provided a solution by classifying positive opinion reviews and negative opinions using the MNB classifier method and comparing models using preprocessing, feature extraction and feature selection.

Multinomial naïve Bayes classifier. Naïve Bayes classifier is a simple probabilistic classifier based on Bayes' theorem and is particularly suited when the dimensionality of the inputs is high. The naïve Bayes classifier uses the Bayes' rule [61]

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)} \tag{2.1}$$

where $P(C|D)$ is the posterior probability of class C given predictor D (a document vector), $P(C)$ is the prior probability of class C, $P(D|C)$ is the likelihood which is the probability of predictor given class, $P(D)$ is the prior probability of predictor.

Since in classification tasks might be not only binary classes, then let's denote a set of classes $C = \{c_1, \ldots, c_m\}$, where m is the total number of classes. Let $D = (d_1, \ldots, d_k)$ (k is the total number of documents) be a given documents vector that belongs to class c_i. Let's assume that all attributes are independent given the value of the class variable. By substituting for D

the equation 2.1 could be rewritten [61]:

$$P(c_i|d_1,\ldots,d_k) = \frac{P(d_1|c_i)\ldots P(d_k|c_i)P(c_i)}{P(d_1)\ldots P(d_k)} = \frac{\left(\prod_{j=1}^{k} P(d_j|c_i)\right) P(c_i)}{P(d_1)\ldots P(d_k)} \tag{2.2}$$

Since $P(d_1)\ldots P(d_k)$ is constant the equation 2.2 can be simplified:

$$P(c_i|d_1,\ldots,d_k) = \left(\prod_{j=1}^{k} P(d_j|c_i)\right) P(c_i) \tag{2.3}$$

In this dissertation, the multinomial naïve Bayes, widely used for document classification, is employed. A document is treated as a sequence of words and it is assumed that each word position is generated independently of every other. $P(c_i)$ can be estimated by dividing the number of documents belonging to c_i by the total number of documents.

The probability to obtain document d_j in class c_i:

$$P(d_j|c_i) = (\sum_n f_{ni})! \prod_n \frac{P(w_n|c_i)^{f_{ni}}}{f_{ni}!} \tag{2.4}$$

where f_{ni} is the count of words n in document d_j, $P(w_n|c_i)$ is the probability of word n given class c_i. According to the authors in [52], $(\sum_n f_{ni})!$ and $\prod_n f_{ni}!$ could be deleted from equation 2.4 without any changes in the results, because neither depends on the class c_i:

$$P(d_j|c_i) = \prod_n P(w_n|c_i)^{f_{ni}} \tag{2.5}$$

Then equation 2.3 could be rewritten for multinomial naïve Bayes as follows:

$$P(c_i|d_1,\ldots,d_k) = \left(\prod_n P(w_n|c_i)^{f_{ni}}\right) P(c_i) \tag{2.6}$$

2.3.2 Logistic regression

The works on logistic regression mainly focused on increasing accuracy. Hamdan et al. [62] used logistic regression in combination with conditional random field and opinion target expression. Logistic regression was used for sentiment polarity. They reported improved accuracy when their method was used. Byrne and Schniter [63] proposed approximate message passing approaches to sparse multinomial logistic regression for the problem of multi class linear classification and feature selection. They stated improved

error-rate and runtime performance on synthetic and real-world datasets. Rafeek and Remya [64] proposed a method by combining syntactic rules and LR and reported improved classification accuracy. Wang et al. [65] proposed fast subsampling algorithms to efficiently approximate the maximum likelihood estimate in logistic regression. The results obtained on synthetic and real datasets indicate that the algorithm is computationally efficient and has a significant reduction in computing time compared to the full data approach. Zhang et al. [66] proposed a method for logistic regression accuracy improvement on imbalanced datasets based on the probability threshold and achieved an accuracy rate up to 95%. Zhu et al. [67] improved LR by integrating principal component analysis for dimensionality reduction and k-Means for clustering, and showed that the method performed at an improved level in predicting diabetes onset. Mai et al. [68] evaluated the asymptotic distribution of the logistic regression classifier and consequently, provided the associated classification performance. According to them, "this brings new insights into the internal mechanism of logistic regression classifier, including a possible bias in the separating hyperplane, as well as on practical issues such as hyperparameter tuning". Zhang et al. [69] proposed a variational Bayesian method for identifying important variables in high-dimensional logistic regression models and showed its effectiveness on synthetic and some publicly available datasets. Okabe et al. [70] proposed f-measure maximizing logistic regression using the relative density ratio. They reported a better performance of the their method in the class imbalanced real data example.

Logistic regression is a linear model. The logistic regression model uses a logistic function to squeeze the output of a linear equation between 0 and 1. The probabilities describing the possible outcomes of a single trial are modeled using a logistic function. A simple logistic function is defined by the formula [71]:

$$sigm(y) = \frac{e^y}{e^y + 1} \tag{2.7}$$

where $sigm(y)$ refers to the sigmoid function, which output is between 0 and 1 (probability estimate), y input to the function, e base of natural log.

Let p be the probability that a value occurs, then $1 - p$ is the probability that it does not occur. The odds is the ratio of these probabilities [72]:

$$odds = \frac{p}{1 - p} \tag{2.8}$$

After taking the natural logarithm of the odds, a linear relationship between the transformed variable and the explanatory variables can be

established, which is called logistic transformation [72]:

$$logit(p) = \ln\left(\frac{p}{1-p}\right) \tag{2.9}$$

The logistic model predicts the *logit* of class C from document D. With logistic regression the mean of the response variable C in terms of an explanatory variable D is modeled relating C and D through the equation $C = \alpha + \beta D$. With LR we model the natural log odds as a linear function of the explanatory variable. The simple logistic model has the form [73]:

$$logit(C) = \ln\left(\frac{p}{1-p}\right) = \alpha + \beta D \tag{2.10}$$

where α is the constant of the equation, β is a coefficient of predictor variables. Then extending the logic of the simple logistic regression to multiple predictors could be written [73]:

$$logit(C) = \ln\left(\frac{p}{1-p}\right) = \alpha + \beta_1 D_1 + \cdots + \beta_n D_n \tag{2.11}$$

Taking the antilog of equation 2.10 on both sides, one can derive an equation for the prediction of the probability of the occurrence of interested outcome as [73]:

$$\frac{p}{1-p} = e^{\alpha+\beta D} \quad => \quad p = P(C|D) = \frac{e^{\alpha+\beta D}}{1+e^{\alpha+\beta D}} \tag{2.12}$$

And for equation 2.11:

$$p = P(C|D) = \frac{e^{\alpha+\beta_1 D_1+\cdots+\beta_n D_n}}{1+e^{\alpha+\beta_1 D_1+\cdots+\beta_n D_n}} \tag{2.13}$$

where $P(C|D)$ is the probability of class C given the documents D (the posterior probability).

2.3.3 Linear support vector machines

Support vector machine was introduced in papers [74, 75] and later extensively described in [76]. According to a number of authors, who worked with SVM, this method proved its efficiency in solving difficult tasks in various domains for: recognition of regulatory DNA sequences (Damaševičius [77]); EEG data classification (Martišius et al. [78]); classification of images ([79, 80]); credit risk evaluation (Danėnas and Garšva [81]); sensor multifault diagnosis (Deng et al. [82]); monitoring metal-oxide surge arrester conditions (Hoang et al. [83]); multi-class parkinsonian disorders classification (Morisi et al. [84]); forecasting stock market movement direction (Ren

et al. [85]); sentiment analysis (Liu and Lee [86], Chen and Zhang [87]) etc. Support vector machine is one of the most frequently used machine learning algorithms to solve a sentiment classification problem [33, 36]. Wang et al. [88] reported that LSVM achieves the best results consistent with SVM with different kernels including SVM-poly. The authors [89, 90, 91] also reported LSVM efficiency in binary text classification. Unfortunately, manual hyperparameter selection still remains one of the practical application issues, while literature has not provided any heuristic rules or rules of thumb for this task [92]. Hence, it still requires training multiple classifiers with different sets of hyperparameters to obtain satisfiable performance. Damaševičius [77] used SVM for classification of DNA sequences and recognition of the regulatory sequences. The results demonstrated that the selection of the kernel type and its parameters have direct impact on the performance of the SVM and accuracy of the results. Sunkad et al. [93] proposed the best set of features and the SVM hyperparameters for obtaining the best results in human activity recognition. Osman et al. [94] tuned hyperparameters of two machine learning algorithms to improve bug prediction accuracy. They concluded that the k-nearest neighbors algorithm always significantly improved, and the prediction accuracy of support vector machines either improved or was at least retained. Liu and Zio [95] used one synthetic dataset and two real time series data, related to the prediction of wind speed in a region and leakage from the reactor coolant pump in a nuclear power plant and proposed the preferable choice for tuning SVM hyperparameters for recursive multi-step-ahead prediction. Han et al. [96] presented an invention related to a product sale prediction method based on a support vector machine model with parameter optimization, which was performed by selecting a kernel function of the support vector machine and adopting a grid search method to optimize predetermined parameters in the kernel function. According to the authors, "the invention overcomes the defects of poor precision and low calculation efficiency in traditional sale prediction, can provide a relatively accurate sale prediction reference for a decision-making level, and has good application value". Lu et al. [97] presented an invention for optimizing two SVM parameters: C and g, by using an improved particle swarm optimization algorithm. The authors reported that the invention presented "improves the classification accuracy of the SVM classification model, and promotes wider applications of the SVM classification model in the fields of model identification, system control, production scheduling, computer engineering, and electronic communications". Asif et al. [98] focused on the sentiment analysis of social media multilingual textual data to discover the intensity of the sentiments of extremism. They reported that in supervised algorithms, linear support vector classifier resulted in the highest accuracy. Meng et al. [99] reported that SVM is proven to be the more effective classifier to distinguish phage virion protein and non-phage virion protein.

Linear support vector machines. Support vector machines attempt to find the best possible surface to separate positive and negative training samples in supervised manner. Here the focus is on LSVM [100] which is optimized for large-scale learning and, therefore, is used in this dissertation.

Given a set of instance-label pairs (x_i, y_i), $i = 1, \ldots, l$, $x_i \in R^n$, $y_i \in \{-1, +1\}$, a linear classifier generates a weight vector w as the model. The decision function is $sign(w^T x)$. SVM solves the following unconstrained optimization problem [100]:

$$\min_w \{\frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i)\} \tag{2.14}$$

where $C > 0$ is a penalty parameter, w is weight vector, $\xi(w; x_i, y_i)$ is loss function. SVM has two common loss functions $\max(1 - w^T x_i y_i, 0)$ and $\max(1 - w^T x_i y_i, 0)^2$. The former is referred to L1-SVM and L2-SVM.

Then the primal form problem of L1-SVM, with the standard hinge loss is [101]:

$$\min_w \{\frac{1}{2} w^T w + C \sum_{i=1}^{l} \max(1 - w^T x_i y_i, 0)\} \tag{2.15}$$

Since L1-SVM is not differentiable, a popular variation is known as the L2-SVM which minimizes the squared hinge loss[101]:

$$\min_w \{\frac{1}{2} w^T w + C \sum_{i=1}^{l} \max(1 - w^T x_i y_i, 0)^2\} \tag{2.16}$$

L2-SVM is differentiable and imposes a bigger (quadratic vs. linear) loss for points which violate the margin.

2.3.4 Random forest

Decision tree classifier is very attractive because its simplicity. The main focus related to research with decision tree is on classification accuracy and tree size. Some authors have proposed a number of different techniques for finding the best solution. Kim [102] presented a method of semi-supervised decision tree that splits internal nodes by utilizing both labels and the structural characteristics of data for subspace partitioning. The classification results showed that his method found better split points at internal nodes compared to traditional DT. Wu et al. [103] proposed an algorithm for decision tree simplifying by constraining the number of leaf nodes and the number of branches from each splitting. The results demonstrated that the method had generated simpler decision tree and a better accuracy. Tanha et al. [104] concluded that improving the probability estimation of the tree classifiers led to better selection metric for the self-training algorithm and

a better classification model. Better probability estimation was produced, when Laplacian correction, no-pruning, grafting, and NB tree were used. Ooi et al. [105] proposed an inductive temporal learner from two models, which, firstly, builds a temporal tree based on fuzzy cognitive maps and then, secondly, revises the temporal rules and refines the information gain ratio based on the output from first learning model. They achieved low computation time, stable and higher classification accuracy, visibility of temporal classification rules. Saremi et al. [106] presented an improved evolutionary decision tree induction with multi-interval discretization. The most important contribution is made by modification operators that improve or correct tree structure and derive a correct order. Nor et al. [107] achieved more reasonable predictive performance of DT, by using random undersampling to correct the imbalanced data. Cai et al. [108] proposed a fuzzy oblique decision tree method based on an axiomatic fuzzy set theory and fuzzy information entropy. The aim of fuzzy rules is to construct leaf nodes for each class in each layer of the tree; the samples that cannot be covered by the fuzzy rules are then put into an additional node – the only non-leaf node in this layer. The results demonstrated the superiority of the method compared with traditional decision trees. Primartha et al. [109] proposed a PSO based feature selection, combined with a decision tree algorithm for sentiment analysis and reported that the method considerably enhanced the performance of decision tree in comparison with the baseline.

Random forest is an ensemble method which contains the number of decision trees. The most common problems, which researchers are trying to solve are the following: classification accuracy and execution speed. Kulkarni et al. [110] presented a method for increasing random forest accuracy by generating individual decision tree using different split measures: information gain, gain ratio and gini index selected randomly. The results showed the increased accuracy of RF. Chaudhary et al. [111] improved the classification performance of RF by combining random forest machine learning algorithm, an attribute evaluator method and an instance filter method. Alam et al. [112] proposed a feature ranking based methodology, which applied only high ranked features to construct RF and reported the effectiveness of the method on the classification of medical data. Fitri et al. [113] used DT and RF for sentiment analysis of social media Twitter, where they performed worse than NB. Khanvilkar and Vora [114] proposed sentiment analysis and the generation of recommendations according to polarity obtained by SA system. They tested the method with MNB, LR, DT, SVM and RF. The results showed the superiority of RF with 95.03% accuracy. Ansari et al. [115] made an attempt to mine tweets, capture the political sentiments from it and model it as a supervised learning problem. They reported that promising results were achieved with a long short-term memory and random forests. Kumar and Kaur [116] proposed a model for sarcastict

tweet classification based upon random forest, which was observed to be the best among the text classification algorithms, where RF reached 84.7% overall accuracy in comparison with other supervised classification models SVM (78.6%), LR (80.5%), and KNN (73.1%).

Random forest is a collection of decision trees. Random forests were introduced by Breiman in [117] who was inspired by earlier work by Amit and Geman in [118]. A decision tree is built on an entire dataset, using all the features/variables of interest, whereas random forest randomly selects observations/rows and specific features/variables to build multiple decision trees from and then averages results – each tree outputs a probability of each class and then the multiple probability outputs are averaged.

Let's denote D – a document vector and C – a class. Then $P(C|D)$ is the posterior probability of a document vector D to be a member of class C. In the case of decision tree (denoted by θ) it could be written as follows [119]:

$$P(C|D,\,\theta) = \frac{N_C}{N} \tag{2.17}$$

where N is the number of examples at a leaf node classifying D and N_C is the number of examples among N with class C.

Let's denote subset of decision trees $\Theta = \{\theta_1, \ldots, \theta_t\}$, where t is the total number of trees. Then the posterior probability $P(C|D)$ for random forest could be written as [119]:

$$P(C|D,\,\Theta) = \sum_{\theta \in \Theta} P(C|D,\,\theta) \cdot P(\theta|T) = \frac{1}{|\Theta|} \sum_{\theta \in \Theta} \frac{N_{C,\theta}}{N_\theta} \tag{2.18}$$

where $P(\theta|T)$ is the probability of decision tree θ after observing training data T. Since no preference is given to any tree, each of them receives uniform posterior $1/|\Theta|$.

2.3.5 Training dataset reduction

Training data reduction is the technique that is used for large scale datasets to increase the learning speed of machine learning algorithms. Lee and Mangasarian [120] proposed a reduced support vector machine algorithm which uses a randomly selected subset of data that is typically 10% or less of the original dataset to obtain a nonlinear separating surface. Reduced SVM gets better test set results than those obtained by using the entire data. Lei and Govindaraju [121] introduced reduction of the feature space using principal component analysis and recursive feature elimination. These methods can speed up the evaluation of SVM by an order of 10 while maintaining comparable accuracy. Graf et al. [122] proposed cascade SVM where the

training set is first divided into a number of subsets and then these subsets are optimized by multiple SVMs. The partial results were combined and filtered again in the "cascade" of SVMs until the global optimum was reached. Later, Meyer et al. [123] introduced a new stepwise bagging approach that exploits parallelization in a better way than cascade SVM and contains an adaptive stopping-time to select the number of stages for improving accuracy. Nandan et al. [124] used a linear time algorithm based on convex hulls and extreme points to select subset, the so-called representative set of the training data for SVM optimization. Wang et al. [125] reduced SVM training time using only the most informative samples, obtained after removing most of the training data. Guo and Boukir [126] proposed a new ensemble margin-based data selection approach based on a simple and efficient heuristic to provide support vector candidates: they selected the lowest margin instances that reduced SVM training task complexity while maintaining the accuracy of the SVM classification. Mao et al. [127] trained a number of kernel SVMs on the randomly selected small subsets of training data and concluded that it is more efficient than training a single kernel SVM on the whole training data especially for large datasets. Mourad et al. [128] proposed a computationally efficient subset selection algorithm for fast SVM training on large scale data. Liu et al. [95] proposed training approximate SVM by using the anchors obtained from non-negative matrix factorization in a divide-and-conquer framework. Instance selection can also be achieved by using cluster sampling. The k-Means algorithm is well known for its efficiency in clustering large data sets [129]. Huang [129] presented two algorithms which extend k-Means to categorical domains and domains with mixed numeric and categorical values. They demonstrated the efficiency of their methods on two real world datasets. The authors [130] proposed a fast prototype selection method for large datasets, based on clustering. They reported that the method showed a competitive performance against a decremental reduction optimization procedure, generalized condensed nearest neighbor rule, pair opposite class-nearest neighbor and cluster method for template selection. Zheng et al. [131] proposed a hybrid k-Means and SVM for breast cancer diagnosis. The k-Means clustering algorithm provides a highly effective and compact feature set for SVM. The results demonstrated reduced computation time without losing diagnosis accuracy. Tang et al. [132] presented a method based on k-Means clustering and SVM for large scale data classification. k-Means was used to select representative instances. The results showed a better accuracy and a faster speed than traditional methods. The authors also stated that the random training sample selection method could achieve good accuracy occasionally in some data sets, but it is not very consistent [132]. This led to the conclusion to use k-Means clustering in this research.

k-Means (MacQueen et al. [133]) is one of the oldest and widely research clustering algorithms. It is often preferred due to its simplicity and generally very fast performance. The main idea is to partition the input dataset into k clusters, represented by adaptively-changing centroids (also called cluster centers); they are initialized using so-called seed-points. k-Means computes the squared distances between the input data points and centroids, and assigns inputs to the nearest centroid.

Let $X = \{x_i\}$, $i = 1,\ldots,n$ be the set of n d-dimensional points to be clustered into a set of K clusters, $C = \{c_k\}$, $k = 1,\ldots,K$. The k-Means algorithm finds such a partition that a squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let μ_k be the mean of cluster c_k. The squared error between μ_k and the points in cluster c_k is defined as [134]:

$$J(c_k) = \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \tag{2.19}$$

The goal of k-Means is to minimize the sum of the squared error over all K clusters [134],

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \tag{2.20}$$

2.3.6 Hyperparameter optimization

Hyperparameter optimization is the problem of choosing a set of optimal hyperparameters for a machine learning algorithm. The goal of many machine learning tasks can be summarized as training a model M which minimizes a loss function $L(T; M)$ on a training set T. Model M is constructed by a learning algorithm A using T, and typically involves solving an optimization problem. The model may be parametrized by the hyperparameters λ, and it is given as $M = A(T; \lambda)$ [135, 136]. The goal of hyperparameter optimization is to find a set of hyperparameters λ^* that yield an optimal model M^* which minimizes $L(V; M^*)$, where V is the validation set [135]:

$$\lambda^* = \arg\min_{\lambda} L(T; M) = \arg\min_{\lambda} f(\lambda; A, T, V, \lambda) \tag{2.21}$$

The objective function f takes the hyperparameters λ, and returns the associated loss value. The datasets T and V (where $T \bigcap V = \oslash$) are given, and the learning algorithm A and the loss function L are chosen [136].

All approaches to the automatic hyperparameter tuning are split into model-free and model-based methods [136].

Model-free methods. The most commonly used methods are grid search and random search. Grid search sets up a grid of hyperparameter values and trains a model and scores on the validation data for each combination. According to the authors [136], grid search downside is time complexity, whose costs grow exponentially at a rate of $O(n^k)$ – taking n distinct values for k parameters. Random search is an alternative to grid search, where search sets up a grid of hyperparameter values and selects random combinations to train the model and scores.

Model-based methods. In the case of model-based methods, the most commonly used are Bayesian optimization, gradient-based optimization and population-based methods. Bayesian optimization finds the value that minimizes an objective function by building a surrogate function (probability model) based on the past evaluation results of the objective. The surrogate is cheaper to optimize than the objective, so the next input values to evaluate are selected by applying a criterion to the surrogate. Gradient-based optimization computes the gradient with respect to hyperparameters for the machine learning algorithms and optimizes the hyperparameters using gradient descent. Population-based methods are optimization algorithms that maintain a population, i.e., a set of configurations, and improve this population by applying local perturbations (so-called mutations) and combinations of different members (so-called crossover) to obtain a new generation of better configurations [137].

Hyperparameter optimization is mostly guided by some heuristics, like genetic algorithm in [138] and [139], particle swarm optimization in [140] and [141], ant colony optimization in [142] and [143] etc. Simple grid search is one of the most common choices of solving this problem [144] as it is often integrated in different machine learning packages, such as LibSVM [145] or *scikit-learn* [11] which helps to simplify research pipelines. In grid search a list of candidate values for each hyperparameter is defined and evaluated, and it could become very time-consuming or even computationally infeasible for the optimization of many hyperparameters in large datasets [146]. The alternative to grid search is the random search method. Bergsta et al. [147] applied Bayesian optimization to tune the hyperparameters of a deep neural network and outperform random search. They stated that random search is unreliable for training deep neural network. Later Bergsta and Bengio [148] reported that random search found better models in most cases and required less computational time compared with neural networks configured by grid search. Compared with deep belief networks random search found statistically equal performance on four of seven data sets, and superior performance on one of seven data sets compared with grid search. Another comparison performed by Bergstra et al. [149] was performed between random search and the Bayesian based method and results showed the superiority of the latter. Mantovani et al. [146] proved random search effectiveness in lower

computational cost for SVM hyperparameters tuning to compare with grid search, genetic algorithm, particle swarm optimization and estimation of distribution algorithms metaheuristics. Random search obtained a solution as good as grid search. Statistical tests showed no significant difference between metaheuristics and the other techniques for the optimization of SVM hyperparameters. Pedregosa [150] proposed gradient-based optimization of simple models' hyperparameters. They reported that approach is highly competitive with respect to state-of-the-art methods like Bayesian optimization models, random search, grid search etc. The invention presented by Han et al. [96] relates to a product sale prediction method based on a support vector machine model with parameter optimization, which was performed by selecting a kernel function of support vector machine and adopting a grid search method to optimize predetermined parameters in the kernel function. According to the authors, "the invention overcomes the defects of poor precision and low calculation efficiency in traditional sale prediction, can provide a relatively accurate sale prediction reference for a decision-making level, and has good application value". Bakhteev and Strijov [151] performed comprehensive analyzes of gradient-based hyperparameter optimization algorithms. The results obtained were compared with random search. The authors concluded that the gradient-based algorithms are significantly more effective than random search-based algorithms when the number of hyperparameters is large. Snoek et al. [152] presented methods for performing Bayesian optimization for hyperparameter selection of general machine learning algorithms and reported that the proposed algorithms can reach or surpass human expert-level optimization for many algorithms. Han et al. [153] applied a genetic algorithm for the hyperparameter optimization of convolutional neural network and compared it with random search and grid search. They suggested that hyperparameters can be optimized better with a genetic algorithm. Bayesian optimization has received a lot of attention in recent years. Wu et al. [154] presented a hyperparameter tuning algorithm for machine learning models based on Bayesian optimization. They concluded that their method could achieve great accuracy in a few samples and could also find the best hyperparameters for the widely used machine learning models, such as the random forest algorithm and the neural networks, even multi-grained cascade forest under the consideration of time cost. Gustafsson et al. [155] proposed a new Bayesian optimization method for finding optimal hyperparameters in econometric models, which can be used to optimize any noisy function where the precision is under the control of the user. They reported that their method found the optimum much quicker than traditional Bayesian optimization or grid search.

Population-based methods are conceptually simple, they can handle different data types and are embarrassingly parallel [156] since a population of N members can be evaluated in parallel on N machines [137]. Particle

swarm optimization is one of the most known population-based methods. It is also a very promising option [157, 158, 159]. One of its strengths is combination with other evolutionary techniques. In [160] the authors proposed an improved-quantum behaved particle swarm algorithm based on a mutation operator. Zhang et al. [161] presented the SVM parameter optimization technique based on intercluster distance in the feature space and a hybrid of the barebones particle swarm optimization and differential evolution. In [83] a differential particle swarm optimization to select parameters for support vector machines is applied. There is a number of works which focus on the combined selection of both features and hyperparameters [93, 162]. Lu et al. [97] presented invention of optimizing two SVM parameters: C and g by using an improved particle swarm optimization algorithm. The authors reported that the presented invention "improves the classification accuracy of the SVM classification model, and promotes wider applications of the SVM classification model in the fields of model identification, system control, production scheduling, computer engineering, and electronic communications".

The reviews showed that the most popular method for hyperparameter optimization is Bayesian optimization, while it is mostly used when the number of hyperparameters is large. However, in this dissertation there is only one hyperparameter which will be optimized; based on the reviews it was decided to use grid search and PSO, which are still very promising [96, 97]. According to Feurer and Hutter [137], random search is a useful baseline because it makes no assumptions on the machine learning algorithm being optimized, and, given enough resources, will achieve performance arbitrarily close to the optimum. Therefore, the random search will be used as baseline for comparison, as well as Bayesian optimization.

Particle swarm optimization was introduced in [163]. Let $a_i(t)$ denote the position of particle i in the search space at time step t; unless otherwise stated, t denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position, i.e.

$$a_i(t+1) = a_i(t) + v_i(t+1) \tag{2.22}$$

with $a_i(0) \sim U(a_{min}, a_{max})$. Velocity vector reflects both the experiential knowledge of the particle and socially exchanged information from the particle's neighborhood.

For the global best PSO, the velocity of particle i is calculated as

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\,[b_{ij}(t) - a_{ij}(t)] + c_2 r_{2j}(t)\left[\hat{b_j}(t) - a_{ij}(t)\right] \tag{2.23}$$

where $v_{ij}(t)$ is the velocity of particle i in dimension $j = 1, \ldots, n_a$ at time step t, $a_{ij}(t)$ is the position of particle i in dimension j at time step t, c_1

and c_2 are positive acceleration constants used to scale the contribution of the cognitive and social components respectively, and $r_{1j}(t)$, $r_{2j}(t) \sim U(0, 1)$ are random values in the range [0, 1], sampled from a uniform distribution. These random values introduce a stochastic element to the algorithm.
The personal best position, b_i, associated with particle i is the best position the particle has visited since the first time step. Considering minimization problems, the personal best position at the next time step, $t+1$, is calculated as [164].

$$b_i(t+1) = \begin{cases} b_i(t) & \text{if } f(a_i(t+1)) \geq f(b_i(t)) \\ a_i(t+1) & \text{if } f(a_i(t+1)) < f(b_i(t)) \end{cases} \tag{2.24}$$

where $f : \mathbb{R}^{n_a} \to \mathbb{R}$ is the fitness function. As with evolutionary algorithms, the fitness function measures how close the corresponding solution is to the optimum, i.e. the fitness function quantifies the performance, or quality, of a particle (or solution) [164].
The global best position, $\hat{b}(t)$, at time step t, is defined as

$$\hat{b}(t) \in \{b_0(t), \ldots, b_{n_s}(t)\} | f(\hat{b}(t)) = \min\{f(b_0(t)), \ldots, f(b_{n_s}(t))\} \tag{2.25}$$

where n_s is the total number of particles in the swarm. $\hat{b}$ is the best position discovered by any of the particles so far – it is usually calculated as the best personal best position. The global best position can also be selected from the particles of the current swarm, in which case [164, 165]

$$\hat{b}(t) = \min\{f(a_0(t)), \ldots, f(a_{n_s}(t))\} \tag{2.26}$$

2.3.7 Ensemble methods

Ensembles of classifiers are one of the most challenging areas yet they often result in increased performance compared to single classifiers. Da Silva et al. [166] proposed an approach that automatically classifies the sentiment of tweets by using classifier ensembles and lexicons. Ensembles were formed by MNB, SVM, RF and LR. They concluded that the ensemble method can improve classification accuracy compared to standalone classifiers. In [167] the proposed ensemble method is based on static classifier selection involving a majority voting error and forward search for text sentiment classification. In [168] an ensemble system based on three classifiers – naïve Bayes, maximum entropy and knowledge-based tool, which are combined via a majority voting for the sentiment analysis of textual data is presented. The same authors [169] presented a classifier ensemble approach based on a knowledge-based tool and two machine learning classifiers (naïve Bayes and a maximum entropy) in order to detect emotional content in social media and to examine its performance under bagging and boosting combination methods. In [170]

the authors reported that their ensemble voting algorithm in conjunction with three classifiers performed better solving a Turkish sentiment classification problem. Sadhasivam and Kalivaradhan [171] proposed an ensemble method based on majority voting, which combine naïve Bayes and support vector machine for sentiment analysis of Amazon products. They reported better accuracy achieved by the ensemble method, compared with standalone NB and SVM. Tan et al. [172] used a combination of multinomial naïve Bayes and AdaBoost method for text emotion classification. MNB was used for the generation of several weak classifiers, while AdaBoost method was employed for obtaining a strong classifier by linear combination of several weak classifiers. They resulted in increased accuracy compared with ordinary MNB. Alrehili and Albalawi [173] proposed an ensemble method based on voting for sentiment analysis on Amazon customer reviews. The ensemble contains five machine learning methods: NB, SVM, RF, bagging and boosting. They demonstrated that ordinary RF outperformed the proposed ensemble in the case of using unigram, while with bigram and trigram it delivered the best performance.

2.3.8 Natural language processing

Natural language processing (NLP) defined by Liddy [174] is "a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications". According Sun et al. [175], opinion mining requires several preprocessing steps for structuring the text and extracting features, including tokenization, word segmentation, part of speech tagging and parsing.

2.3.8.1 Features extraction

Features extraction is one of the NLP techniques and very important part of sentiment analysis. Properly extracted features can increase the accuracy of machine learning algorithm [26, 27]. Tommasel and Godoy [176] reviewed features selection techniques for short texts. According to the summary of these techniques, the most commonly used data preprocessing are the following:

- tokenisation [177, 178, 179, 180, 181, 182];
- removal of stop words [180, 182, 183, 184, 185, 186];
- stemming [182, 184, 186];
- TF-IDF irrelevant feature removal [186];
- special symbol removal [180, 185];
- URLs removal [180, 181, 187, 188];
- removal of usernames [181, 187, 188];

- removal of tweets with less than 7 tokens [181, 189];
- removal of hashtags and non-alphabetic characters [187];
- nouns, verbs, and adjectives are kept [177, 178, 179];
- tweet segmentation [190];
- POS [191, 192, 193];
- n-grams [5, 6, 34, 194].

2.3.8.2 N-grams

N-grams is another feature extracting technique which is commonly used in NLP. The mostly used are unigrams, bigrams, trigrams and their combinations [5, 6, 194]. Agarwal and Mittal [195] gave a definition of unigrams and bigrams. According to them, unigram features are "simply bag-of-words features extracted by eliminating extra spaces and noisy characters between two words". Respectively bigrams are "the features, consisting of every two consecutive words in the text". Then we can define trigrams as the features, consisting of every three consecutive words in the text. We can see from the definitions that the prefix before "gram" points to a number of consecutive words in the text.

An example of n-grams:

full sentence:	*purchased this with a store credit*
unigrams:	*purchased* | *this* | *with* | *a* | *store* | *credit*
bigrams:	*purchased this* | *this with* | *with a* | *a store* | *store credit*
trigrams:	*purchased this with* | *this with a* | *with a store* | *a store credit*

2.3.8.3 Part of speech tagging

Part of speech tagging and parsing are techniques that analyze lexical and syntactic information. POS tagging is used to determine the corresponding POS tag for each word. The POS tags, such as adjective and nouns are quite helpful because opinion words are usually adjectives and opinion targets (i.e., entities and aspects) are nouns or a combination of nouns. While POS tagging provides lexical information, parsing obtains syntactic information. Parsing produces a tree which represents the grammatical structure of a given sentence with the corresponding relationship of different constituents [175]. Neethu and Rajasree [191] proposed a feature vector creation technique, which includes these eight features: part of speech tag, special keyword, presence of negation, emoticon, number of positive keywords, number of negative keywords, number of positive hash tags and number of negative hash tags. According to them, "keywords that are adjective, adverb or verb

show more emotion than others." Ortigosa et al. [196] combined a lexicon-based approach with machine learning algorithms and achieved better results. Sentiment extraction contains these steps: lowercase, idiom detection, segmentation into sentences, tokenization, emoticon detection, interjection detection, token score assignation, POS tagging and syntactical analysis, polarity calculation. For feature extraction Pak and Paroubek [197] used filtering, which includes the removal URL links, Twitter user names, Twitter special words, emoticons, as well as tokenization, stop words removal, n-grams constructing. Boiy and Moens [198] used unigrams, stems, negation, discourse features, depth difference, path distance and simple distance for feature selection. Abdi et al. [193] also used sentiment lexicon feature, negation features, sentence types, punctuation feature, POS feature, sentiment score feature. Yousefpour [192] used n-gram features, POS, negation, sentiment lexicon, syntactic and semantic dependency. Bharti and Singh [199] applied stop words removal, stemming, tokenization, term weighting, relevance score computation, term variance, document frequency and merge feature sublists.

2.3.8.4 Text preprocessing

Text preprocessing is a very important step in building a machine learning model. It transforms text into a form which is predictable and analyzable. The main steps of text preprocessing are as follows:

- converting to lowercase
- noise removal
- stop word removal
- TF-IDF

Let's define *df*[' text '] as dataframe *df* with column [' text '], which contains reviews about a product. Further text preprocessing will be presented by using this definition.

Converting to lowercase

Converting to lowercase is the easiest and a very effective form of text preprocessing, which can impact the classification results. It means that the words 'Word', 'WORD' and 'WorD' are treated in the same way. Lowercasing could be performed by using Python function *lower*(). i.e. of lowercasing:

```
#lowercasing by using Python
df['text'].apply(lambda x: x if type(x)!=str else x.lower())
```

Input: *Purchased this with a store credit and LIKE IT A LOT. I paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). RECOMMENDED!!!!! # @user: I put a photo on my website http://www.user.com*

Output: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # @user: i put a photo on my website http://www.user.com*

The frame "Input" contains an example of a raw Amazon product review (slightly modified for showing how test preprocessing works), while "Output" is the text after lowercasing.

Noise removal

Noise removal is another a very important step in NLP. It cleans up the text from noise, such as the removal of URLs, usernames, special characters, numbers, stop words etc. Noise removal could be performed by using Python function *str.replace*().

URLs removal:

```
#URL's removal by using Python
df['text'].str.replace('http\S+|www.\S+', ' ', case=False)
```

The frame "Input" contains the text after the previous step, when lowercasing is applied, while "Output" is the text after the removal of URLs.

Input: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # @user: i put a photo on my website http://www.user.com*

Output: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # @user: i put a photo on my website*

removal of usernames:

```
#removal of usernames by using Python
df['text'].str.replace('@[^\s]+', ' ', case=False)
```

The frame "Input" contains the text after the previous step, when URLs are removed, while "Output" is the text after the removal of usernames.

Input: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # @user: i put a photo on my website*

Output: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # i put a photo on my website*

removal of special characters:

```
#removal of special characters by using Python
df['text'].str.replace('[^\w\s]', ' ', case=False)
```

The frame "Input" contains the text after the previous step, when the usernames are removed, while "Output" is the text after the removal of special characters.

Input: *purchased this with a store credit and like it a lot. i paid 100$ and it compares well with more expensive ones, so don't pay more :)))))). recommended!!!!! # i put a photo on my website*

Output: *purchased this with a store credit and like it a lot i paid 100 and it compares well with more expensive ones so don t pay more recommended i put a photo on my website*

removal of numbers:

```
#removal of numbers by using Python
df['text'].str.replace('\d+', ' ', case=False)
```

The frame "Input" contains the text after the previous step, when the removal of special characters is applied, while "Output" is the text after the removal of numbers.

Input: *purchased this with a store credit and like it a lot i paid 100 and it compares well with more expensive ones so don t pay more recommended i put a photo on my website*

Output: *purchased this with a store credit and like it a lot i paid and it compares well with more expensive ones so don t pay more recommended i put a photo on my website*

It is very important to process noise removal in sequence as it is presented above because for example, if special character removal is performed before the removal of URLs or usernames, then URLs and usernames could not be recognized and removed.

Removal of stop words

Stop words can be imported and printed by using Python code:

```
from nltk.corpus import stopwords
nltk.download('stopwords') # update list of stopwords
print(stopwords.words('english'))
```

This code proceeded the output as follows:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',

'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',"wouldn't"]

Stop words removal is also one of the very common NLP tasks. Stop words are treated as very common words. Their removing could save computing time and efforts in processing large textual datasets. It is very important for sentiment analysis to carefully check the list of stop words, before removing them from textual datasets because it could impact the results. For example, the list of stop words presented above contains such words as 'not', 'don't', 'shouldn't' etc., which can change the polarity of sentiment if they are removed. A modified list of stop words is presented below:

[*'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'will', 'just', 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y'*]

In Python the removal of stop words could be performed during convertion into vector of numbers (see TF-IDF below). It will save the

execution time of text preprocessing in the case of executing these tasks separately.

The frame "Input" contains the text after the previous step – noise removal, while "Output" contains the text after the romoval of stop words.

Input: *purchased this with a store credit and like it a lot i paid and it compares well with more expensive ones so don t pay more recommended i put a photo on my website*

Output: *purchased store credit like lot paid compares well expensive ones don pay recommended put photo website*

TF-IDF

Example of results achieved by converting text from the previous step.

```
vectorizer = TfidfVectorizer(stop_words=set(stopwords))
text_vector = vectorizer.fit_transform(text)
```

Table 2.2 presents the results after the execution of TF-IDF. It is important to notice that stop word removal could affect the results because there is a possibility that text (review) contains only stop words; then after performing removal the result will be empty string.

Table 2.2. TF-IDF values

Word	TF-IDF	Word	TF-IDF	Word	TF-IDF	Word	TF-IDF
purchased	0.19052	lot	0.18542	expensive	0.25197	recommended	0.24000
store	0.23592	paid	0.25772	ones	0.23271	put	0.17966
credit	0.30158	compares	0.37898	don	0.32908	photo	0.29534
like	0.12004	well	0.14507	pay	0.23737	website	0.27897

The whole textual dataset could contain more than one empty string; then if this happens in training dataset it will contain less informative data. To avoid this, it was decided not use stop words removal in this dissertation.

TF-IDF description Before textual data is passed to a machine learning algorithm input it should be converted into vector of numbers because ML algorithms cannot work with text data directly. The most com-

mon solution for that is TF-IDF. It works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus.

Term Frequency (TF) is the simplest measure to weight each term in a text. In this method, each term is assumed to have importance proportional to the number of times it occurs in a text [200]. The weight of a term t in a text d is given by [201]

$$W(d, t) = \mathrm{TF}(d, t) \tag{2.27}$$

where $\mathrm{TF}(d, t)$ is the term frequency of the term t in the text d.

Inverse Document Frequency (IDF) concerns term occurrence across a collection of texts. The importance of each term is assumed to be inversely proportional to the number of texts that contain the term [202]. The IDF factor of a term t is given by [201]

$$\mathrm{IDF}(t) = \log(N/df(t)) \tag{2.28}$$

where N is the total number of texts in the collection and $df(t)$ is the number of texts that contain the term t.

Since TF is known to improve recall and IDF is expected to improve the precision, Salton [203] proposed to combine them to weight terms, and showed that they gave better performance. The combination weight of a term t in text d is given by [201]

$$W(d, t) = \mathrm{TF}(d, t) \cdot \mathrm{IDF}(t) \tag{2.29}$$

2.4 Conclusions of Chapter 2

In this chapter related works in the field of textual data sentiment analysis using machine learning algorithms are presented. Problems related to concrete ML algorithms, which other authors are trying to solve are reviewed, as well as training data reduction and hyperparameter tuning methods; ensemble methods and natural language processing are also presented.

1. The literature analysis showed that naïve Bayes classification, support vector machine, logistic regression, random forest and decision

tree are still very attractive to researchers in textual data sentiment analysis. Consequently, they were selected for the experimental part of this dissertation. It was decided to use classical versions of these ML algorithms on purpose as it was easier to implement a parameter tuning and altering the model designs, because they are quite easy to interpret and understand. In the case of state-of-the-art methods, very often they are "black box" and even researchers do not fully understand their "inside" due to the lack of theoretical foundation. The review of patents showed that textual data sentiment analysis is still attractive to inventors, and there are a number of inventions whose goal is to increase classification accuracy and mostly classical methods are used as the starting point.

2. The review of aforementioned five machine learning algorithms showed that increasing accuracy and reducing classification time are the most common problems, which many authors are trying to solve.
3. The review of hyperparameter optimization led to conclude that Bayesian optimization, grid search, particle swarm optimization and random search are very promising options, while the Bayesian optimization and random search are very competitive compared to PSO and grid search. Grid search and PSO methods were selected for hyperparameters tuning taking into account these reviews. Random search and Bayesian optimization will be used as an option for comparison.
4. The review of ensemble methods shows that majority voting is the most common technique for combining classifiers, which can improve classification accuracy. Therefore, the technique was also selected for this research.
5. Text preprocessing could increase the classification accuracy of a machine learning algorithm, while it could become very time-consuming in large datasets. While the goal of this dissertation is to propose a hybrid textual data classification method for large scale datasets, it was decided to use simple text preprocessing: lowercasing, noise removal and TF-IDF for converting text into vector of numbers. The removal of stop words could affect the results because there is a possibility that text (review) consists of a lot of stop words or of only stop words; then after performing removal, the result will contain less informative data or the meaning of a review will be changed. To avoid this, it was decided not use the removal of stop words in this dissertation.

3. METHODOLOGY OF THE RESEARCH

In this part of the dissertation a hybrid method for textual data sentiment analysis is presented. The aim of the method is to increase the classification speed of classical methods with a similar classification accuracy compared with classical methods. The diagrams and algorithms of the method are briefly described in this chapter. Parts of this chapter are published in [A1],[A2],[A3],[A4],[A5],[A7].

Further a description of the datasets which are used for experimental research and a performance evaluation metrics for the comparison of the results are presented.

3.1 Proposed hybrid method

The proposed hybrid method is a combination of four methods: classical machine learning algorithm, k-Means clustering, particle swarm optimization metaheuristic and ensemble, which are integrated into the SpeedUP method. The expression (Expr) of the SpeedUP method (the default parameters are set considering the experimental results) is as follows:

$$\textbf{SpeedUP}(ml,\ kmeans,\ ensemble,\ pso,\ Subset_{size},\ D_{text},\ num_{class}) \tag{3.1}$$

Parameters:

ml – string, 'LSVM', 'MNB', 'RF', 'DT' or 'LR' (default = 'LSVM'); *ml* indicates the classical machine learning algorithm which is used.

kmeans – integer, 0 or 1 (default = 1); it is a training dataset selection method. When *kmeans* = 1, the k-Means clustering method is used for training dataset selection, when *kmeans* = 0 – training dataset is selected by using random sampling.

ensemble – integer (default = 0); value *ensemble* indicates the number of voters in the method. If *ensemble* = 0 or 1 (recommended when *pso* = 1), then the ensemble method is not used.

pso – integer, 0 or 1 (default = 1); *pso* = 1 indicates that particle swarm optimization metaheuristic is used for parameter tuning (only for LR or LSVM). For MNB, RF and DT – *pso* = 0.

$\textbf{Subset}_{\textbf{size}}$ – integer (default = 30K); this value indicates the size of the subsets into which the testing dataset is divided. When $Subset_{size}$ =

30K all testing dataset is divided into equal subsets, which contains 30K records.

$\mathbf{D_{text}}$ – textual dataset.

$\mathbf{num_{class}}$ – integer (default = 2); this value indicates the number of different classes in dataset.

The diagram of the proposed hybrid method is presented in Figure 3.1. The regions bounded by the red rectangles present whose parts of the hybrid method, which are fully described later in this chapter.

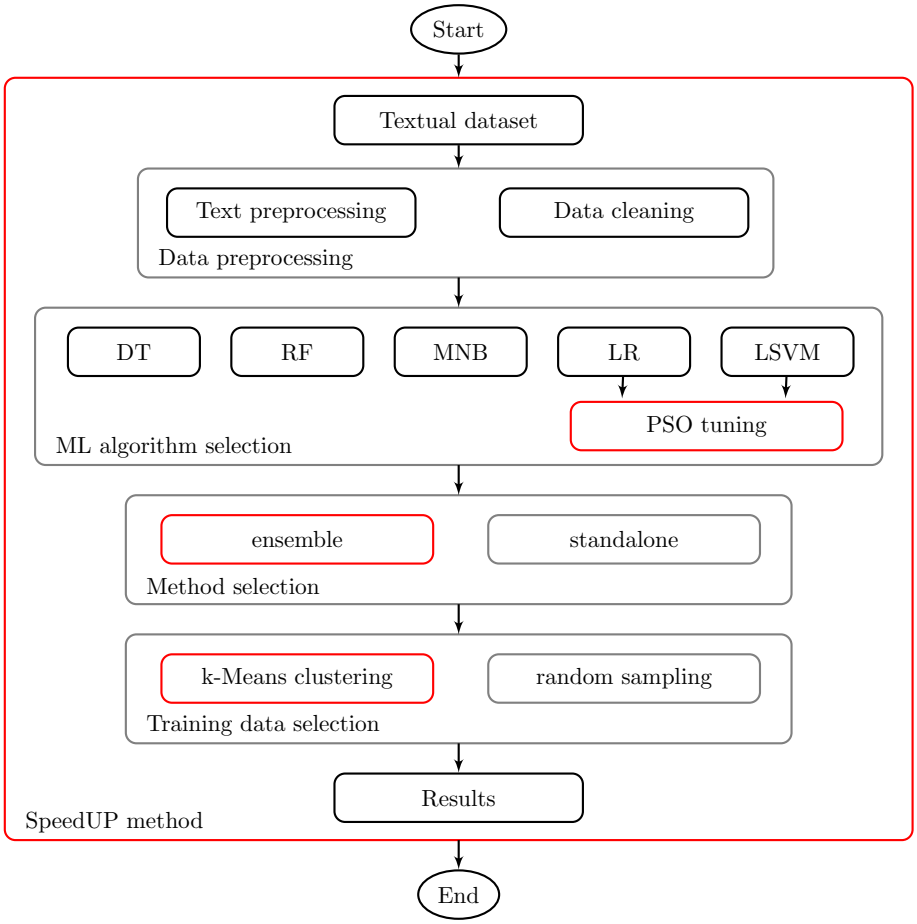


Fig. 3.1. Proposed hybrid method

The diagram consists of the following steps:

- **SpeedUP method**. This is the main part of the hybrid method; other parts are integrated into it. The aim of this method is to increase the

classification speed of the classical machine learning algorithms. Its more detailed description is presented below (See Subsec. 3.1.1).

- **Textual dataset**. Textual dataset, which will be used for sentiment analysis, is read from the data source and data preprocessing is performed on it. Datasets are described in Section 3.2.
- **Data preprocessing**. This step contains two actions: text preprocessing and data cleaning. Text preprocessing includes actions like converting to lowercase, removing redundant tokens such as hashtag, symbols @, numbers, "http" for links, punctuation symbols, usernames etc. Later, in the data cleaning part, the dataset is checked for empty strings and if any exist they are removed. The aim of this step is to prepare dataset for the next stage of SpeedUP, where it will be prepared for processing on the machine learning algorithm.
- **ML algorithm selection**. This step contains the selection of machine learning algorithm, which will be used with the SpeedUP method. The selection is performed by changing *ml* value in SpeedUP (See Expr. 3.1). This step also included the PSO tuning method, which is proposed for logistic regression and linear support vector machine to increase classification accuracy. The selection of this method is performed by setting *pso* value to 1 in SpeedUP (See Expr. 3.1). A detailed description of this method is given below (See Subsec. 3.1.3).
- **Method selection**. The aim of this step is to select whether the standalone machine learning algorithm or ensemble, which is described below (See Subsec. 3.1.4) will be used. This method is used when value *ensemble* is set bigger than 1 in SpeedUP (See Expr. 3.1). This value defines how many classifiers there will be in the ensemble. It is important to stress that the ensemble contains only one kind of the machine learning algorithm, which is set in value *ml*, i.e. if *ml* = 'LSVM', then the ensemble will contain only LSVM algorithms.
- **Training data selection method**. In this step the method for training data selection is chosen. There is a possibility for the random selection of training data or by applying the part of proposed method k-Means clustering. This can be done by turning on *kmeans* parameter in SpeedUP (See Expr. 3.1). A detailed description of this method is given below (See Subsec. 3.1.2).
- **Results**. The results of the textual dataset sentiment analysis are expressed in statistical measures: accuracy, precision, recall, harmonic

mean and area under the receiver operating characteristics (See Section 3.3).

3.1.1 SpeedUP method

The SpeedUP method (introduced in [A4]) is the main part of the proposed hybrid method, whose configuration is presented when other parts are excluded. In other words, the values *kmeans*, *ensemble* and *pso* are set to 0 (See Expr. 3.1). The main idea is to reduce the size of the training dataset depending on subset size. Thus, the testing dataset is split into equal subsets and the size of training data is calculated on the basis of the first subset size. The subset size is defined with value $Subset_{size}$. After the machine learning algorithm is trained on reduced training dataset, subsets of testing dataset are passed to it one by one and later the results of each subset are combined into one result set. This is done anticipating that a smaller training dataset will reduce time and computational effort to train classifier, and it will provide similar accuracy compared to the classical machine learning algorithm. The diagram of the SpeedUP method is presented in Figure 3.2. The dashed line in the diagram represents steps for additional calculations executed before a concrete step is joined to it.

The diagram consists of the following steps:

- Textual dataset is split into a training and testing datasets. It is assumed that the testing dataset is 30% of the full textual dataset, therefore, the training dataset should be 70%.
- Training dataset is divided into sets of certain categories of sentiments (D_{class_1}, D_{class_2} etc.). Usually two categories are used : "positive" and "negative".
- Depending on values $Subset_{size}$ and num_{class} (See Expr. 3.1), the size of reduced training dataset is calculated and text instances of certain category ($Train_{count}$. This value is inside $f(x)$) in it are counted. After that selected instances of each category are combined into a reduced dataset – "Training data". By default, selection is performed with random sampling.
- "Training data" is passed to the ML algorithm for learning.
- Depending on value $Subset_{size}$ testing dataset is divided into subsets td_1, td_2 etc. (function $y(x)$).
- Subsets are passed to the ML algorithm one by one and the results achieved are stored in separate datasets r_1 – for td_1, r_2 – for td_2 etc.

- Finally, the results are combined into one result set – Results.

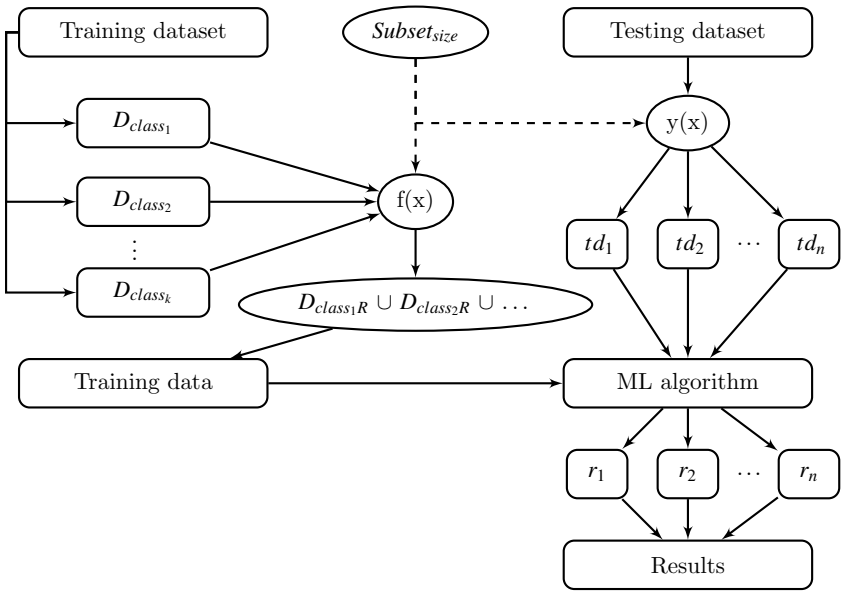


Fig. 3.2. Diagram of the SpeedUP method

Algorithm 1 presents the SpeedUP method.

List of parameters used in the algorithm:

D_{text} – textual dataset

ml – machine learning algorithm

num_{class} – number of different classes in a dataset

$class_i$ – certain category of sentiment assigned to the text in a dataset

td_i – testing data subset

$Subset_{size}$ – size of the subsets into which the testing dataset is divided

D_{train} – training dataset

D_{class_i} – set of text instances of certain class ($class_i$) from D_{train}

D_{class_iR} – selected data from D_{class_i} depending on $Train_{count}$

D_{test} – testing dataset

$Train_{count}$ – count of text instances of certain class should be selected from the dataset. This value is calculated by the proposed formula:

$$Train_{count} = \frac{1}{k} * Subset_{size} * \frac{|D_{train}|}{|D_{test}|}$$

where k is the number of different classes, $|D_{train}|$ and $|D_{test}|$ are the number of instances in D_{train} and D_{test}, respectively

$\mathbf{MLres_i}$ – classification results of td_i

$\mathbf{R_{ML}}$ – classification results of all td_i

Algorithm 1 SpeedUP

Require: ml, D_{text}, num_{class}, $Subset_{size}$

1. D_{text} is split into a training (D_{train}) and testing datasets (D_{test}).
2. Randomly select data of each presented class in D_{train}.

$k \leftarrow num_{class}$

$Train_{count} \leftarrow 1/k * Subset_{size} * (|D_{train}|/|D_{test}|)$

$D_{class_1R} \leftarrow (random.sample(D_{class_1}, Train_{count}))$

$D_{class_2R} \leftarrow (random.sample(D_{class_2}, Train_{count}))$

$D_{class_kR} \leftarrow (random.sample(D_{class_k}, Train_{count}))$

3. Create a reduced training dataset – Training data (D_{train_R})

$D_{train\_R} \leftarrow D_{class_1R} \cup D_{class_2R} \cup \cdots \cup D_{class_kR}$

4. Train ml with D_{train_R}
5. Split the testing dataset in subsets (td_i) and run on ml.

$n \leftarrow 0$

for $i = 1 : trunc(len(D_{test})/Subset_{size})$ **do**

$td_i \leftarrow D_{test}[(n+1) : (Subset_{size} * i),]$

$MLres_i \leftarrow ml.predict(td_i)$

$R_{ML} \leftarrow R_{ML} \cup MLres_i$

$n \leftarrow (Subset_{size} * i)$

end for

if $(len(D_{test} \ \% \ Subset_{size}) > 0$ **then**

$td_{i+1} \leftarrow D_{test}[(n+1) : (len(D_{test})),]$

$MLres_{i+1} \leftarrow ml.predict(td_{i+1})$

$R_{ML} \leftarrow R_{ML} \cup MLres_{i+1}$

end if

Output : $Results \leftarrow R_{ML}$

3.1.2 k-Means clustering

The k-Means clustering method (introduced in [A2]) is another part of the proposed hybrid method, which can be used if value *kmeans* is set to 1 in SpeedUP (See Expr. 3.1). The main goal of this method is to reduce the training dataset for the ML algorithm instead of using a random training data selection. The need of this method appeared because when random

training data selection is used there is a risk that training data will contain the same data or the data will be not useful i.e. contain only one word, stop words etc. and this could negatively affect accuracy in different runs; therefore, multiple runs are required for more objective results, which is affecting classification speed. The diagram of the k-Means clustering method is presented in Figure 3.3.

The diagram consists of the following steps:

- The diagram starts with "Training dataset", which is a full dataset after "Data preprocessing" step (See Fig. 3.1).
- The k-Means clustering method depends on value *ensemble* in SpeedUP (See Expr. 3.1), if it is disabled (set to 0 or 1), then only one reduced training dataset "Training data" will be selected; otherwise as many datasets as are set in value *ensemble* will be chosen.
- Finally, the method selects the number of reduced training datasets, which will be used as input in the ML algorithm.

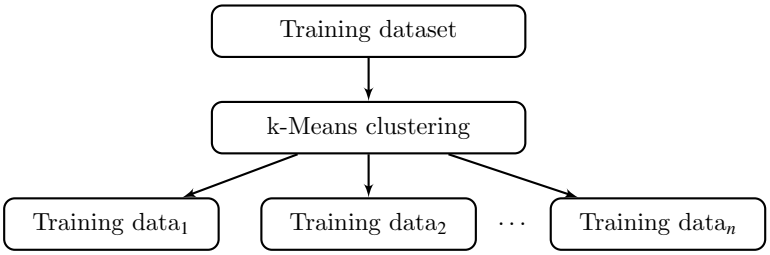


Fig. 3.3. Diagram of the k-Means clustering method

Algorithm 2 presents the k-Means clustering method.

List of parameters used in the algorithm:

$\mathbf{D_{class_i}}$ – set of text instances of certain class from training dataset
$\mathbf{D_{class_iR}}$ – selected data from D_{class_i} depending on $Train_{count}$
$\mathbf{k_{opt}}$ – optimal number of clusters
$\mathbf{Train_{count}}$ – count of text instances of certain class should be selected from the training dataset
txt1 – text from all clusters with *MAX* distance to the cluster center
txt2 – text from all clusters closest to *MEAN* distance to the cluster center
txt3 – text from all clusters with *MIN* distance to the cluster center
$\mathbf{kMeans_{res}}$ – results after the k-Means algorithm is executed

Algorithm 2 k-Means clustering method

Require: D_{class_i}, $Subset_{size}$, $Train_{count}$, k_{opt}

$D_{class_iR} \leftarrow \{\}$

$clf \leftarrow kmeans(k_{opt})$

while $(len(D_{class_iR}) <= Train_{count})$ **do**

$rD \leftarrow random.sample(D_{class_i}, Subset_{size})$

$clf.predict(rD)$

$distance \leftarrow pairwise\_distances(clf.cluster\_centers\_, rD)$

$$kMeans_{res} = \begin{cases} \text{txt1} \leftarrow \max(distance) \\ \text{txt2} \leftarrow \min(distance, key = lambda\ x : abs(x - median(distance))) \\ \text{txt3} \leftarrow pairwise\_distances\_argmin\_min(clf.cluster\_centers\_, rD) \end{cases}$$

$D_{class_iR} \leftarrow D_{class_iR} \cup kMeans_{res}$

end while

Output : D_{class_iR}

It is important to stress that the k-Means clustering method is performed separately to each certain category of sentiment assigned to the text. Finally, the results are combined into one reduced training dataset. It is done with intend to have the equal number of instances of each class in reduced training dataset.

Algorithm 3 presents a modified SpeedUP (See Algorithm 1) with integrated k-Means clustering to it:

Algorithm 3 SpeedUP with k-Means clustering

Require: ml, D_{text}, num_{class}, $Subset_{size}$

1. D_{text} is split into a training (D_{train}) and testing datasets (D_{test}).
2. Select data of each presented class in D_{train} by applying k-Means clustering.

$k \leftarrow num_{class}$

$Train_{count} \leftarrow 1/k * Subset_{size} * (|D_{train}|/|D_{test}|)$

$D_{class_1R} \leftarrow$ K-MEANS_CLUSTERING$(D_{class_1}, Subset_{size}, Train_{count}, k_{opt})$

$D_{class_2R} \leftarrow$ K-MEANS_CLUSTERING$(D_{class_2}, Subset_{size}, Train_{count}, k_{opt})$

$D_{class_kR} \leftarrow$ K-MEANS_CLUSTERING$(D_{class_k}, Subset_{size}, Train_{count}, k_{opt})$

3. Create a reduced training dataset – Training data (D_{train_R})

$D_{train\_R} \leftarrow D_{class_1R} \cup D_{class_2R} \cup \cdots \cup D_{class_kR}$

4. Train ml with D_{train_R}

▷ Continued on the next page

Algorithm 3 SpeedUP with k-Means clustering – continued from the previous page.

5. Split the testing dataset in subsets (td_i) and run on ml.
$n \leftarrow 0$
for $i = 1 : trunc(len(D_{test})/Subset_{size})$ **do**
  $td_i \leftarrow D_{test}[(n+1) : (Subset_{size} * i),]$
  $MLres_i \leftarrow ml.predict(td_i)$
  $R_{ML} \leftarrow R_{ML} \cup MLres_i$
  $n \leftarrow (Subset_{size} * i)$
end for
if $(len(D_{test} \ \% \ Subset_{size}) > 0$ **then**
  $td_{i+1} \leftarrow D_{test}[(n+1) : (len(D_{test})),]$
  $MLres_{i+1} \leftarrow ml.predict(td_{i+1})$
  $R_{ML} \leftarrow R_{ML} \cup MLres_{i+1}$
end if
Output : $Results \leftarrow R_{ML}$

The parameters in Algorithm 3 are the same as in Algorithm 1 and Algorithm 2, so they are not described here.

3.1.3 PSO tuning method

The PSO tuning method (introduced in [A3]) is also a part of the proposed hybrid method, which can be used if value *pso* is set to 1 in SpeedUP (See Expr. 3.1). This method is suitable only for the LSVM and LR algorithms. The main goal of this method is to select penalty (cost) parameter of the error term C for the aforementioned algorithms in order to increase the accuracy of the hybrid method. The diagram of the PSO tuning method is presented in Figure 3.4. The dashed line in the diagram represents steps for additional calculations executed before a concrete step is joined to it.

The diagram consists of the following steps:

- The diagram starts with ML algorithm selection. This selection is performed in SpeedUP (See Expr. 3.1) by setting parameter *ml* to 'LSVM' or 'LR'.
- Reduced training dataset (Training data), obtained by the k-Means clustering method (See Subsec. 3.1.2) or random sampling, is divided into training and validation data for tuning. These datasets also will be used in the C_Tuning method (introduced in [A2]) and

PSO search. It is important to stress that if parameter *ensemble* is enabled in SpeedUP, then this and the following steps of the diagram are performed for other reduced training datasets as well.

- The starting C (penalty parameter) value is defined by using cross-validated grid search (GridSearchCV) over a predefined grid of possible C values.
- The range is reduced with the C_Tuning method, which is a part of the hybrid method and is presented below in Algorithm 4.
- Finally, the particle swarm optimization method, presented below in Algorithm 5, searches for the best C value in the previously obtained range and then passing it to ML algorithm.

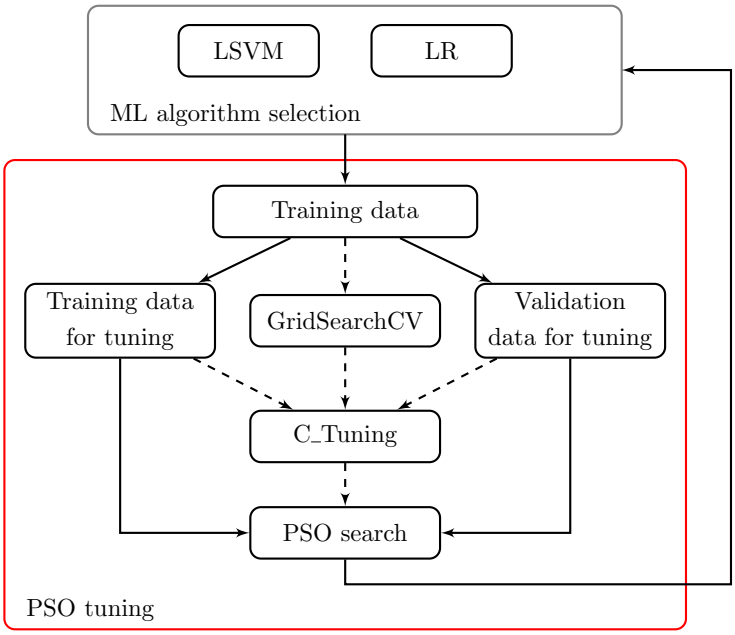


Fig. 3.4. Diagram of the PSO tuning method

GridSearchCV is a standard method, which was used with no modifications. During experiments it was obtained that it is not very efficient in time metrics and there is the need of a lot of iterations with reducing ranges to find the best C value for LR and LSVM algorithms. This led to the development of faster methods for that; GridsearchCV is used only for searching the start C value. Algorithm 4 presents the C_Tuning method whose aim is to search the start range for PSO.

List of parameters used in the algorithm:

ml – machine learning algorithm ('LSVM', 'LR')
$\mathbf{D_{trn}}$ – training data for tuning
$\mathbf{D_{val}}$ – validation data for tuning
$\mathbf{C_{Gsearch}}$ – C value obtained using cross-validated grid search

Algorithm 4 C_Tuning

Require: ml, D_{trn}, D_{val}, $C_{Gsearch}$
$results \leftarrow pandas.DataFrame$(columns=['acc', 'c'])
$C_{min} \leftarrow C_{Gsearch} - 1$
$C_{max} \leftarrow C_{Gsearch} + 1$
$i \leftarrow C_{min}$
while $(i <= C_{max})$ **do**
  $clf \leftarrow ml(C = i)$
  Train clf with D_{trn}
  $ACC \leftarrow clf.predict(D_{val})$
  $results \leftarrow results.append$('acc': ACC, 'c' : i, $ignore\_index = True$)
  $i \leftarrow i + 0.1$
end while
C, $ACC \leftarrow results.loc[results['acc'].astype(float).idxmax()]$
Output : C – penalty (cost) parameter of the error term, ACC – classification accuracy achieved by ml with C

Algorithm 5 presents the PSO tuning method, which includes GridSearchCV and C_Tuning methods.

List of parameters used in the algorithm:

ml – machine learning algorithm ('LSVM', 'LR')
$\mathbf{D_{trn}}$ – training data for tuning
$\mathbf{D_{val}}$ – validation data for tuning
$\mathbf{ACC_{Gbest}}$ – the best global accuracy
$\mathbf{C_{Gbest}}$ – the best global C value
param_grid – grid of parameters to sequences of allowed values
R – the radius; (neighborhood) of C_{Gbest} where PSO search is performed. R = 0.1, which was determined during the set of experiments
Ps – particle swarm dataframe
C – ml ('LR' or 'LSVM') penalty (cost) parameter
acc – accuracy obtained after execution ml

v – velocity vector of the direction and magnitude of the particle movement

best_acc – the best personal accuracy of the particle

best_C – the best personal C value of the particle (equal to position)

nPs – the number of particles in the swarm; nPs = 50, which was determined during the set of experiments

nIteration – number of iterations; nIteration = 100, which was determined during the set of experiments

$\mathbf{r_1}, \mathbf{r_2}$ – random vectors

c1, **c2** – acceleration coefficients; default value 2 for both coefficients

w – coefficient of inertia; default value 1

Algorithm 5 PSO tuning

Require: ml, D_{trn}, D_{val}

1. Find initial C value using standard cross-validated grid search

ACC_{Gbest}, $C_{Gbest} \leftarrow$ GridSearchCV(ml, D_{trn}, $param\_grid = (1, 2, \ldots, m)$)

2. Run C_Tuning.

ACC_{Gbest}, $C_{Gbest} \leftarrow$ C_Tuning(ml, D_{trn}, D_{val}, C_{Gbest})

3. Declare variables.

$C_{min} \leftarrow C_{Gbest} - R$

$C_{max} \leftarrow C_{Gbest} + R$

$Ps \leftarrow pandas.DataFrame(columns = [\text{'}C\text{'}, \text{'}acc\text{'}, \text{'}v\text{'}, \text{'}best\_acc\text{'}, \text{'}best\_C\text{'}])$

4. Initialize particles with population size nPs.

for $k = 1 : nPs$ **do**

  $Ps[k].C \leftarrow random.uniform(C_{min}, C_{max})$

  $Ps[k].v \leftarrow 0$

  $clf \leftarrow ml(C = Ps[k].C)$

  Train clf with D_{trn}

  $Ps[k].acc \leftarrow clf.predict(D_{val})$

  $Ps[k].best\_acc \leftarrow Ps[k].acc$

  $Ps[k].best\_C \leftarrow Ps[k].C$

  if $Ps[k].best\_acc > ACC_{Gbest}$ **then**

    $ACC_{Gbest} \leftarrow Ps[k].best\_acc$

    $C_{Gbest} \leftarrow Ps[k].best\_C$

  end if

end for

▷ Continued on the next page

Algorithm 5 PSO tuning – continued from the previous page.

5. Run PSO algorithm.

for $j = 1 : nIteration$ **do**
  for $k = 1 : nPs$ **do**
    $r_1 \leftarrow random.uniform(0,1)$
    $r_2 \leftarrow random.uniform(0,1)$
    $Ps[k].v \leftarrow w \times Ps[k].v + c1 \times r_1 \times (Ps[k].best\_C - Ps[k].C) +$
      $+ c2 \times r_2 \times (C_{Gbest} - Ps[k].C)$
    $Ps[k].C \leftarrow Ps[k].C + Ps[k].v$
    if $Ps[k].C <= 0$ **then**
      break
    end if
    $clf \leftarrow ml(C = Ps[k].C)$
    Train clf with D_{trn}
    $Ps[k].acc \leftarrow clf.predict(D_{val})$
    if $Ps[k].acc > Ps[k].best\_acc$ **then**
      $Ps[k].best\_acc \leftarrow Ps[k].acc$
      $Ps[k].best\_C \leftarrow Ps[k].C$
    end if
    if $Ps[k].best\_acc > ACC_{Gbest}$ **then**
      $ACC_{Gbest} \leftarrow Ps[k].best\_acc$
      $C_{Gbest} \leftarrow Ps[k].best\_C$
    end if
  end for
  $w \leftarrow w \times 0.99$
end for
Output : C_{Gbest}

3.1.4 Ensemble method

The ensemble method (first publication in [A1], [A5] and later in [A3]) is another part of the proposed hybrid method, which can be used if value *ensemble* is bigger than 1 in SpeedUP (See Expr. 3.1). This approach is majority voting based ensemble, which is presented to improve classification accuracy for machine learning algorithms as well. The diagram of the ensemble method is presented in Figure 3.5.

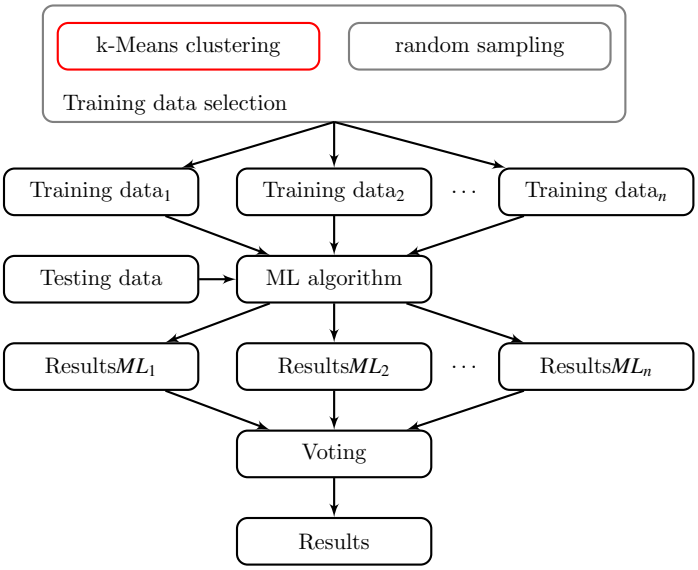


Fig. 3.5. Diagram of the ensemble method

The diagram consists of the following steps:

- The diagram starts with training data selection – selection of reduced training dataset. The method for it is set in SpeedUP by setting value *kmeans* = 1 for k-Means clustering and *kmeans* = 0 – for random sampling. The value *ensemble* should also be set, which determines the number of classifiers for voting (See Expr. 3.1).
- Depending on the selected value *ensemble* the same number of different reduced training datasets will be selected.
- After that all datasets are passed one by one to ML algorithm for training and evaluating on the testing data. Thus different results sets are obtained: ResultsML_1, ResultsML_2, . . . , ResultsML_n.
- Finally, voting is performed with the obtained result sets and the final step contains the results of the majority voting.

Algorithm 6 presents the ensemble method.

List of parameters used in the algorithm:

ml – machine learning algorithm, which specified in SpeedUP (is used one kind of ML algorithm i.e. 'LSVM')
D$_{trn}$ – set of reduced training dataset
D$_{test}$ – testing dataset
results – set of results of each classifier

ensemble – number of voters in ensemble

$\mathbf{td_k}$ – testing data subset

$\mathbf{Subset_{size}}$ – size of the subsets into which the testing dataset is divided

Algorithm 6 Ensemble method

Require: ml, $D_{trn} = \{D_1, D_2, \ldots, D_{ensemble}\}$, D_{test}, $ensemble$, $Subset_{size}$

$clf \leftarrow ml$

$results \leftarrow \{\}$

for $i = 1 : ensemble$ **do**

1. Train ml with D_i

$clf.fit(D_i)$

2. Split the testing dataset in subsets (td_k) and run on ml.

$n \leftarrow 0$

for $k = 1 : trunc(len(D_{test})/Subset_{size})$ **do**

$td_k \leftarrow D_{test}[(n+1) : (Subset_{size} * k),]$

$results[i] \leftarrow results[i] \cup clf.predict(td_k)$

$n \leftarrow (Subset_{size} * k)$

end for

if $(len(D_{test} \ \% \ Subset_{size}) > 0$ **then**

$td_{k+1} \leftarrow D_{test}[(n+1) : (len(D_{test})),]$

$results[i] \leftarrow results[i] \cup clf.predict(td_{k+1})$

end if

end for

Output : $\dfrac{1}{ensemble}\left(\sum_{i=1}^{ensemble} results[i]\right)$

3.2 Datasets

For experiments and the comparison of results the proposed method is evaluated on the largest labeled datasets available:

- Stanford Twitter sentiment corpus dataset[4] (sentiment140), is introduced by Go et al. in [20]. The data is a CSV with emoticons removed. Tweets are in English. Data file format has six fields:

  0 – the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)

  1 – the id of the tweet

[4] `http://help.sentiment140.com/`

2 – the date of the tweet
3 – the query
4 – the user that tweeted
5 – the text of the tweet

In the experiments only fields 0 (the polarity of the tweet) and 5 (the text of the tweet) were used. The dataset was reconstructed into two classes – "positive" and "negative". The classes were converted into binary as follows: tweets with polarity 0 or 2 were labeled as "negative", whereas tweets with polarity 4 received a label "positive".

- Amazon customer reviews dataset[5] (AmazonTest). It is constructed from the dataset presented by Zhang et al. [204], which originally is created from Amazon product data dataset[6] (AmazonProduct), by randomly taking training samples and testing samples for each review score from 1 to 5. At the end dataset was reconstructed into two classes. Most of the reviews are in English, but there are a few in other languages, like Spanish.
- Amazon product data dataset[6] introduced by McAuley et al. in [205]. In particular, these datasets from the Amazon product data were used: Books (Books), Electronics (Electronics), Kindle Store (KindleStore), Cell Phones and Accessories (Phones&Accessories). AmazonProduct dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). The format is one-review-per-line in (loose) json. The review consists of these fields:

  1. **reviewerID** - ID of the reviewer
  2. **asin** - ID of the product
  3. **reviewerName** - name of the reviewer
  4. **helpful** - helpfulness rating of the review
  5. **reviewText** - text of the review
  6. **overall** - rating of the product
  7. **summary** - summary of the review
  8. **unixReviewTime** - time of the review (unix time)
  9. **reviewTime** - time of the review (raw)

  In the experiments only fields reviewText (text of the review) and

[5] `https://www.kaggle.com/bittlingmayer/amazonreviews/`
[6] `http://jmcauley.ucsd.edu/data/amazon/`

overall (rating of the product) were used.

A brief description of datasets is presented in Table 3.1.

Table 3.1. Description of datasets

Dataset	Num. of reviews	Num. of classes
sentiment140	1,600,000	3
AmazonTest	4,000,000	2
Books	22,507,155	5
Electronics	7,824,482	5
KindleStore	3,205,467	5
Phones&Accessories	3,447,249	5

In the case of two classes we have "positive" and "negative" sentiment. Table 3.2 shows the meaning of the review of the five classes. Rating value 3 "is acceptable" means neutral sentiment. Inferior two levels (rating 1 and 2) are negative sentiments and superior two (rating 4 and 5) are positive sentiments.

Table 3.2. Meaning of the review

Rating	Meaning	Sentiment
1	avoid	negative
2	bad	negative
3	acceptable	neutral
4	good	positive
5	exceptional	positive

3.3 Performance evaluation

3.3.1 Effectiveness

Effectiveness is measured using statistical measures which are often used for similar tasks, particularly, accuracy, precision, recall, $F_1 score$ and AUC. Formulas are presented below (Sammut and Webb, [206]):

Accuracy: $ACC = \frac{TP+TN}{TP+TN+FP+FN}$

Precision. Positive predictive value: $PPV = \frac{TP}{TP+FP}$

Precision. Negative predictive value: $NPV = \frac{TN}{TN+FN}$

Recall. True positive rate: $TPR = \dfrac{TP}{TP+FN}$

Recall. True negative rate: $TNR = \dfrac{TN}{TN+FP}$

Harmonic mean of PPV and TPR: $F_1 score = \dfrac{2}{\frac{1}{PPV} + \frac{1}{TPR}}$

where TP is count of correctly classified "positive" sentiments, TN – count of correctly classified "negative" sentiments. FP is count of incorrectly classified "positive" sentiments and FN – count of incorrectly classified "negative" sentiments.

$time$ – is execution time of the machine learning algorithm. $time = stop - start$, where $start$ – the start of ML algorithm learning, $stop$ – the output of results.

3.3.2 Ranking

The evaluation of machine learning algorithms is performed with average ranks ranking method presented by the authors in [207], who were inspired by Friedman's statistic [208]. It is modified and adopted for ranking the proposed hybrid method applied on classical machine learning algorithms. The idea of this method is to order the algorithms according to the measured effectiveness metrics (See Subsec 3.3.1) and assign ranks accordingly. The best algorithm will be assigned rank 1, second – 2 and so on. Let $rank(k)_{ij}$ be the rank of algorithm j on cross-validation (CV) split i and k dataset. The average rank for each algorithm:

$$\overline{Rank_j} = \frac{1}{(m \times n)} \left(\sum_{k=1}^{m} \left(\sum_{i=1}^{n} rank(k)_{ij} \right) \right) \tag{3.2}$$

where n is the number of cross-validation splits datasets; m is the number of datasets.

The final ranking is obtained by ordering the average ranks and assigning ranks to the algorithms accordingly.

3.3.3 Statistical significance

Normal distribution

A random variable X is said to follow a normal distribution with parameters μ and S^2 if its probability density function is given by [209]

$$f(x) = \frac{1}{S\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2S^2} \right); \ -\infty < x < \infty, \ -\infty < \mu < \infty, \ S^2 > 0 \tag{3.3}$$

where μ – mean, S – standard deviation, S^2 – variance.

When a random variable X is distributed normally with mean μ and variance S^2 it is written $X \sim N(\mu, S^2)$ [209].

The hypotheses are:

H_0 : the variable is distributed normally

H_1 : the distribution is not normal

Welch's t-test

For statistical significance hypothesis testing is used, in order to make statistical decisions employing experimental data. For this issue Welch's t-test [209] is used, which assumes that both groups of data are sampled from populations that follow a normal distribution, but it does not assume that those two populations have the same variance. Welch test statistic can be written as [209]:

$$T = \frac{|\overline{d}|}{SE} \tag{3.4}$$

where d is the difference between two groups X and Y, then $\overline{d}$ is the difference between two groups means, which can be written $|\overline{d}| = |\overline{X} - \overline{Y}|$. SE – standard error of difference [209]:

$$SE = \sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}} \tag{3.5}$$

where n_X and n_Y – the sizes of groups, S_X and S_Y – the standard deviation of the groups [209]:

$$S_X^2 = \frac{1}{n_X - 1}\sum_{i=1}^{n_X}(X_i - \overline{X})^2, \;\; S_Y^2 = \frac{1}{n_Y - 1}\sum_{i=1}^{n_Y}(Y_i - \overline{Y})^2 \tag{3.6}$$

The equation 3.4 can be rewritten as follows:

$$T(X,Y) = \frac{|\overline{X} - \overline{Y}|}{\sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}}} \tag{3.7}$$

the degrees of freedom is calculated by [209]:

$$df = \left(\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}\right)^2 / \left(\frac{(S_X^2/n_X)^2}{n_X - 1} + \frac{(S_Y^2/n_Y)^2}{n_Y - 1}\right) \tag{3.8}$$

The hypotheses are:

H_0 : difference between the two methods is not significant

H_1 : difference between the two methods is significant

The null hypothesis is rejected at significance level α if $|T| > t$, where t is critical value in the t-distribution table that corresponds to a two-tailed test with α for df – degree of freedom [209].

p-value

The p-value is another statistical measure for hypothesis testing, which could help with decision making when it is necessary to decide whether to reject or not to reject the null hypothesis. In most cases $\alpha = 0.05$ is used.

$p \leq \alpha$ null hypothesis H_0 : is rejected.

$p > \alpha$ null hypothesis H_0 : is not rejected

3.4 Conclusions of Chapter 3

The research methodology for textual data sentiment analysis in large scale datasets is presented in this chapter.

1. A hybrid method for textual data sentiment analysis is proposed. The method consists of:
   - SpeedUP method – this is the main part of the proposed hybrid method, whose aim is to increase the classification speed of classical machine learning algorithms.
   - k-Means clustering method is responsible for training data selection.
   - PSO tuning method – this method performs the tuning of the hyperparameters.
   - Ensemble method – this is the last part of the proposed hybrid method, which performs combination and voting of the machine learning algorithms.
2. A brief description of the datasets, which will be used in the experimental research part is given.
3. For hybrid method classification experiments statistical effectiveness measures are selected: accuracy, precision, recall, harmonic mean and area under the receiver operating characteristics. For methods' ranking average ranks ranking method and Welch's t-test for statistical hypothesis testing are selected.

4. EXPERIMENTS AND RESULTS

In this chapter, experimental cycles, experimental settings, also experiments performed with the proposed hybrid method and presented results are described. For more clarity it was decided to label the method as follows (the variables in curly brackets are optional):

$$\mathbf{ML}^{\{\mathbf{PSO}\}}_{\{\mathbf{n}\}}\_\{\mathbf{km}\}\_\mathbf{s}\_\mathbf{SpeedUP}$$

where:

ML – machine learning algorithm (DT, RF, MNB, LR, LSVM).

n – defines how many classifiers are combined in the case of ensemble. If the ensemble method is not used, then this variable is blank.

PSO – indicates if the PSO tuning method is enabled. If it is not used, then this variable is blank.

km – indicates if k-Means clustering is enabled. If the method is not used, then this variable is blank.

s – defines subset size i.e. 30K, 60K, 120K, 180K etc.

SpeedUP – the SpeedUP method.

The aim of the experiments is to experimentally prove the effectiveness of the proposed method, to perform comparative analysis of it and to recommend a better setup for it. In order to achieve it the experiments are conducted with the five classical machine learning algorithms with a proposed method applied. Comparison with other authors' work is also provided, and the experiment is provided with real-world data. Parts of this chapter are published in [A1],[A2],[A3],[A4],[A5],[A7].

The experiments are performed on the largest labeled text datasets available (See Sec 3.2). A computer with a processor Intel(R) Core(TM) i7-4712MQ CPU @ 2.30 GHz and 16.00 GB installed memory (RAM) was used for the experiments.

4.1 Experimental cycles

Six cycles of experiments are carried out in this dissertation:

1. **Experiment cycle with classical machine learning algorithms**. In this cycle, experiments with five classical machine learning algorithms are performed: multinomial naïve Bayes, random forest, decision tree, linear support vector machines and logistic regression. For

this cycle ordinary dataset split into training (70% of dataset) and testing (30% of dataset) data is used. The aim of this experiment is to compare the results of classical machine learning algorithms and set baselines for comparison with the hybrid method.

2. **Experiment cycle with SpeedUP**. Experiments by applying SpeedUP on the aforementioned classical machine learning algorithms and on the same testing data as in the previous cycle are performed. The main goal is to compare the results between the classical machine learning algorithms and SpeedUP.
3. **Experiment cycle with k-Means clustering**. SpeedUP is supplemented with the k-Means clustering method. The main goal is to show the effectiveness of training data selection. This cycle is also performed on the same testing dataset as the previous experiments.
4. **Experiment cycle with the full proposed hybrid method**. Experiments by supplementing SpeedUP with k-Means clustering and ensemble are performed. Additionally, for LR and LSVM experiments with integrated the PSO tuning method are carried out. The aim of this cycle is to select a better set of parameters for the proposed hybrid method.
5. **Experiment cycle of the comparison of the results**. Comparison with other authors by applying the proposed hybrid method on the same datasets as theirs is performed.
6. **Experiment cycle with real-world data**. Experiments with real-world data collected from the Internet in the public opinion research domain are performed. The results are compared with random search, Bayesian optimization and real results presented by two institutions of public opinion and market research: Vilmorus ltd. and Baltic Surveys.

Detailed experiment settings and datasets are described in further sections. For LSVM classification LinearSVC module with this default parameters is used – all parameters are selected as they are in the module (See App. A). It is similar to SVC (implementation of conventional SVM) with parameter kernel='linear', but it is implemented in terms of LibLinear (a library for large linear classification[7]) rather than LibSVM (a library for support vector machines[8]), so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples

[7] `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[8] `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[11]. Modules MultinomialNB, LogisticRegression, DecisionTreeClassifier and RandomForestClassifier from scikit-learn library with their default parameters (all parameters are selected as they are in the relevant module; see App. A) are used for relative machine learning algorithms (MNB, LR, DT, RF).

The experiments are implemented with Python programming language (v3.6.2) and scikit-learn (v0.19.2) [11]: library for machine learning. Training and testing data is converted into a matrix of TF-IDF features.

4.1.1 Experiment cycle with classical machine learning algorithms

4.1.1.1 Experimental settings

Figure 4.1 presents the diagram of the experiment cycle with classical machine learning algorithms.

List of parameters used in the figure is as follows:

$\mathbf{class_i}$ – certain category of sentiment assigned to the text in dataset
$\mathbf{Trn_i}$ – set of training data selected from $class_1$ or $class_2$
$\mathbf{td_i}$ – set of testing data selected from $class_1$ or $class_2$
$\mathbf{Trn_{CV_i}}$ – set of training data
$\mathbf{td_{CV_i}}$ – set of testing data
$\mathbf{R_{CV_i}}$ – set of results of every cross-validation split
$\overline{\mathbf{R}}_{\mathbf{CV_i}}$ – averaged final results of every cross-validation split

The diagram consists of the following steps:

- The diagram starts with "Textual dataset". It is important to mention that it follows the data preprocessing step (See Fig. 3.1).
- Next "Textual dataset" is split into two classes and each class is split into 10 training Trn_i and testing td_i data folds.
- "Train and Test data" combines each training fold from $class_1$ with training fold from $class_2$ to training data fold Trn_{CV_i}, which contains both classes. The same is done with testing data folds. At the end we have 10 different training and 10 testing data folds. These two steps can be performed automatically by using Stratified ShuffleSplit cross-validator (StratifiedShuffleSplit) from scikit-learn library. The most common split ratio (70/30) into training and testing data is used. Prepared datasets are saved to CSV files for further experiments. The size of training and testing data for both datasets (sentiment140 and AmazonTest) is shown in Table 4.1.

- In the next step the aforementioned training and testing datasets are passed to ML algorithms (MNB, LR, DT, RF, LSVM).
- Finally the obtained classification results R_{CV_i} of each CV fold are averaged for final results $\overline{R}_{CV_i}$.

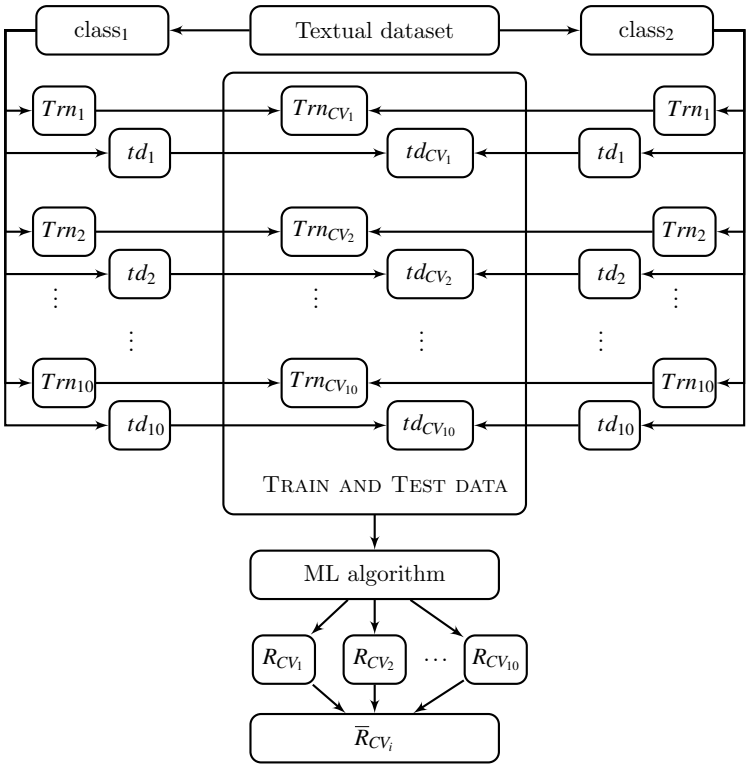


Fig. 4.1. Diagram of the experiment cycle with classical machine learning algorithms

Table 4.1 presents the sizes of training and testing data, when using the aforementioned dataset split.

Table 4.1. Fold size of training and testing data in sentiment140 and AmazonTest datasets

Dataset	Training data (70%)	Testing data (30%)
sentiment140	1.12M	480K
AmazonTest	2.8M	1.2M

Later, the results are compared and ranks are calculated for classical machine learning algorithms by applying an average ranks ranking method (See Subsec 3.3.2).

4.1.1.2 Results

Ten-fold cross-validation was used to compare the effectiveness of the ML algorithms. Table 4.2 gives averaged results of the classical machine learning algorithms, when sentiment140 and AmazonTest datasets were used.

Table 4.2. Averaged effectiveness metrics and ranks of classical ML algorithms in the experiment cycle with classical machine learning algorithms

Method	time (s)	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
				sentiment140 dataset				
MNB	**3.86**	77.53%	77.94%	77.12%	76.78%	78.27%	77.36%	85.80%
rank	1	3	4	3	4	4	3	4
LR	35.62	**79.96%**	79.31%	**80.63%**	**81.06%**	78.85%	**80.17%**	**87.84%**
rank	2	1	2	1	1	2	1	1
LSVM	377.16	79.52%	78.83%	80.24%	80.71%	78.32%	79.76%	87.59%
rank	3	2	3	2	2	3	2	2
RF	1012.73	76.18%	**79.44%**	73.58%	70.66%	**81.71%**	74.79%	86.14%
rank	4	4	1	4	5	1	4	3
DT	6196.69	72.09%	72.05%	72.13%	72.18%	72.00%	72.11%	79.55%
rank	5	5	5	5	3	5	5	5
$\overline{rank}_{LR_1} = 1.375$ | $\overline{rank}_{LSVM_1} = 2.375$ | $\overline{rank}_{MNB_1} = 3.25$ | $\overline{rank}_{RF_1} = 3.25$ | $\overline{rank}_{DT_1} = 4.75$								
				AmazonTest dataset				
MNB	**12.02**	84.47%	85.10%	83.87%	83.58%	85.36%	84.33%	92.33%
rank	1	3	3	3	3	4	3	4
LR	241.31	**90.20%**	90.07%	**90.33%**	**90.36%**	90.03%	**90.21%**	**96.38%**
rank	2	1	2	1	1	2	1	1
LSVM	813.54	89.58%	**91.77%**	87.60%	86.95%	**92.20%**	89.29%	96.33%
rank	3	2	1	2	2	1	2	2
RF	4581.63	80.37%	84.89%	76.89%	73.90%	86.84%	79.02%	92.34%
rank	4	4	4	5	5	3	4	3
DT	98743.92	77.31%	77.31%	77.33%	77.34%	77.27%	77.31%	85.95%
rank	5	5	5	4	4	5	5	5
$\overline{rank}_{LR_2} = 1.375$ | $\overline{rank}_{LSVM_2} = 1.875$ | $\overline{rank}_{MNB_2} = 3$ | $\overline{rank}_{RF_2} = 4$ | $\overline{rank}_{DT_2} = 4.75$								
$\overline{Rank}_{LR} = 1.375$ | $\overline{Rank}_{LSVM} = 2.125$ | $\overline{Rank}_{MNB} = 3.125$ | $\overline{Rank}_{RF} = 3.625$ | $\overline{Rank}_{DT} = 4.75$								

The results of the experiment cycle with classical machine learning algorithms showed that the best accuracy was provided when logistic regression on both datasets was used. The accuracy of linear support vector machines was slightly smaller. Other metrics – NPV, TPR, F_1score – also showed the superiority of the logistic regression on both datasets, while in terms of PPV and TNR it lost slightly to random forest on the sentiment140 dataset and to LSVM on the AmazonTest dataset. The quality measure of

the model's predictions AUC showed that the logistic regression outperformed all classifiers on both datasets. The AUC of linear support vector machines was slightly smaller.

The results of an execution time average showed that the best execution time was achieved by multinomial naïve Bayes on both datasets. The execution time of logistic regression was slightly slower, while the worst result was obtained when decision tree on both datasets was used.

The average ranking $\overline{rank}_{ML_1}$ on the sentiment140 showed that the LR algorithm was of higher rank and LSVM was second best. MNB and RF had the same rank and the last one was DT. The same results of two best algorithms were on the AmazonTest dataset. This time MNB showed a better rank compared with RF, DT was still the last one. The average ranking of both datasets $\overline{Rank}_{ML}$ are distributed as follows: higher rank was obtained by LR, the second place by LSVM, the third place was taken by MNB, the fourth place by RF and the fifth place by DT.

The distribution of classical machine learning algorithms results in terms of ACC, PPV, NPV, TPR, TNR, F_1score and AUC are depicted in Figure 4.2 and Figure 4.3. Figure 4.4 presents the execution times of five classical machine learning algorithms on both datasets.

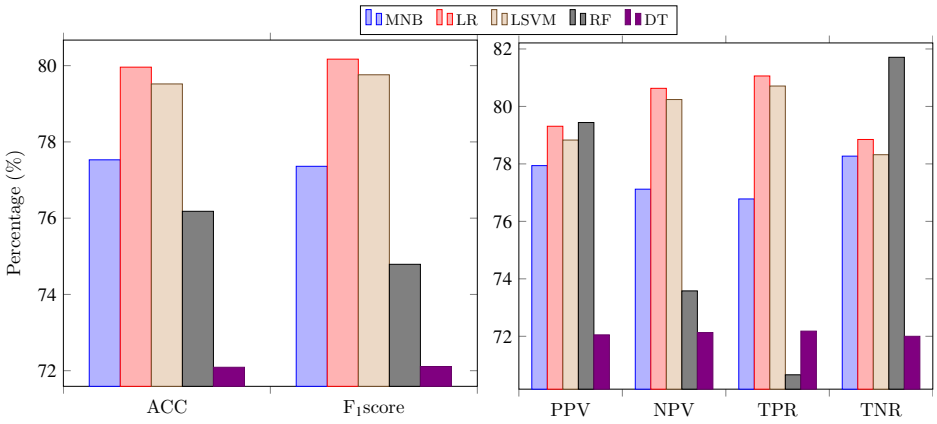


Fig. 4.2. Effectiveness metrics of classical ML algorithms on sentiment140 dataset

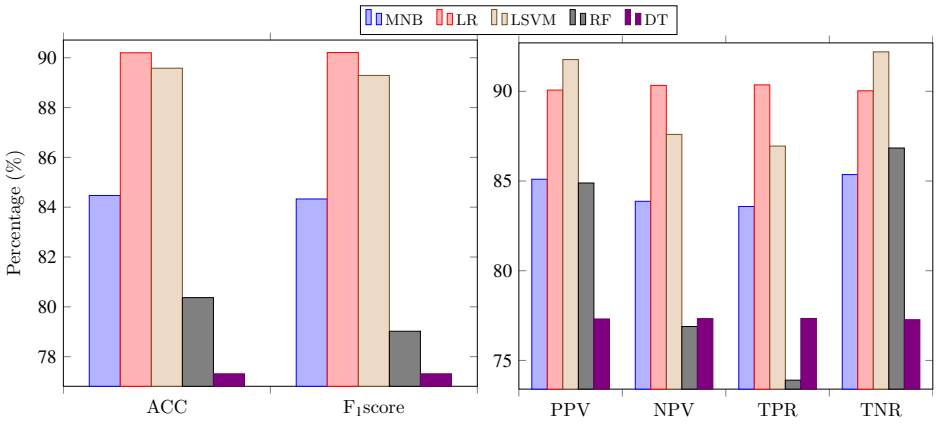


Fig. 4.3. Effectiveness metrics of classical ML algorithms on AmazonTest dataset

As described in Section 3.3, the execution time was measured from the start of ML algorithm learning and finished after classification results were shown. It is evident that the best execution time was when MNB classifier on both datasets was used. LR was the second best and LSVM was the third. The worst results were obtained by DT. The results clearly showed that there is a need of execution time reduction for LR, LSVM and especially for DT.

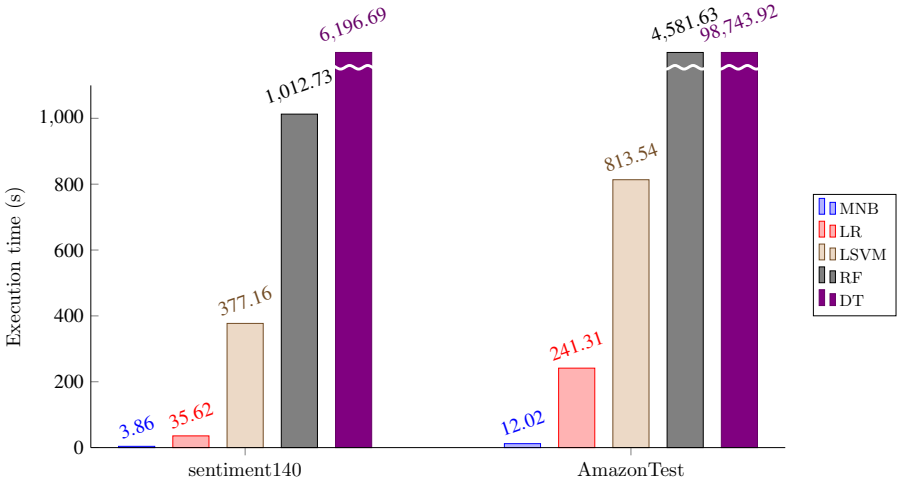


Fig. 4.4. Execution time of classical ML algorithms on both datasets

The obtained results in the experiment cycle with classical machine learning algorithms will be a baseline for further experiments where comparison with the proposed method is performed. Part of the results of this

experiment cycle is published in [A1], [A4] and [A5]. They are slightly different because different CV folds were used and memory RAM was 8GB (while in this dissertation – 16GB).

4.1.2 Experiment cycle with SpeedUP

4.1.2.1 Experimental settings

Experiments with ML_s_SpeedUP are performed here. It means that parameters in the SpeedUP method are set as follows:

$$\textbf{SpeedUP}(ml = \{\text{'LSVM', 'MNB', 'LR', 'DT', 'RF'}\},\ kmeans = 0,\ ensemble = 0,\ pso = 0,\ Subset_{size} = \{30\text{K}, 60\text{K}, 120\text{K}, 180\text{K}\},\ D_{text},\ num_{class} = 2)$$

Values in curly brackets mean that experiments will be performed by setting only one parameter at a time. Therefore, in the case of *ml* parameter, one machine learning algorithm will be selected and experiments will be performed with different $Subset_{size}$: firstly with 30K, then 60K etc. The same is done with the second, third, fourth and fifth machine learning algorithms. It is important to emphasize that these sizes are selected manually by the author after a set of experiments were conducted (it is possible to set different sizes if there is a need). In the further cycles of experiments only $Subset_{size} = 30\text{K}$ will be used, other sizes are presented for testing the SpeedUP method.

Figure 4.5 presents the diagram of the experiment cycle with SpeedUP. Average of all 10 cross-validation folds is given as the final results.

List of parameters used in the figure is as follows:

$\mathbf{CV_i}$ – cross-validation folds, which contain training and testing data
SpeedUP – the main part of the proposed hybrid method
$\mathbf{R_{CV_i}}$ – the results of relevant cross-validation fold, achieved after the SpeedUP method
$\overline{\mathbf{R}}_{\mathbf{CV_i}}$ – the averaged results of all CV folds

The diagram consists of the following steps:

- The diagram starts with $CV_1, CV_2, \ldots, CV_{10}$ in other words they are cross-validation folds each containing training and testing data.
- Later, each CV fold is passed to SpeedUP one by one.
- SpeedUP returns results R_{CV_i} for each CV fold.
- Finally results are averaged – $\overline{R}_{CV_i}$.

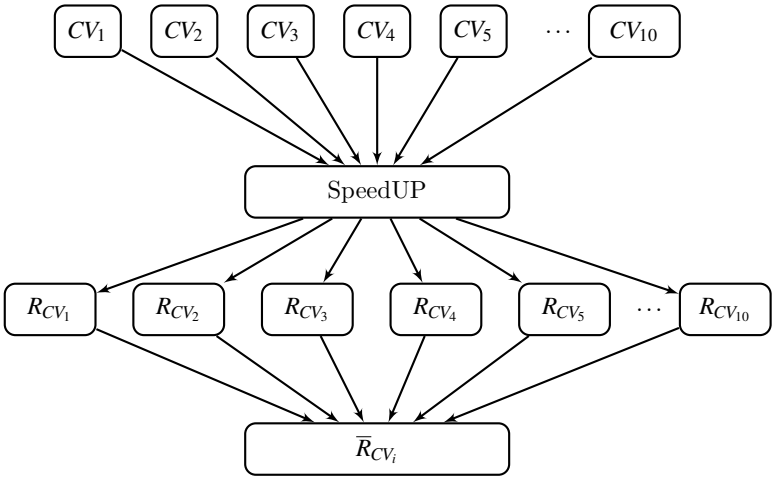


Fig. 4.5. Diagram of the experiment cycle with SpeedUP

Further in Figure 4.6 a detailed diagram of the first CV fold – CV_1 is shown. The same steps are performed for others: CV_2, CV_3 etc.

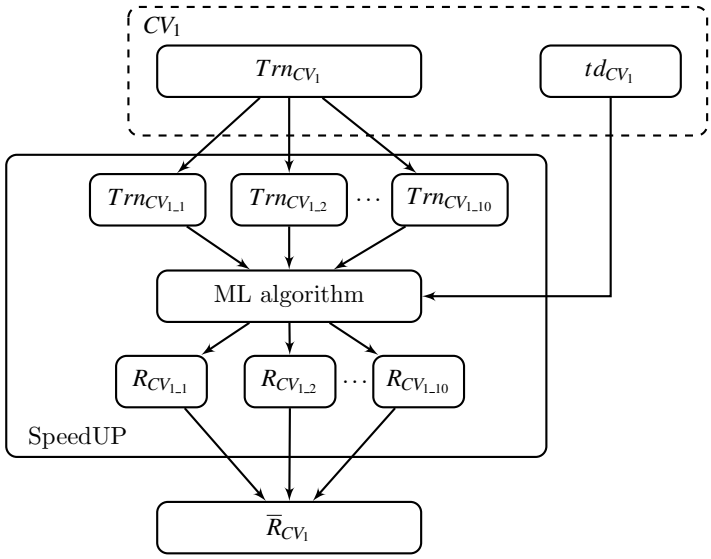


Fig. 4.6. Diagram of the CV_1 in the experiment cycle with SpeedUP

List of parameters used in the figure is as follows:

$\mathbf{Trn_{CV_1}}$ – training data from CV fold

$\mathbf{td_{CV_1}}$ – testing data from CV fold

$\mathbf{Trn_{CV_{1\_n}}}$ – set of training data subsets calculated depending on $Subset_{size}$ (See Table 4.3)

$\mathbf{R_{CV_{1\_i}}}$ – results of every training data subset

$\overline{\mathbf{R}}_{\mathbf{CV_1}}$ – the averaged final results

The diagram consists of the following steps:

- The diagram starts with training and testing data from CV_1.
- Rectangle contains SpeedUP. Because random training data selection is used it was decided to select 10 datasets for training for more objective results, which are calculated depending on $Subset_{size}$.
- Training data are passed to the machine learning algorithm one by one and are performed on the same testing data.
- SpeedUP returns results $R_{CV_{1\_i}}$, which finally are averaged – $\overline{R}_{CV_1}$.

Table 4.3. Training and testing data sizes depending on $Subset_{size}$ for SpeedUP

Dataset	Testing data size (TDs)	Subset size (Ss)	Subsets quantity (SQ) $trunc(TDs/Ss)$	Remainder $TDs-(Ss*SQ)$	Training data depending on Ss $Ss*Split_{ratio}$
sentiment140	480K	30K	16	0	70K
	480K	60K	8	0	140K
	480K	120K	4	0	280K
	480K	180K	2	120K	420K
AmazonTest	1.2M	30K	40	0	70K
	1.2M	60K	20	0	140K
	1.2M	120K	10	0	280K
	1.2M	180K	6	120K	420K

Table 4.3 presents the sizes of training and testing data depending on $Subset_{size}$. In SpeedUP it is performed automatically. For example, $Subset_{size}$ is set to 30K and split ratio is 70% for training and 30% for testing ($Split_{ratio} = 70/30$); then training data ratio, depending on subset size, will be 30K $*(70/30)$ =70K. The whole testing data contains 480K instances (in the case of sentiment140 dataset). The machine learning algorithm is trained with 70K instances and performed on the testing data, which is automatically divided into subsets (16 subsets) with size 30K (in total 480K) during execution. Practically, the trained machine learning algorithm is performed on 16 testing data subsets and finally all results are combined into one set.

The main goal of the experiment cycle with SpeedUP is to carry out experiments with it and to compare results with the results achieved

by classical ML algorithms in the experiment cycle with classical machine learning algorithms.

4.1.2.2 Results

As described in Section 4.1, in this cycle the experiments were performed with SpeedUP. For all experiments the same training and testing data from the experiment cycle with classical machine learning algorithms were used. Furthermore, testing data was divided into subsets, which contain 30K rows, 60K rows, 120K rows and at finally, 180K rows of a dataset. Depending on subset size training data size was calculated and after that it was randomly selected from the whole training dataset.

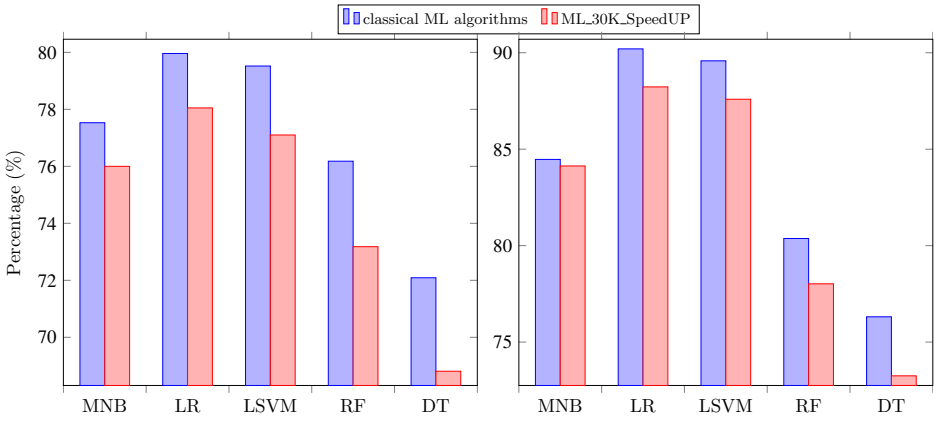
Table 4.4. Averaged effectiveness metrics and ranks of ML_30K_SpeedUP in the experiment cycle with SpeedUP

Method	time (s)	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
sentiment140 dataset								
MNB_	**7.19**	76.00%	76.89%	75.17%	74.34%	77.66%	75.60%	84.54%
rank	1	3	2	3	3	2	3	3
LR_	7.56	**78.05%**	**77.54%**	**78.57%**	**78.96%**	77.13%	**78.25%**	**85.89%**
rank	3	1	1	1	1	3	1	1
LSVM_	7.43	77.10%	76.60%	77.62%	78.05%	76.16%	77.32%	85.36%
rank	2	2	3	2	2	4	2	2
RF_	13.99	73.18%	76.17%	70.81%	67.48%	**78.89%**	71.56%	82.65%
rank	4	4	4	4	5	1	4	4
DT_	36.33	68.81%	68.73%	68.89%	69.03%	68.60%	68.88%	75.95%
rank	5	5	5	5	4	5	5	5
$\overline{rank}_{LR_1} = 1.5$ | $\overline{rank}_{LSVM_1} = 2.375$ | $\overline{rank}_{MNB_1} = 2.5$ | $\overline{rank}_{RF_1} = 3.75$ | $\overline{rank}_{DT_1} = 4.875$								
AmazonTest dataset								
MNB_	68.90	84.13%	84.84%	83.45%	83.11%	85.15%	83.97%	92.19%
rank	3	3	3	3	3	3	3	3
LR_	68.61	**88.23%**	**88.25%**	**88.20%**	**88.19%**	**88.26%**	**88.22%**	94.99%
rank	2	1	1	1	1	1	1	2
LSVM_	**67.51**	87.59%	87.50%	87.68%	87.71%	87.47%	87.60%	**95.04%**
rank	1	2	2	2	2	2	2	1
RF_	80.02	78.02%	82.64%	74.54%	70.94%	85.10%	76.34%	90.07%
rank	4	4	4	4	5	4	4	4
DT_	155.53	73.25%	73.16%	73.34%	73.44%	73.06%	73.30%	81.92%
rank	5	5	5	5	4	5	5	5
$\overline{rank}_{LR_2} = 1.25$ | $\overline{rank}_{LSVM_2} = 1.75$ | $\overline{rank}_{MNB_2} = 3$ | $\overline{rank}_{RF_2} = 4.125$ | $\overline{rank}_{DT_2} = 4.875$								
$\overline{Rank}_{LR} = 1.375$ | $\overline{Rank}_{LSVM} = 2.0625$ | $\overline{Rank}_{MNB} = 2.75$ | $\overline{Rank}_{RF} = 3.9375$ | $\overline{Rank}_{DT} = 4.875$								

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table 4.4 gives averaged results of the ML_30K_SpeedUP, when sentiment140 and AmazonTest datasets were used. The distribution of the results of the accuracy compared with baseline is depicted in Figure 4.7. The results showed that the best accuracy was achieved when logistic regression is used on both datasets. The accuracy of linear support vector machines was slightly smaller. All classifiers resulted with lower accuracy compared with baseline. This happened because the less data was used for training – only 70K.

Other metrics – PPV, NPV, TPR, $F_1 score$ – also showed the superiority of the logistic regression on both datasets, while in terms of TNR there was a slight loss to random forest on the sentiment140 dataset. The quality measure of the model's predictions AUC also showed that the logistic regression outperformed all classifiers on sentiment140 datasets, while LSVM outperformed other classifiers on AmazonTest dataset.



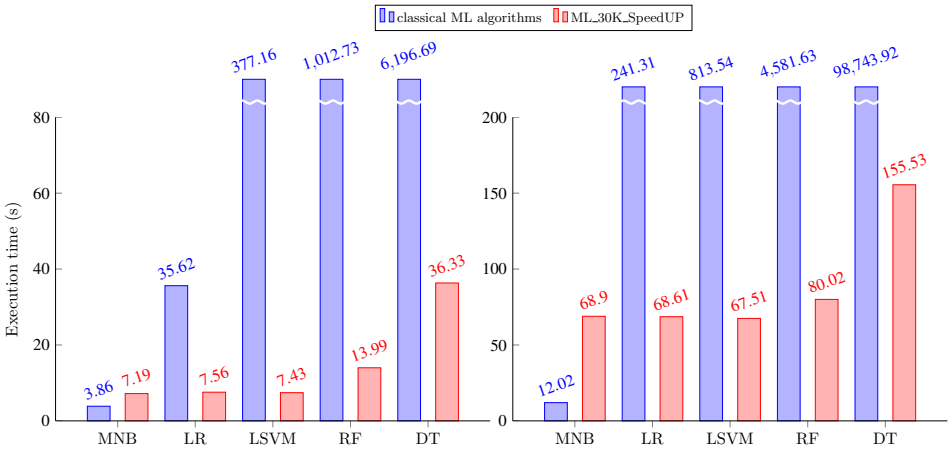
(a) sentiment140 (b) AmazonTest

Fig. 4.7. Accuracy of the ML_30K_SpeedUP and classical ML algorithms

The results of an execution time average showed that the best execution time was achieved by multinomial naïve Bayes on sentiment140 dataset and by LSVM on the AmazonTest dataset.

The average ranking $\overline{rank}_{ML_1}$ on the sentiment140 showed that the higher rank was obtained by the LR algorithm and LSVM was second best. The same results of the two best algorithms were achieved on the AmazonTest dataset. The average ranking of both datasets $\overline{Rank}_{ML}$ is distributed as follows: higher rank was obtained by LR, the second place by LSVM, the third place was reached by MNB, the fourth by RF and the fifth by DT.

Figure 4.8 shows the execution time of the ML_30K_SpeedUP and classical ML algorithms. It is evident that execution time was reduced for LR, LSVM, RF and DT compared with classical ML algorithms. However in the case of MNB it performed slightly worse compared with baseline.



(a) sentiment140 (b) AmazonTest

Fig. 4.8. Execution time of ML_30K_SpeedUP and classical ML algorithms

The best results were achieved by RF and DT on both datasets, whose execution time was reduced: in the case of RF up to 72.3x times when they were applied on the sentiment140 dataset and up to 57.2x times when were applied on the AmazonTest reviews dataset; in the case of DT it was reduced up to 170.5x and up to 634.8x times respectively. The other classifiers also reported reduced execution time: LR – 4.7x, LSVM – 50.7x, when the sentiment140 dataset was used, and LR – 3.5x, LSVM – 12.1x on AmazonTest.

The obtained results showed that SpeedUP fits better for DT in execution time metric, but it also experienced bigger loses in accuracy metrics compared with baseline. According to the ranking results, DT was removed from further steps of this experiments cycle.

The following experiments were performed by using subset size 60K rows of a dataset. Table 4.5 gives averaged results of the ML_60K_SpeedUP, when sentiment140 and AmazonTest datasets were used. The results showed that the best accuracy was achieved when LR on both datasets was used. The accuracy of LSVM was slightly smaller.

Other metrics – PPV, NPV, TPR, $F_1 score$ – also showed the supe-

riority of the logistic regression on both datasets, while in terms of TNR there was a slight loss to random forest on the sentiment140 dataset.

Table 4.5. Averaged effectiveness metrics and ranks of ML_60K_SpeedUP in the experiment cycle with SpeedUP method

Method	time (s)	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
				sentiment140 dataset				
MNB_	**7.11**	76.51%	77.30%	75.76%	75.06%	77.96%	76.17%	85.00%
rank	1	3	3	3	3	2	3	3
LR_	8.88	**78.71%**	**78.15%**	**79.31%**	**79.72%**	77.71%	**78.93%**	**86.58%**
rank	3	1	1	1	1	3	1	1
LSVM_	8.82	77.92%	77.37%	78.50%	78.93%	76.91%	78.14%	86.13%
rank	2	2	2	2	2	4	2	2
RF_	29.83	74.06%	77.14%	71.62%	68.40%	**79.72%**	72.51%	83.71%
rank	4	4	4	4	4	1	4	4
$\overline{rank}_{LR_1} = 1.5 \mid \overline{rank}_{LSVM_1} = 2.25 \mid \overline{rank}_{MNB_1} = 2.625 \mid \overline{rank}_{RF_1} = 3.625$								
				AmazonTest dataset				
MNB_	**67.64**	84.25%	84.95%	83.59%	83.26%	85.25%	84.10%	92.27%
rank	1	3	3	3	3	4	3	3
LR_	70.45	**88.81%**	**88.77%**	**88.85%**	**88.86%**	**88.76%**	**88.82%**	95.42%
rank	3	1	1	1	1	1	1	2
LSVM_	68.99	88.26%	88.16%	88.38%	88.41%	88.12%	88.28%	**95.43%**
rank	2	2	2	2	2	2	2	1
RF_	98.10	78.60%	83.23%	75.10%	71.63%	85.57%	77.00%	90.64%
rank	4	4	4	4	4	3	4	4
$\overline{rank}_{LR_2} = 1.375 \mid \overline{rank}_{LSVM_2} = 1.875 \mid \overline{rank}_{MNB_2} = 2.875 \mid \overline{rank}_{RF_2} = 3.875$								
$\overline{Rank}_{LR} = 1.4375 \mid \overline{Rank}_{LSVM} = 2.0625 \mid \overline{Rank}_{MNB} = 2.75 \mid \overline{Rank}_{RF} = 3.75$								

Underscore "_" means that 60K_SpeedUP should be added at the end.

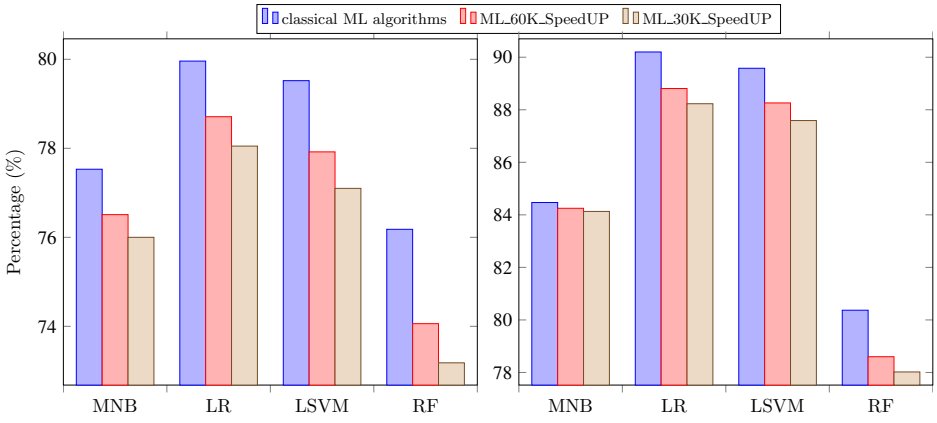
The quality measure of the model's predictions AUC also showed that the logistic regression outperformed all classifiers on sentiment140 datasets, while LSVM outperformed other classifiers on AmazonTest dataset.

The results of an execution time average showed that the best execution time was achieved by MNB on both datasets.

The average ranking on the sentiment140 and AmazonTest datasets showed that as in the previous experiments, the higher rank was of LR algorithm and the second best was LSVM. The other classifiers lined up as follows: the third place was taken by MNB, the fourth by RF.

The distribution of the results of the accuracy compared with baseline is depicted in Figure 4.9. As in the previous experiments, when subset size 30K rows of a dataset was used, all classifiers resulted in slightly lower

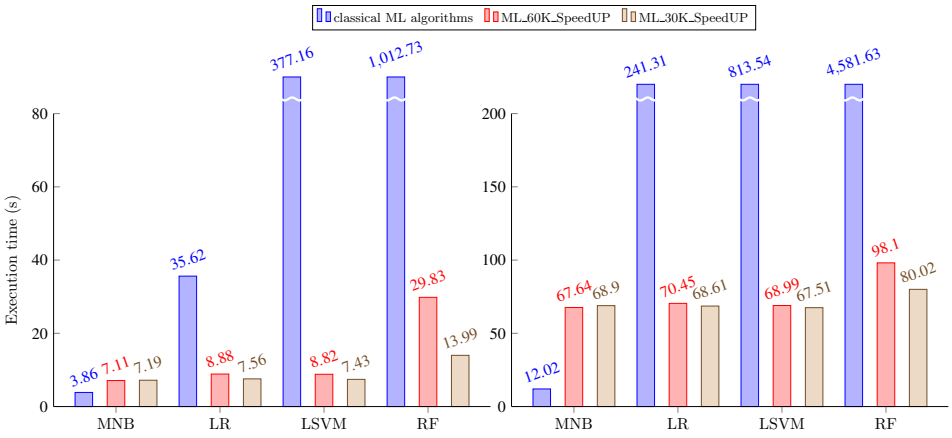
accuracy compared with baseline.



(a) sentiment140 (b) AmazonTest

Fig. 4.9. Accuracy of ML_30K_SpeedUP, ML_60K_SpeedUP and classical ML algorithms

Figure 4.10 shows the execution time of the ML_30K_SpeedUP, ML_60K_SpeedUP and classical ML algorithms. The results clearly showed that execution time of three classifiers was reduced compared with baseline, but again in the case of MNB it increased up to 1.8x on sentiment140 and up to 5.6x on AmazonTest datasets.



(a) sentiment140 (b) AmazonTest

Fig. 4.10. Execution time of ML_30K_SpeedUP, ML_60K_SpeedUP and classical ML algorithms

Compared with experiments when subset size was 30K rows of a

dataset, experiments with subset size of 60K rows slightly lost in execution time metric. Execution time was reduced compared with baseline: LR – 4.1x, LSVM – 42.7x, RF – 33.9x when sentiment140 dataset was used and LR – 3.4x, LSVM – 11.7x, RF – 10.3x on AmazonTest dataset.

The obtained results showed that the SpeedUP method did not suit MNB and it was removed from further steps of this experiment cycle. According to ranks, RF was also removed as the weakest classifier.

The following experiments were performed by using subset size 120K rows of a dataset. The results obtained by classifiers are shown in Table 4.6.

Table 4.6. Averaged effectiveness metrics and ranks of ML_120K_SpeedUP in the experiment cycle with SpeedUP

Method	time (s)	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
sentiment140 dataset								
LR_	**12.26**	**79.23%**	**78.62%**	**79.86%**	**80.29%**	**78.16%**	**79.44%**	**87.11%**
rank	1	1	1	1	1	1	1	1
LSVM_	18.50	78.57%	77.97%	79.19%	79.64%	77.49%	78.79%	86.73%
rank	2	2	2	2	2	2	2	2
$\overline{rank}_{LR_1} = 1 \mid \overline{rank}_{LSVM_1} = 2$								
AmazonTest dataset								
LR_	78.76	**89.28%**	**89.20%**	**89.36%**	**89.38%**	**89.17%**	**89.29%**	**95.76%**
rank	2	1	1	1	1	1	1	1
LSVM_	**74.31**	88.82%	88.70%	88.93%	88.96%	88.67%	88.83%	95.74%
rank	1	2	2	2	2	2	2	2
$\overline{rank}_{LR_2} = 1.125 \mid \overline{rank}_{LSVM_2} = 1.875$								
$\overline{Rank}_{LR} = 1.0625 \mid \overline{Rank}_{LSVM} = 1.9375$								

Underscore "_" means that 120K_SpeedUP should be added at the end.

The results showed that logistic regression performed better in all effectiveness metrics – ACC, PPV, NPV, TPR, TNR, F_1score, AUC – on both datasets. The results of an execution time average showed the superiority of LR on sentiment140 dataset, while LSVM performed better on AmazonTest.

The average ranking on the sentiment140 and AmazonTest datasets showed that as in the previous experiments, LR achieved a higher rank and LSVM was second best.

The following experiments were performed by using subset size 180K rows of dataset. The results obtained by classifiers are shown in Table 4.7.

Table 4.7. Averaged effectiveness metrics and ranks of ML_180K_SpeedUP in the experiment cycle with SpeedUP

Method	time (s)	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
sentiment140 dataset								
LR_	**16.49**	**79.47%**	**78.84%**	**80.14%**	**80.57%**	**78.37%**	**79.70%**	**87.36%**
rank	1	1	1	1	1	1	1	1
LSVM_	48.02	78.90%	78.28%	79.54%	79.99%	77.80%	79.12%	87.02%
rank	2	2	2	2	2	2	2	2
$\overline{rank}_{LR_1} = 1 \mid \overline{rank}_{LSVM_1} = 2$								
AmazonTest dataset								
LR_	87.15	**89.50%**	**89.40%**	**89.61%**	**89.63%**	**89.37%**	**89.52%**	**95.91%**
rank	2	1	1	1	1	1	1	1
LSVM_	**83.97**	89.08%	88.97%	89.20%	89.24%	88.93%	89.10%	95.89%
rank	1	2	2	2	2	2	2	2
$\overline{rank}_{LR_2} = 1.125 \mid \overline{rank}_{LSVM_2} = 1.875$								
$\overline{Rank}_{LR} = 1.0625 \mid \overline{Rank}_{LSVM} = 1.9375$								

Underscore "_" means that 180K_SpeedUP should be added at the end.

The results showed that logistic regression provided better in all effectiveness metrics – ACC, PPV, NPV, TPR, TNR, F_1score, AUC – on both datasets. The results of an execution time average showed the superiority of LR on sentiment140 dataset, while LSVM performed better on AmazonTest.

The average ranking on the sentiment140 and AmazonTest datasets showed that as in the previous experiments, LR achieved a higher rank and LSVM was second best.

Table 4.8. Averaged effectiveness metrics of ML_s_SpeedUP and classical ML algorithms on sentiment140 in the experiment cycle with SpeedUP

Method	time (s)	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
classical LSVM	377.16	**79.52%**	**78.83%**	**80.24%**	**80.71%**	**78.32%**	**79.76%**	**87.59%**
LSVM_30K_	**7.43**	77.10%	76.60%	77.62%	78.05%	76.16%	77.32%	85.36%
LSVM_60K_	8.82	77.92%	77.37%	78.50%	78.93%	76.91%	78.14%	86.13%
LSVM_120K_	18.50	78.57%	77.97%	79.19%	79.64%	77.49%	78.79%	86.73%
LSVM_180K_	48.02	78.90%	78.28%	79.54%	79.99%	77.80%	79.12%	87.02%
classical LR	35.62	**79.96%**	**79.31%**	**80.63%**	**81.06%**	**78.85%**	**80.17%**	**87.84%**
LR_30K_	**7.56**	78.05%	77.54%	78.57%	78.96%	77.13%	78.25%	85.89%
LR_60K_	8.88	78.71%	78.15%	79.31%	79.72%	77.71%	78.93%	86.58%
LR_120K_	12.26	79.23%	78.62%	79.86%	80.29%	78.16%	79.44%	87.11%
LR_180K_	16.49	79.47%	78.84%	80.14%	80.57%	78.37%	79.70%	87.36%

Underscore "_" means that SpeedUP should be added at the end.

Table 4.8 presents the results of both machine learning algorithms: LR and LSVM from all experiments including the classical ML algorithms, when the sentiment140 dataset was used.

The results showed that both classical LR and LSVM outperformed ML_s_SpeedUP in terms of ACC, PPV, NPV, TPR, TNR, $F_1 score$ and AUC. The difference of accuracy average was not very big: in the case of LR difference it was 0.49%-1.91% and in the case of LSVM it was 0.62%-2.42%.

The distribution of results in terms of ACC, PPV, NPV, TPR, TNR and $F_1 score$ of LSVM is depicted in Figure 4.11 and of LR in Figure 4.12.

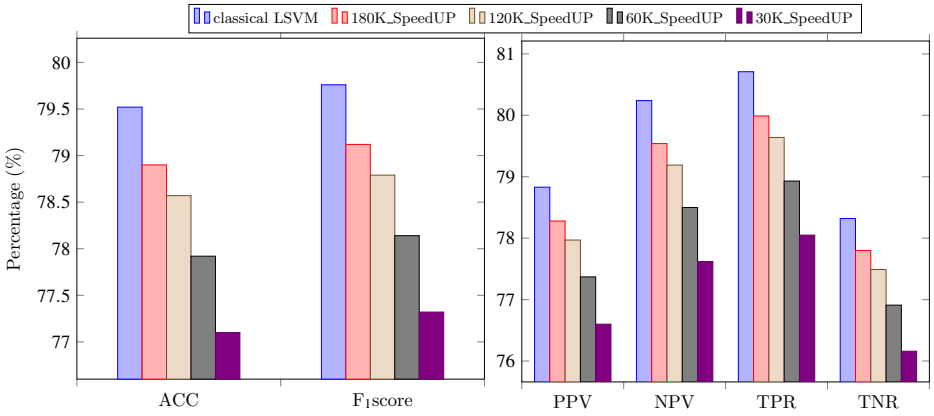


Fig. 4.11. Effectiveness metrics of LSVM_s_SpeedUP and classical LSVM on sentiment140

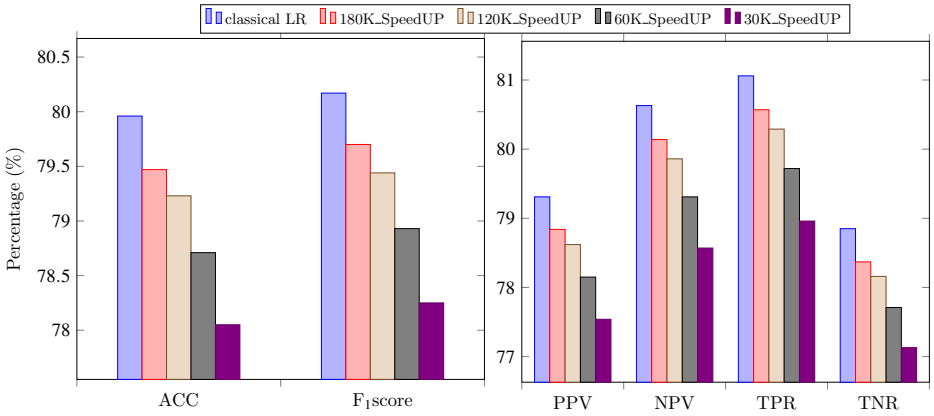


Fig. 4.12. Effectiveness metrics of LR_s_SpeedUP and classical LR on sentiment140

Table 4.9 presents the results of both machine learning algorithms:

LR and LSVM from all experiments including the classical ML algorithms, when AmazonTest dataset was used. As in the case of sentiment140 dataset the results showed that both classical LR and LSVM outperformed ML_s_ SpeedUP method in terms of ACC, PPV, NPV, TPR, TNR, F_1score and AUC. The difference of accuracy average was not very big: in the case of LR it was 0.7%-1.97% and in the case of LSVM it was 0.5%-1.99%.

Table 4.9. Averaged effectiveness metrics of ML_s_SpeedUP and classical ML algorithms on AmazonTest in the experiment cycle with SpeedUP

Method	time (s)	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
classical LSVM	813.54	**89.58%**	**91.77%**	**87.60%**	**86.95%**	**92.20%**	**89.29%**	**96.33%**
LSVM_30K_	**67.51**	87.59%	87.50%	87.68%	87.71%	87.47%	87.60%	95.04%
LSVM_60K_	68.99	88.26%	88.16%	88.38%	88.41%	88.12%	88.28%	95.43%
LSVM_120K_	74.31	88.82%	88.70%	88.93%	88.96%	88.67%	88.83%	95.74%
LSVM_180K_	83.97	89.08%	88.97%	89.20%	89.24%	88.93%	89.10%	95.89%
classical LR	241.31	**90.20%**	**90.07%**	**90.33%**	**90.36%**	**90.03%**	**90.21%**	**96.38%**
LR_30K_	**68.61**	88.23%	88.25%	88.20%	88.19%	88.26%	88.22%	94.99%
LR_60K_	70.45	88.81%	88.77%	88.85%	88.86%	88.76%	88.82%	95.42%
LR_120K_	78.76	89.28%	89.20%	89.36%	89.38%	89.17%	89.29%	95.76%
LR_180K_	87.15	89.50%	89.40%	89.61%	89.63%	89.37%	89.52%	95.91%

Underscore "_" means that SpeedUP should be added at the end.

The distribution of results in terms of ACC, PPV, NPV, TPR, TNR and F_1score of LSVM is depicted in Figure 4.13 and of LR in Figure 4.14. Figure 4.15 shows the execution time of the ML_s_SpeedUP and classical ML algorithms from all experiments on both datasets.

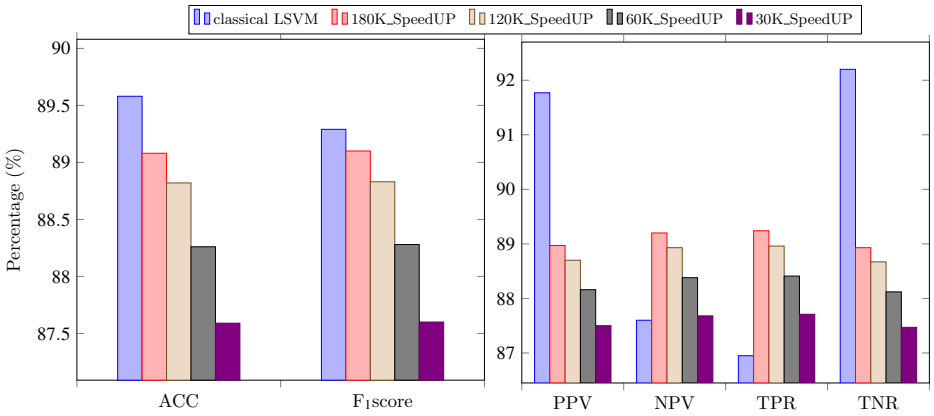


Fig. 4.13. Effectiveness metrics of LSVM_s_SpeedUP and classical LSVM on AmazonTest

Comparison of the results between LR and LSVM also showed the superiority of LR in all metrics – ACC, PPV, NPV, TPR, TNR, $F_1 score$ on both datasets. LR slightly lost to LSVM in execution time metrics, when 30K and 60K rows of dataset applied on sentiment140 were used. In the case of AmazonTest, LR lost to LSVM on all SpeedUP splits: 30K, 60K, 120K and 180K rows.

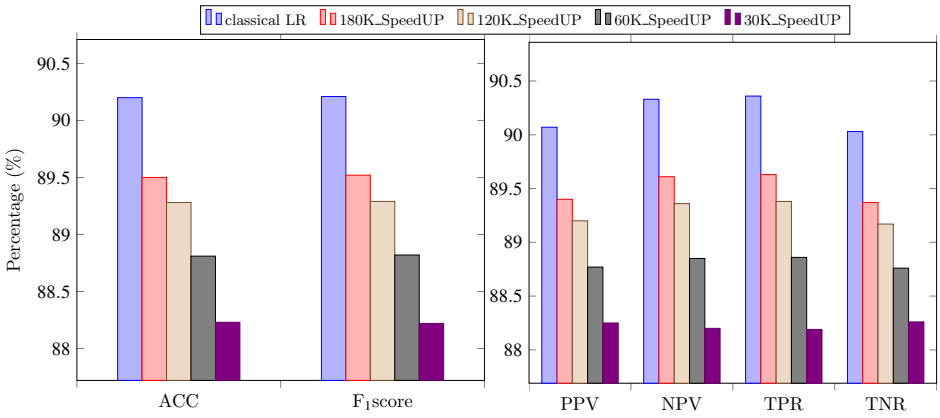
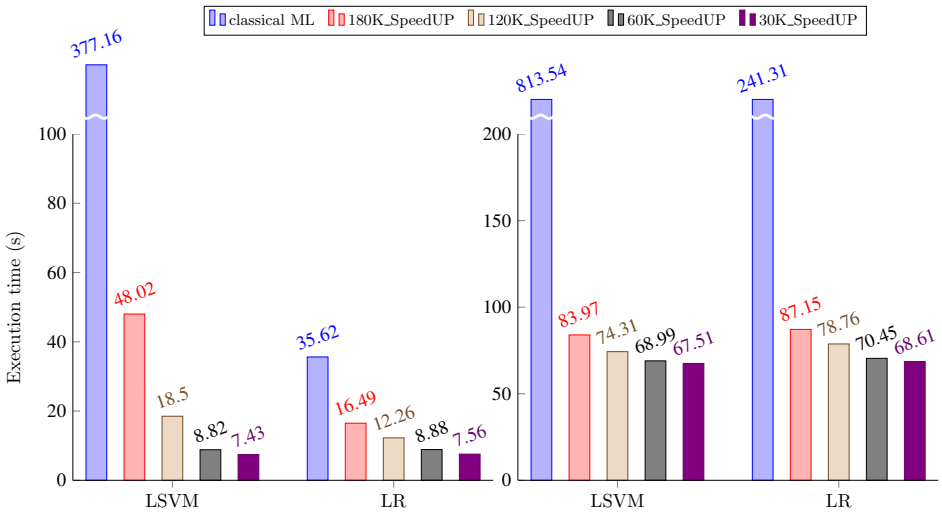


Fig. 4.14. Effectiveness metrics of LR_s_SpeedUP and classical LR on AmazonTest



(a) sentiment140 (b) AmazonTest

Fig. 4.15. Execution time of ML_s_SpeedUP and classical ML algorithms on both datasets

Considering the results, it was concluded that the SpeedUP method was suitable for classical algorithms LR, LSVM, RF and DT, while in the case of MNB it provided opposite results. The best execution time of all classifiers was when 30K subset size was used. Part of the results of this experiment cycle is published in [A4]. They are slightly different because different CV folds were used, memory RAM was 8GB (while in this dissertation – 16GB) and experiments were performed only with SVM. In the following cycle experiments to increase accuracy of ML_30K_SpeedUP by applying the k-Means clustering method are performed.

4.1.3 Experiment cycle with k-Means clustering

4.1.3.1 Experimental settings

Experiments with ML_km_30K_SpeedUP are performed. It means that parameters in the SpeedUP method are set as follows:

$$\textbf{SpeedUP}(ml = \{\text{'LSVM', 'MNB', 'LR', 'DT', 'RF'}\},\ kmeans = 1,\ ensemble = 0,\ pso = 0,\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

Values in curly brackets mean that experiments will be performed by setting only one parameter at a time. For this cycle, the same diagram (Figure 4.5) presented in the experiment cycle with SpeedUP is used. In the diagram only the cross-validation splits are different. Figure 4.16 presents the diagram of the experiment cycle with k-Means clustering. The dashed line in the diagram represents steps for additional calculations executed before a concrete step is joined to it. For the final results average is calculated.

List of parameters used in the figure is as follows:

$\mathbf{Trn_{CV_1}}$ – training data from CV fold
$\mathbf{td_{CV_1}}$ – testing data from CV fold
$\mathbf{Trn_{km_{1\_i}}}$ – set of training data after k-Means clustering is applied
$\mathbf{td_{km_i}}$ – testing data for k-Means clustering
$\mathbf{R_{km_{1\_i}}}$ – results of every training data subset
$\mathbf{Trn_{km(best\_i)}}$ – training data selected by k-Means clustering
$\mathbf{R_{km(best\_i)}}$ – results of every training data subset
$\overline{\mathbf{R}}_{\mathbf{CV_1}}$ – the averaged final results

The diagram consists of the following steps:

- The diagram starts with training and testing data from CV_1.

- The smaller rectangle contains the k-Means clustering method. As later an ensemble method will be tested, it was decided to select 10 datasets for training by performing k-Means clustering.
- After training data is selected it is passed to ML algorithm one by one. For validating, training data from CV_1 is used from which training data selected by k-Means clustering is excluded.
- Dependending on the obtained results, n (*ensemble* value in SpeedUP, see Expr. 3.1) training data sets are selected for next step – ML_km_30K_SpeedUP.
- The difference between ML_30K_SpeedUP and ML_km_30K_SpeedUP is in the number of training data sets. In ML_30K_SpeedUP, 10 training data sets are used, while in this cycle its depends on value n.
- Last step contains averaged $\overline{R}_{CV_1}$ results returned by ML_km_30K_SpeedUP method.

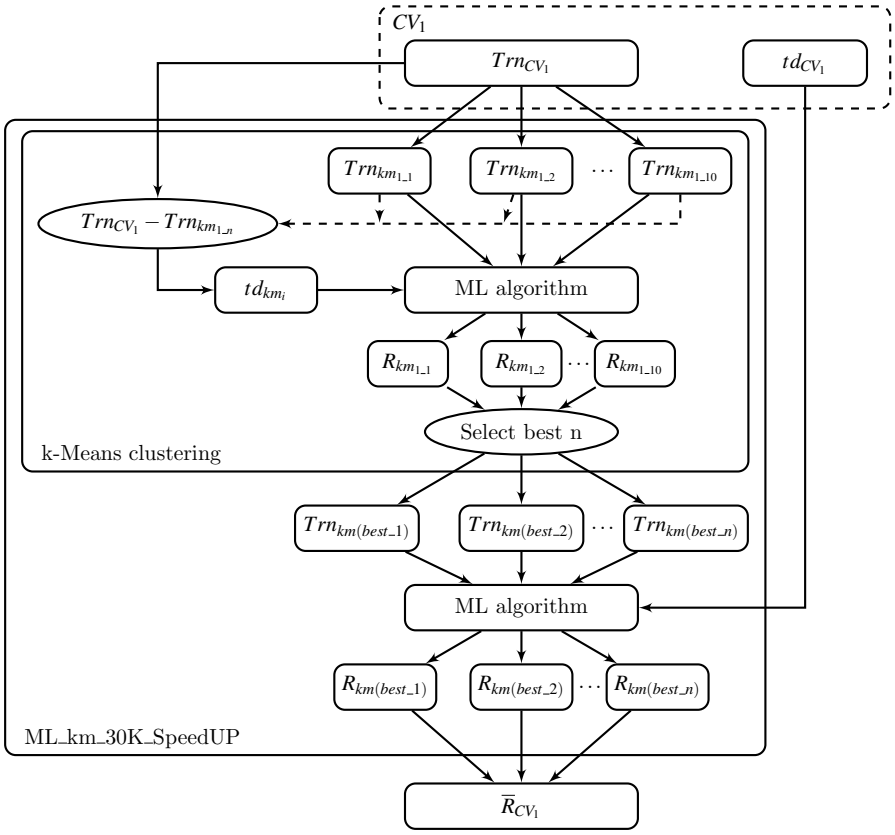


Fig. 4.16. Diagram of the CV_1 in the experiment cycle with k-Means clustering

Table 4.10 shows detailed experimental settings for data splitting.

Table 4.10. Training and testing data sizes for ML_km_30K_SpeedUP

Dataset	Testing data size (TDs)	Subset size (Ss)	Subsets quantity (SQ) $trunc(TDs/Ss)$	Remainder $TDs-(Ss*SQ)$	Training data depending on Ss $Ss*Split_{ratio}$
sentiment140	480K	30K	16	0	70K
AmazonTest	1.2M	30K	40	0	70K

While k-Means clustering needs a defined number of clusters, the number of cluster selection method (See Algorithm 7) is proposed. This algorithm measure time depends on the number of clusters obtained after running k-Means. The main goal is to find the cluster configuration which will be optimal in terms of time and separation. It is worth mentioning that this method is not a part of the proposed hybrid method – this is a separate method, which could help to decide how many clusters should be used. By default, an optimal number of clusters (k_{opt}) is 120, which is hard-coded in the proposed method.

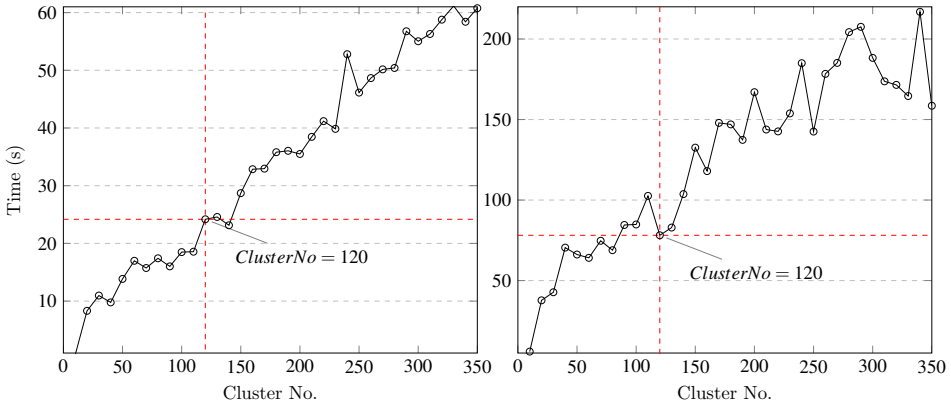
Algorithm 7 Number of cluster selection

Require: $cluster\_range$ – max number of clusters, D_{train} – training dataset
  $results \leftarrow pandas.DataFrame$(columns=['k','time'])
  for $k = 2 : cluster\_range$ **do**
    $clf \leftarrow kmeans(k)$
    $start = time.time()$
    $clf.predict(D_{train})$
    $stop = time.time()$
    $time = stop - start$
    $results \leftarrow results.append$('k': k, 'time' : $time$, $ignore\_index = True$)
  end for
  $k_{opt} \leftarrow results['k'].loc[abs(results['time']-(results['time'].max/3)).idxmin()]$
  Output : k_{opt} – optimal number of clusters.

The main goal of this experiment cycle is to perform experiments with ML_km_30K_SpeedUP and to compare the results with the results achieved in the experiment cycle with classical ML algorithms and by ML_30K_SpeedUP.

4.1.3.2 Results

In this cycle, the same CV datasets, which were prepared in the experiment cycle with classical ML algorithms are used. Firstly an optimal number of clusters were selected (See Algorithm 7) for k-Means clustering. It is worth mentioning that the more clusters will be selected, the more different data will there will be in training dataset, while it will be the reason of clustering speed loses, and vice verse – less clusters, the less different data, but a higher clustering speed. Taking the visual output (See Fig. 4.17) of Algorithm 7 into account it was decided that an optimal number of clusters should be 120. This value was selected by the author based on a number of experimental attempts.



(a) sentiment140 dataset (b) AmazonTest dataset

Fig. 4.17. Number of cluster selection results

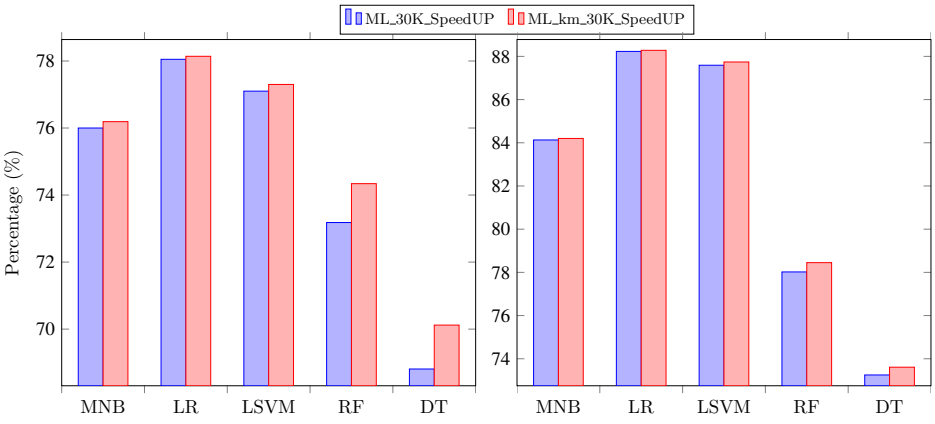
The obtained value was hard-coded into k-Means clustering, so it cannot be selected as SpeedUP variable. After a number of clusters was selected, k-Means clustering was run 10 times on each part of CV training dataset to perform a selection of 10 training datasets from each CV fold. Ten-fold cross-validation was performed for each selected training datasets splitting it into training (70%) and validating (30%) datasets and passing it into ML algorithm. Depending on the averaged accuracy, five training datasets were selected from each CV training fold (See Subsec 4.1.3).

Table 4.11 gives averaged results of the ML_30K_SpeedUP and ML_km_30K_SpeedUP, when sentiment140 and AmazonTest datasets were used. The results indicated that ML_km_30K_SpeedUP outperformed ML_30K_SpeedUP almost in all effectiveness metrics on both datasets.

Table 4.11. Averaged effectiveness metrics of ML_30K_SpeedUP and ML_km_30K_SpeedUP in the experiment cycle with k-Means clustering

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
sentiment140 dataset							
MNB_	76.00%	**76.89%**	75.17%	74.34%	**77.66%**	75.60%	84.54%
MNB_km_	**76.19%**	76.69%	**75.70%**	**75.24%**	77.14%	**75.96%**	**84.58%**
LR_	78.05%	77.54%	78.57%	78.96%	77.13%	78.25%	85.89%
LR_km_	**78.14%**	**77.66%**	**78.64%**	**79.02%**	**77.26%**	**78.33%**	**85.98%**
LSVM_	77.10%	76.60%	77.62%	78.05%	76.16%	77.32%	85.36%
LSVM_km_	**77.30%**	**76.78%**	**77.83%**	**78.26%**	**76.33%**	**77.51%**	**85.55%**
RF_	73.18%	76.17%	70.81%	67.48%	78.89%	71.56%	82.65%
RF_km_	**74.34%**	**77.18%**	**72.05%**	**69.13%**	**79.56%**	**72.93%**	**83.60%**
DT_	68.81%	68.73%	68.89%	69.03%	68.60%	68.88%	75.95%
DT_km_	**70.12%**	**70.00%**	**70.25%**	**70.43%**	**69.81%**	**70.21%**	**77.04%**
AmazonTest dataset							
MNB_	84.13%	84.84%	**83.45%**	**83.11%**	85.15%	83.97%	92.19%
MNB_km_	**84.20%**	**85.10%**	83.34%	82.91%	**85.48%**	**83.99%**	**92.23%**
LR_	88.23%	88.25%	**88.20%**	**88.19%**	88.26%	88.22%	94.99%
LR_km_	**88.28%**	**88.43%**	88.13%	88.08%	**88.47%**	**88.25%**	**95.01%**
LSVM_	87.59%	87.50%	87.68%	87.71%	87.47%	87.60%	95.04%
LSVM_km_	**87.74%**	**87.75%**	**87.72%**	87.71%	**87.76%**	**87.73%**	**95.11%**
RF_	78.02%	82.64%	74.54%	70.94%	85.10%	76.34%	90.07%
RF_km_	**78.45%**	**83.45%**	**74.76%**	**70.98%**	**85.92%**	**76.71%**	**90.34%**
DT_	73.25%	73.16%	73.34%	73.44%	73.06%	73.30%	81.92%
DT_km_	**73.61%**	**73.68%**	**73.54%**	**73.46%**	**73.75%**	**73.57%**	**82.23%**

Underscore "_" means that 30K_SpeedUP should be added at the end.



(a) sentiment140 (b) AmazonTest

Fig. 4.18. Accuracy of ML_30K_SpeedUP and ML_km_30K_SpeedUP

However, there was a slight loss in terms of PPV and TNR when MNB on sentiment140 was used; also in terms of NPV and TPR on AmazonTest when MNB and LR were used.

The distribution of the accuracy results of ML_30K_SpeedUP and ML_km_30K_SpeedUP is depicted in Figure 4.18.

The average ranking (See Table B.1 in App. B) on the sentiment140 and AmazonTest datasets again showed the superiority of LR and LSVM: LR achieved a higher rank while LSVM was second best.

Table 4.12. Accuracy comparison between ML_30K_SpeedUP and ML_km_30K_SpeedUP in the experiment cycle with k-Means clustering using Welch's t-test

Method	MNB	LR	LSVM	RF	DT
sentiment140 dataset					
Statistics	$\lvert\overline{d}\rvert = 0.19$ $SE = 0.026$ $\lvert T\rvert = 7.21$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.09$ $SE = 0.023$ $\lvert T\rvert = 4.05$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.19$ $SE = 0.016$ $\lvert T\rvert = 11.79$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 1.16$ $SE = 0.033$ $\lvert T\rvert = 34.80$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 1.31$ $SE = 0.026$ $\lvert T\rvert = 49.52$ $t_{table} = 2.12$
p-value	$< .00001$	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected
AmazonTest dataset					
Statistics	$\lvert\overline{d}\rvert = 0.07$ SE =0.021 $\lvert T\rvert = 3.26$ $t_{table} = 2.145$	$\lvert\overline{d}\rvert = 0.05$ $SE = 0.007$ $\lvert T\rvert = 6.59$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.15$ $SE = 0.007$ $\lvert T\rvert = 20.06$ $t_{table} = 2.145$	$\lvert\overline{d}\rvert = 0.44$ $SE = 0.044$ $\lvert T\rvert = 9.95$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.36$ $SE = 0.03$ $\lvert T\rvert = 11.85$ $t_{table} = 2.11$
p-value	0.0057	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$).

Before applying Welch's t-test it was checked whether variables follow a standard normal distribution (See Table B.16 in App. B). The calculation was performed based on the accuracy of each CV fold achieved

by relevant methods (See Table B.2 in App. B). The results in Table 4.12 showed that the biggest difference between accuracy average ($\overline{d}$) of the two methods was 1.31%, when DT on sentimen140 dataset was used and in the case of RF – $\overline{d} = 0.44\%$ – on AmazonTest. However, these ML methods had the lowest accuracy compared with LR, LSVM and MNB. Considering Welch's t-test and p-value, the achieved accuracy by ML_km_30K_SpeedUP was significant compared with ML_30K_SpeedUP, because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection 3.3.3).

Part of the results of this experiment cycle is published in [A2]. They are slightly different because different CV folds were used, experiments were performed only with LSVM and the C_Tuning method was applied.

In the cycle with the full proposed hybrid method experiments to increase the accuracy of ML_km_30K_SpeedUP are performed by applying an ensemble method and additionally PSO tuning for LSVM and LR.

4.1.4 Experiment cycle with the full proposed hybrid method

4.1.4.1 Experimental settings

Experiments with $\mathrm{ML}^{\{PSO\}}_{\{n\}}$_km_30K_SpeedUP are performed here. It means that parameters in the SpeedUP method are set as follows:

$$\textbf{SpeedUP}(ml = \{\text{'LSVM', 'MNB', 'LR', 'DT', 'RF'}\},\ kmeans = 1,$$
$$ensemble = \{3, 5\},\ pso = \{0, 1\},\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

Values in curly brackets mean that experiments will be performed by setting only one parameter at a time.

For this cycle the same diagram (Figure 4.5) presented in the experiment cycle with SpeedUP is used. Only cross-validation splits are different in the diagram. Figure 4.19 presents the diagram of the experiment cycle with the full proposed hybrid method. The diagram is slightly simplified, because it is the same as in Figure 4.16, only the last step is different. As the final results of CV fold, the results achieved from three and five voters are taken.

List of parameters used in the figure is as follows:

$\mathbf{Trn_{CV_1}}$ – training data from CV fold
$\mathbf{td_{CV_1}}$ – testing data from CV fold
$\mathbf{Trn_{km(best\_i)}}$ – training data selected by k-Means clustering
$\mathbf{R_{km(best\_i)}}$ – results of every training data subset

$\mathbf{R_{CV_1}}$ – the final results

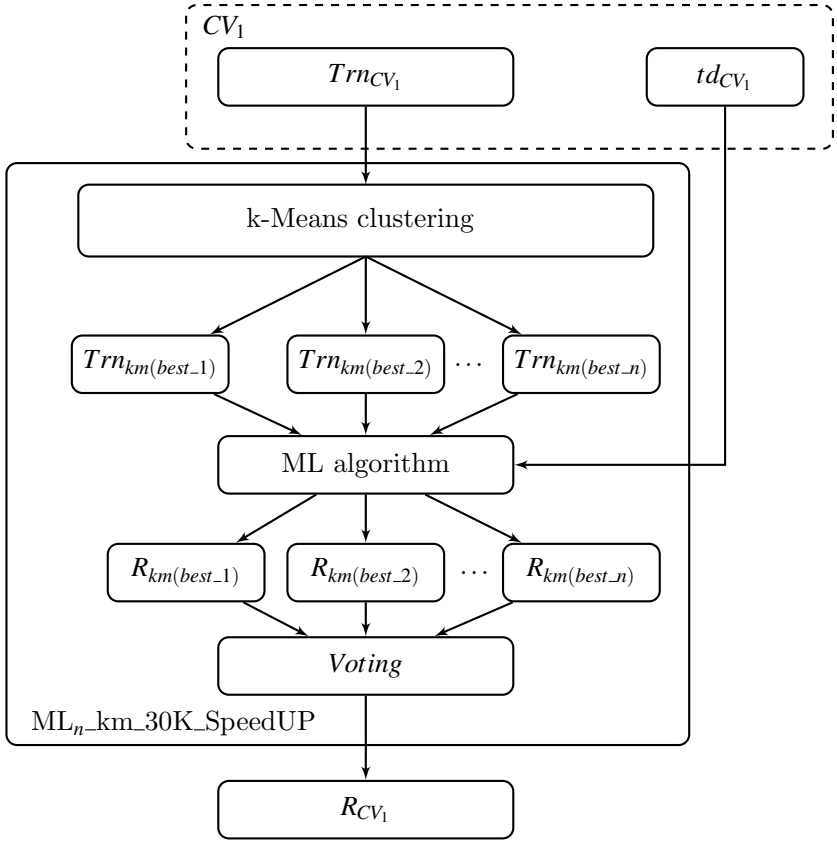


Fig. 4.19. Diagram of the CV_1 in the experiment cycle with the full proposed hybrid method

The diagram consists of the following steps:

- The diagram starts with training and testing data from CV_1
- In the next step k-Means clustering (See Fig. 4.16) is applied for selecting the number of training data set by value *ensemble* in SpeedUP.
- After that selected training data is passed to ML algorithm.
- Last step contains results after voting is performed.

Further this cycle is expanded by applying PSO tuning to LSVM and LR machine learning algorithms.

Experiments with ML^{PSO}_km_30K_SpeedUP and ML$_n^{PSO}$_km_30K_SpeedUP are performed here. It means that parameters in the SpeedUP method are set as follows:

$$\textbf{SpeedUP}(ml = \{\text{'LSVM', 'LR'}\},\ kmeans = 1,\ ensemble = 0,\ pso = 1,$$
$$Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

and for the second case:

$$\textbf{SpeedUP}(ml = \{\text{'LSVM', 'LR'}\},\ kmeans = 1,\ ensemble = \{3, 5\},\ pso = 1,$$
$$Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

Values in curly brackets mean that experiments will be performed by setting only one parameter at a time. Starting C value is defined by using cross-validated grid search over a predefined grid of possible C values (this range is selected manually and is hard-coded in PSO tuning) and after that range is minimized by applying C_Tuning.

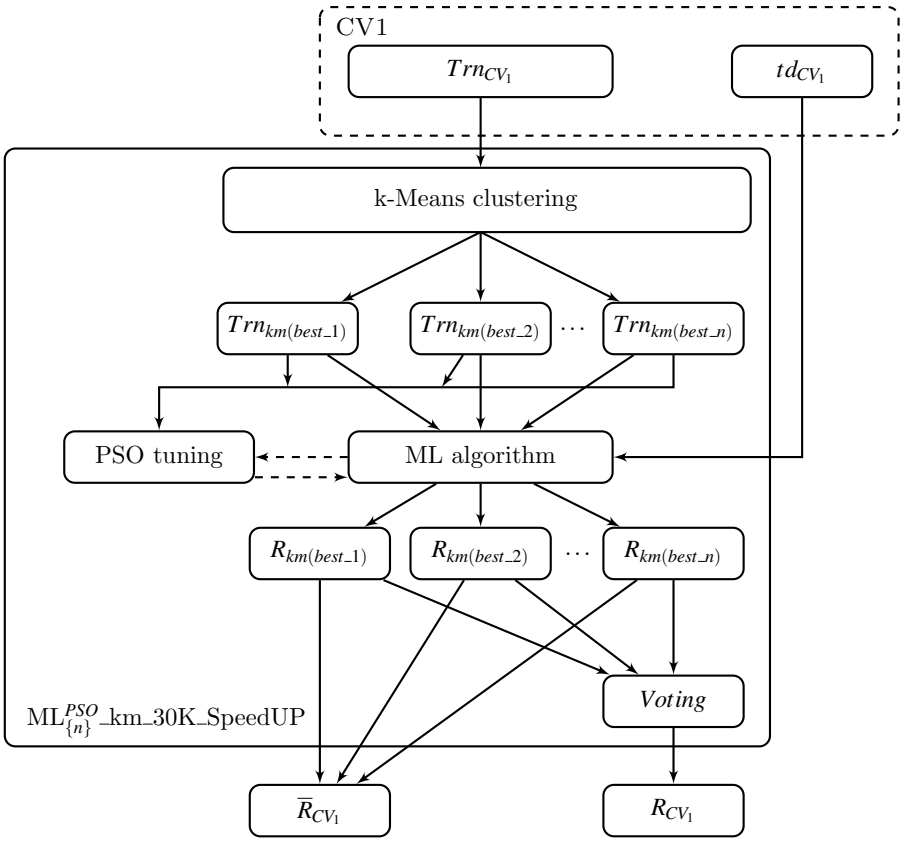


Fig. 4.20. Diagram of the CV_1 in the extended experiment cycle with the full proposed hybrid method

Figure 4.20 presents an extended diagram of the experiment cycle with the full proposed hybrid method. The parameters in the diagram

are the same as in Figure 4.19, so they are not described here. The steps of the diagram are also described above, only one step is added – PSO tuning. In this step tuning of the parameters of LR and LSVM by using training datasets is performed, which are achieved after k-Means clustering. The dashed line in the diagram represents steps for additional calculations executed before a concrete step is joined to it.

The main goal is to propose the recommended settings for the hybrid method.

4.1.4.2 Results

In this cycle, the same CV datasets, which were prepared in the experiment cycle with classical ML algorithms are used. It is assumed that the ensemble contains three ($n = 3$) and five ($n = 5$) classifiers.

Table 4.13. Averaged effectiveness metrics of ML_km_30K_SpeedUP, ML_n_km_30K_SpeedUP and classical ML algorithms in the experiment cycle with the full proposed hybrid method on sentiment140

Method	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
sentiment140 dataset							
MNB_km_	76.19%	76.69%	75.70%	75.24%	77.14%	75.96%	84.58%
MNB_3_km_	76.93%	77.39%	76.49%	76.10%	77.77%	76.74%	85.38%
MNB_5_km_	77.15%	77.60%	76.72%	76.34%	77.97%	76.97%	85.57%
classical MNB	**77.53%**	**77.94%**	**77.12%**	**76.78%**	**78.27%**	**77.36%**	**85.80%**
LR_km_	78.14%	77.66%	78.64%	79.02%	77.26%	78.33%	85.98%
LR_3_km_	78.68%	78.15%	79.23%	79.62%	77.75%	78.88%	86.44%
LR_5_km_	78.84%	78.28%	79.42%	79.82%	77.86%	79.04%	86.53%
classical LR	**79.96%**	**79.31%**	**80.63%**	**81.06%**	**78.85%**	**80.17%**	**87.84%**
LSVM_km_	77.30%	76.78%	77.83%	78.26%	76.33%	77.51%	85.55%
$LSVM_3$_km_	78.51%	77.87%	79.18%	79.65%	77.36%	78.75%	86.84%
$LSVM_5$_km_	78.93%	78.25%	79.65%	80.14%	77.72%	79.18%	87.14%
classical LSVM	**79.52%**	**78.83%**	**80.24%**	**80.71%**	**78.32%**	**79.76%**	**87.59%**
RF_km_	74.34%	77.18%	72.05%	69.13%	79.56%	72.93%	83.60%
RF_3_km_	76.14%	79.51%	73.47%	70.45%	81.84%	74.70%	85.25%
RF_5_km_	**76.87%**	**80.44%**	**74.06%**	**71.02%**	**82.73%**	**75.44%**	**85.79%**
classical RF	76.18%	79.44%	73.58%	70.66%	81.71%	74.79%	86.14%
DT_km_	70.12%	70.00%	70.25%	70.43%	69.81%	70.21%	77.04%
DT_3_km_	73.49%	73.45%	73.52%	73.57%	73.41%	73.51%	82.16%
DT_5_km_	**75.06%**	**75.08%**	**75.05%**	**75.03%**	**75.10%**	**75.05%**	**83.64%**
classical DT	72.09%	72.05%	72.13%	72.18%	72.00%	72.11%	79.55%

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table 4.13 gives averaged results of ML_km_30K_SpeedUP, ML_3_km_30K_SpeedUP, ML_5_km_30K_SpeedUP and classical ML algorithms on sentiment140 and Table 4.14 on AmazonTest datasets.

Table 4.14. Averaged effectiveness metrics of ML_km_30K_SpeedUP, ML_n_km_30K_SpeedUP and classical ML algorithms in the experiment cycle with the full proposed hybrid method on AmazonTest

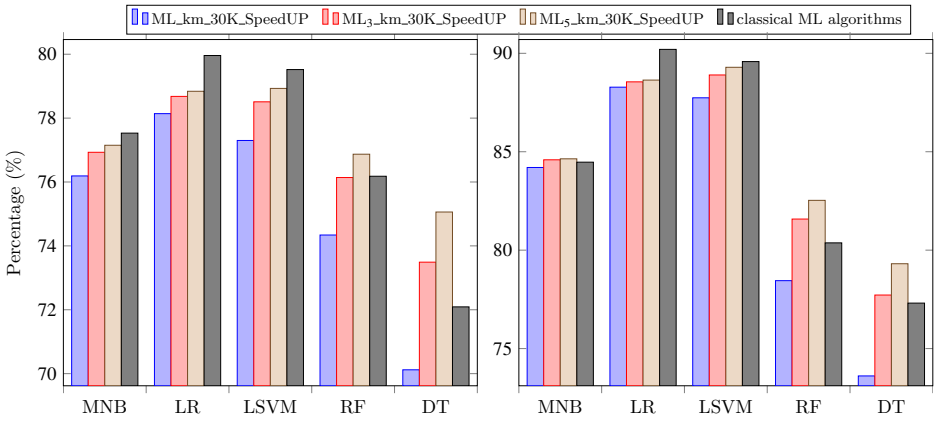
Method	ACC	PPV	NPV	TPR	TNR	F_1*score*	AUC
AmazonTest dataset							
MNB_km_	84.20%	85.10%	83.34%	82.91%	85.48%	83.99%	92.23%
MNB_3_km_	84.59%	85.51%	83.72%	83.30%	85.88%	84.39%	92.56%
MNB_5_km_	**84.64%**	**85.54%**	83.78%	83.37%	**85.91%**	**84.44%**	**92.59%**
classical MNB	84.47%	85.10%	**83.87%**	**83.58%**	85.36%	84.33%	92.33%
LR_km_	88.28%	88.43%	88.13%	88.08%	88.47%	88.25%	95.01%
LR_3_km_	88.55%	88.70%	88.40%	88.35%	88.75%	88.53%	95.18%
LR_5_km_	88.64%	88.80%	88.48%	88.43%	88.85%	88.62%	95.22%
classical LR	**90.20%**	**90.07%**	**90.33%**	**90.36%**	**90.03%**	**90.21%**	**96.38%**
LSVM_km_	87.74%	87.75%	87.72%	87.71%	87.76%	87.73%	95.11%
$LSVM_3$_km_	88.90%	88.90%	88.89%	88.89%	88.90%	88.90%	95.78%
$LSVM_5$_km_	89.29%	89.30%	**89.27%**	**89.27%**	89.30%	89.28%	95.93%
classical LSVM	**89.58%**	**91.77%**	87.60%	86.95%	**92.20%**	**89.29%**	**96.33%**
RF_km_	78.45%	83.45%	74.76%	70.98%	85.92%	76.71%	90.34%
RF_3_km_	81.58%	87.56%	77.24%	73.61%	89.54%	79.98%	92.45%
RF_5_km_	**82.53%**	**88.95%**	**77.93%**	**74.29%**	**90.77%**	**80.96%**	**92.89%**
classical RF	80.37%	84.89%	76.89%	73.90%	86.84%	79.02%	92.34%
DT_km_	73.61%	73.68%	73.54%	73.46%	73.75%	73.57%	82.23%
DT_3_km_	77.72%	77.97%	77.48%	77.28%	78.16%	77.62%	87.49%
DT_5_km_	**79.31%**	**79.65%**	**78.98%**	**78.73%**	**79.89%**	**79.19%**	**88.75%**
classical DT	77.31%	77.31%	77.33%	77.34%	77.27%	77.31%	85.95%

Underscore "_" means that 30K_SpeedUP should be added at the end.

The results showed the superiority of ensembles compared to ML_km_30K_SpeedUP of all classifiers – MNB, LR, LSVM, RF and DT. The classical algorithms still reported better in terms of ACC, PPV, NPV, TPR, TNR, F_1*score* and AUC, when MNB, LR and LSVM with SpeedUP on sentiment140 and when LR and LSVM on AmazonTest were used. However, classical ML algorithms lost to ensembles ML_3_km_30K_SpeedUP in all effectiveness metrics, when DT on sentiment140 was employed and when RF and DT were used on AmazonTest. In the case of ML_5_km_30K_SpeedUP, classical ML algorithms lost to RF and DT on sentiment140 and to RF and DT on AmazonTest. ML_3_km_30K_SpeedUP and ML_5_km_30K_SpeedUP,

when MNB was used on AmazonTest, also outperformed classical MNB in terms of ACC, PPV, TNR, F_1score and AUC, while there was a slight loss to NPV and TPR.

The distribution of results is depicted in Figure 4.21.



(a) sentiment140 (b) AmazonTest

Fig. 4.21. Accuracy of the ML_km_30K_SpeedUP, ML$_n$_km_30K_SpeedUP and classical ML algorithms

The average ranking $\overline{rank}_{ML_1}$ (See Table B.3 and B.4 in App. B) on the sentiment140, when ensemble ML$_3$_km_30K_SpeedUP was used, showed that LR reached a higher rank and LSVM was second best. However, on AmazonTest and in both cases when ensemble ML$_5$_km_30K_SpeedUP was used, LSVM showed the superiority over LR, and final average ranks are distributed as follows: LSVM reached a higher rank, LR was second best, then MNB and RF followed and DT was the last one.

Before applying Welch's t-test it was checked whether variables follow standard normal distribution (See Table B.16 in App. B).

The calculation was performed based on the accuracy of each CV fold achieved by relevant methods (See Table B.5 in App. B). The results in Table 4.15 showed that the biggest difference between accuracy average ($\overline{d}$) of the two methods was 3.37%, when DT on sentimen140 dataset was used and $\overline{d} = 4.11\%$ – on AmazonTest. LR received the least $\overline{d} = 0.54\%$ on sentiment140 and $\overline{d} = 0.27\%$ – on the AmazonTest. Considering Welch's t-test and p-value, the achieved accuracy by ML$_3$_km_30K_SpeedUP was significant compared with ML_km_30K_SpeedUP, because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection 3.3.3).

Table 4.15. Accuracy comparison between ML_km_30K_SpeedUP and ML_3_km_30K_SpeedUP in the experiment cycle with the full proposed hybrid method using Welch's t-test

Method	MNB	LR	LSVM	RF	DT
sentiment140 dataset					
Statistics	$\lvert\overline{d}\rvert = 0.74$ $SE = 0.035$ $\lvert T\rvert = 20.99$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.54$ $SE = 0.023$ $\lvert T\rvert = 23.94$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 1.21$ $SE = 0.026$ $\lvert T\rvert = 46.38$ $t_{table} = 2.145$	$\lvert\overline{d}\rvert = 1.80$ $SE = 0.031$ $\lvert T\rvert = 57.74$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 3.37$ $SE = 0.031$ $\lvert T\rvert = 109.94$ $t_{table} = 2.131$
p-value	$< .00001$	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected
AmazonTest dataset					
Statistics	$\lvert\overline{d}\rvert = 0.40$ $SE = 0.028$ $\lvert T\rvert = 13.95$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.27$ $SE = 0.008$ $\lvert T\rvert = 32.87$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 1.16$ $SE = 0.011$ $\lvert T\rvert = 105.56$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 3.12$ $SE = 0.044$ $\lvert T\rvert = 70.51$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 4.11$ $SE = 0.038$ $\lvert T\rvert = 108.22$ $t_{table} = 2.131$
p-value	$< .00001$	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$).

The results in Table 4.16 were calculated based on the accuracy of each CV fold achieved by relevant methods (See Table B.6 in App. B). Again, the biggest difference between accuracy average ($\overline{d}$) of the two methods was 1.58%, when DT on sentiment140 dataset was used and $\overline{d} = 1.59\%$ – on AmazonTest dataset. LR received the least $\overline{d} = 0.16\%$ on sentiment140, while MNB – $\overline{d} = 0.05\%$ – on the AmazonTest. Considering Welch's t-test and p-value, the achieved accuracy by ML_5_km_30K_SpeedUP was significant compared with ML_3_km_30K_SpeedUP (except in the case between MNB_3_km_30K_SpeedUP and MNB_5_km_30K_SpeedUP on the AmazonTest dataset where the H_0 hypothesis was not rejected), because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection 3.3.3).

Table 4.16. Accuracy comparison between ML_3_km_30K_SpeedUP and ML_5_km_30K_SpeedUP in the experiment cycle with the full proposed hybrid method using Welch's t-test

Method	MNB	LR	LSVM	RF	DT
sentiment140 dataset					
Statistics	$\lvert\overline{d}\rvert = 0.22$ $SE = 0.035$ $\lvert T\rvert = 6.33$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.16$ $SE = 0.025$ $\lvert T\rvert = 6.28$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.42$ $SE = 0.027$ $\lvert T\rvert = 15.66$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 0.73$ $SE = 0.028$ $\lvert T\rvert = 25.69$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 1.58$ $SE = 0.033$ $\lvert T\rvert = 48.48$ $t_{table} = 2.12$
p-value	$< .00001$	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected
AmazonTest dataset					
Statistics	$\lvert\overline{d}\rvert = 0.05$ $SE = 0.03$ $\lvert T\rvert = 1.67$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.09$ $SE = 0.008$ $\lvert T\rvert = 11.44$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 0.39$ $SE = 0.011$ $\lvert T\rvert = 34.43$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.95$ $SE = 0.043$ $\lvert T\rvert = 22.32$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 1.59$ $SE = 0.041$ $\lvert T\rvert = 39.14$ $t_{table} = 2.12$
p-value	0.1126	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	not rejected not rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$).

As ensembles ML_5_km_30K_SpeedUP (in the case of MNB, RF and DT) achieved better accuracy compared with baseline, then there is a need to check the statistical significance of the results. The results presented in Table 4.17 were calculated based on the accuracy of each CV fold achieved by relevant methods (See Table B.8 in App. B). The biggest difference between accuracy average ($\overline{d}$) of the two methods was 2.98%, when DT on sentiment140 was used and in the case of RF –$\overline{d} = 2.17\%$ – on AmazonTest. Considering Welch's t-test and p-value, the achieved accuracy by ML_5_km_30K_SpeedUP was significant compared with classical ML, because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection 3.3.3).

Table 4.17. Accuracy comparison between classical ML and ML5_km_30K_SpeedUP in the experiment cycle with the full proposed hybrid method using Welch's t-test

Method	RF	DT	MNB	RF	DT
	sentiment140 dataset		AmazonTest dataset		
Statistics	$\lvert\overline{d}\rvert = 0.69$ $SE = 0.03$ $\lvert T\rvert = 23.01$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 2.98$ $SE = 0.024$ $\lvert T\rvert = 125.51$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 0.17$ $SE = 0.021$ $\lvert T\rvert = 8.00$ $t_{table} = 2.201$	$\lvert\overline{d}\rvert = 2.17$ $SE = 0.079$ $\lvert T\rvert = 27.44$ $t_{table} = 2.228$	$\lvert\overline{d}\rvert = 2.00$ $SE = 0.037$ $\lvert T\rvert = 53.71$ $t_{table} = 2.11$
p-value	$< .00001$	$< .00001$	$< .00001$	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$).

Table 4.18. Accuracy comparison between $LSVM_5$_km_30K_SpeedUP and LR_5_km_30K_SpeedUP in the experiment cycle with the full proposed hybrid method using Welch's t-test

Method	LR_5_km_ $LSVM_5$_km_	LR_5_km_ $LSVM_5$_km_
	sentiment140	AmazonTest
Statistics	$\lvert\overline{d}\rvert = 0.09$ $SE = 0.025$ $\lvert T\rvert = 3.67$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.64$ $SE = 0.008$ $\lvert T\rvert = 79.27$ $t_{table} = 2.228$
p-value	0.0021	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$). Underscore "_" means that 30K_SpeedUP should be added at the end.

The results in Table 4.13 and Table 4.14 showed that in the case

of ML_5_km_30K_SpeedUP, LSVM outperformed LR, then there is a need to check the statistical significance of the results. The results in Table 4.18 were calculated based on the accuracy of each CV fold achieved by relevant methods (See Table B.7 in App. B). The results showed that LSVM outperformed LR with difference between accuracy average ($\overline{d}$) of the two methods – 0.09% on sentiment140 and $\overline{d}$ = 0.64% – on the AmazonTest. Statistically the difference was significant, because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection Subsection 3.3.3).

Futher experiments with LSVM and LR machine learning algorithms by applying PSO tuning are performed. Figure 4.22 depicts the results of the PSO tuning method performed with LSVM on sentiment140 training dataset, where each point was represented by Δ value. Δ_d is always equal to 0, because this is difference between accuracy obtained by using default C ($C = 1$) value (LSVM_km_30K_SpeedUP). Δ_{PSO} is the accuracy difference between $LSVM^{PSO}$_km_30K_SpeedUP and LSVM_km_30K_SpeedUP.

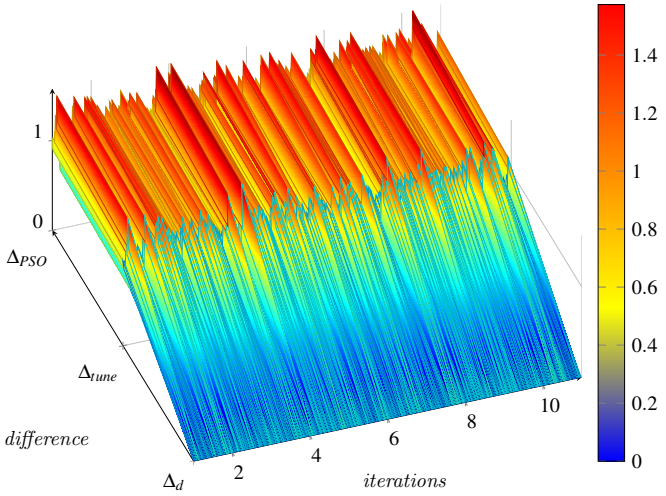


Fig. 4.22. Results of PSO tuning on the sentiment140 dataset performed with LSVM

It is important to stress that the results shown in Figures (4.22, 4.23, 4.24) and Tables (4.19, 4.20, 4.23) are only a visual depiction of the PSO tuning results during execution process; they are processed inside the method and are not shown as final results. Selected C value is automaticaly set to relevant ML algorithm.

The results clearly showed that PSO tuning obtained higher accu-

racy (more than 1.2%) compared with the method executed with default C value. Δ_{tune} is intermediate result (the result of C_Tuning) before PSO tuning is executed.

Table 4.19 contains the best results of sentiment140 training data in CV fold proceeded by PSO tuning with LSVM. C_{PSO} presented in the table is the value which is assigned to ML algorithm as parameter.

Table 4.19. Results of PSO tuning on sentiment140 dataset performed for LSVM_km_30K_SpeedUP

CV	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_{tune} ($acc_2 - acc_1$)	C_{tune}	Δ_{PSO} ($acc_3 - acc_1$)	C_{PSO}
CV1	76.97%	78.20%	78.29%	1.23%	0.1	1.32%	0.07326
CV2	77.02%	78.25%	78.40%	1.23%	0.1	1.38%	0.07676
CV3	76.66%	78.11%	78.23%	1.45%	0.1	1.57%	0.08243
CV4	76.75%	78.04%	78.05%	1.29%	0.1	1.30%	0.09315
CV5	76.77%	77.94%	78.12%	1.17%	0.1	1.35%	0.05658
CV6	76.73%	77.99%	78.16%	1.26%	0.1	1.43%	0.06470
CV7	77.15%	78.45%	78.53%	1.30%	0.1	1.38%	0.06151
CV8	77.11%	78.56%	78.59%	1.45%	0.1	1.48%	0.08413
CV9	77.14%	78.35%	78.35%	1.21%	0.1	1.21%	0.09963
CV10	77.20%	78.43%	78.55%	1.23%	0.1	1.35%	0.08645

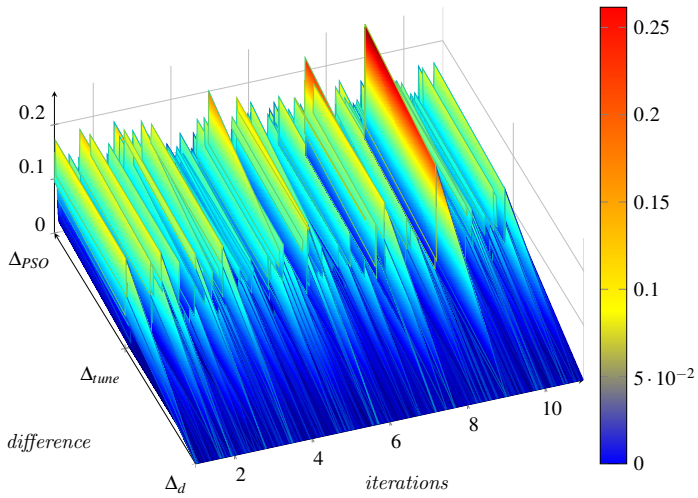


Fig. 4.23. Results of PSO tuning on the sentiment140 dataset performed with LR

Figure 4.23 visually depicts the results of PSO tuning performed with LR on sentiment140 training dataset, where each point is represented by Δ value.

Table 4.20 contains the best results of sentiment140 training data in CV fold proceeded by PSO tuning with LR. C_{PSO} presented in the table is the value which is assigned to ML algorithm as parameter.

Table 4.20. Results of PSO tuning on the sentiment140 dataset performed for LR_km_30K_SpeedUP

CV	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_{tune} ($acc_2 - acc_1$)	C_{tune}	Δ_{PSO} ($acc_3 - acc_1$)	C_{PSO}
CV1	77.83%	77.95%	77.96%	0.12%	0.7	0.13%	0.70398
CV2	78.14%	78.28%	78.29%	0.14%	1.5	0.15%	1.50783
CV3	78.02%	78.11%	78.12%	0.09%	1.6	0.10%	1.60344
CV4	77.98%	78.05%	78.08%	0.07%	0.9	0.10%	0.92663
CV5	78.00%	78.13%	78.14%	0.13%	1.4	0.14%	1.38613
CV6	77.82%	77.93%	77.99%	0.11%	0.7	0.17%	0.66908
CV7	78.14%	78.24%	78.28%	0.1%	0.7	0.14%	0.64271
CV8	78.35%	78.45%	78.46%	0.10%	0.7	0.11%	0.70233
CV9	78.09%	78.17%	78.19%	0.08%	1.2	0.10%	1.24708
CV10	78.01%	78.10%	78.12%	0.09%	1.4	0.11%	1.37735

Table 4.21 gives averaged results of ML_km_30K_SpeedUP and ML^{PSO}_km_30K_SpeedUP on the sentiment140. The results indicated that $LSVM^{PSO}$_km_30K_SpeedUP outperformed the LSVM_km_30K_SpeedUP in all effectiveness metrics, while in the case of LR^{PSO}_km_30K_SpeedUP there was a slight loss to LR_km_30K_SpeedUP in all metrics.

Table 4.21. Averaged effectiveness metrics of ML_km_30K_SpeedUP and ML^{PSO}_km_30K_SpeedUP on the sentiment140 in the experiment cycle with the full proposed hybrid method

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
				sentiment140 dataset			
LR_km_	**78.14%**	**77.66%**	**78.64%**	**79.02%**	**77.26%**	**78.33%**	**85.98%**
LR^{PSO}_km_	78.11%	77.61%	78.63%	79.02%	77.20%	78.30%	85.95%
LSVM_km_	77.30%	76.78%	77.83%	78.26%	76.33%	77.51%	85.55%
$LSVM^{PSO}$_km_	**78.12%**	**77.64%**	**78.62%**	**79.99%**	**77.24%**	**78.31%**	**85.95%**

Underscore "_" means that 30K_SpeedUP should be added at the end.

Before applying Welch's t-test it was checked whether variables follow a standard normal distribution (See Table B.16 in App. B). The results in Table 4.22 were calculated based on the accuracy of each CV fold achieved by relevant methods (See Table B.9 in App. B). They showed that the difference between accuracy average ($\overline{d}$) of the two methods increased by 0.82%, when LSVM was used and in the case of LR it decreased by 0.02%.

Table 4.22. Accuracy comparison between ML^{PSO}_km_30K_SpeedUP and ML_km_30K_SpeedUP in the experiment cycle with the full proposed hybrid method using Welch's t-test

Method	LR	LSVM
	sentiment140 dataset	
Statistics	$\lvert\overline{d}\rvert = 0.02$ $SE = 0.021$ $\lvert T\rvert = 0.874$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 0.82$ $SE = 0.022$ $\lvert T\rvert = 36.80$ $t_{table} = 2.131$
p-value	0.3961	p < .00001
Results	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	not rejected not rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$).

Considering Welch's t-test and p-value, the accuracy achieved by LSVM^{PSO}_km_30K_SpeedUP was significant compared with LSVM_km_30K_SpeedUP, because the H_0 hypothesis was rejected, while in the case of LR^{PSO}_km_30K_SpeedUP and LR_km_30K_SpeedUP was not significant, because the H_0 hypothesis was not rejected (the hypotheses are defined in Subsection 3.3.3). Taking the results into aacount it was decided to exclude LR from the further experiments.

Futher experiments with LSVM on AmazonTest dataset are performed.

Figure 4.24 visually depicts the results of PSO tuning performed with LSVM on AmazonTest training dataset, where each point represented by Δ value.

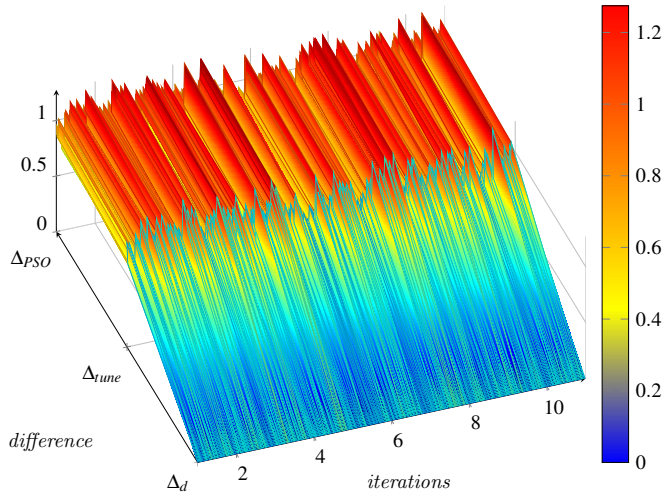


Fig. 4.24. Results of PSO tuning on the AmazonTest dataset performed with LSVM

Table 4.23 contains the best results of AmazonTest training data in CV fold proceeded by PSO tuning with LSVM. C_{PSO} presented in the table is the value which is assigned to LSVM as parameter.

Table 4.23. Results of PSO tuning on AmazonTest performed for LSVM_km_30K_SpeedUP

CV	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_{tune} ($acc_2 - acc_1$)	C_{tune}	Δ_{PSO} ($acc_3 - acc_1$)	C_{PSO}
CV1	87.40%	88.35%	88.47%	0.95%	0.2	1.07%	0.15276
CV2	87.33%	88.34%	88.38%	1.01%	0.2	1.05%	0.17476
CV3	87.11%	88.13%	88.22%	1.02%	0.1	1.11%	0.10908
CV4	87.22%	88.31%	88.37%	1.09%	0.2	1.15%	0.13103
CV5	87.24%	88.15%	88.21%	0.91%	0.2	0.97%	0.18523
CV6	87.28%	88.27%	88.33%	0.99%	0.2	1.05%	0.18628
CV7	87.45%	88.49%	88.56%	1.04%	0.2	1.11%	0.14771
CV8	87.46%	88.54%	88.59%	1.08%	0.2	1.13%	0.21088
CV9	87.46%	88.46%	88.52%	1.0%	0.3	1.06%	0.28866
CV10	87.39%	88.63%	88.64%	1.24%	0.2	1.25%	0.19760

Table 4.24 gives averaged results of ML_km_30K_SpeedUP and ML^{PSO}_km_30K_SpeedUP on the AmazonTest. The results indicated that LSVM^{PSO}_km_30K_SpeedUP outperformed LSVM_km_30K_SpeedUP in all

effectiveness metrics – ACC, PPV, NPV, TPR, TNR, F_1score and AUC on AmzonTest dataset.

Table 4.24. Averaged effectiveness metrics of LSVM_km_30K_SpeedUP and LSVM^{PSO}_km_30K_SpeedUP on the AmazonTest in the experiment cycle with the full proposed hybrid method

Method	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
LSVM_km_	87.74%	87.75%	87.72%	87.71%	87.76%	87.73%	95.11%
LSVM^{PSO}_km_	**88.45%**	**88.57%**	**88.33%**	**88.29%**	**88.61%**	**88.43%**	**95.24%**

Underscore "_" means that 30K_SpeedUP should be added at the end.

Considering the results it was decided to apply PSO tuning with classical LSVM without SpeedUP. The aim is to increase the accuracy of classical ML algorithm. Results proceeded by PSO tuning for classical LSVM are presented in Appendices B in Table B.12. Next LSVM^{PSO}_km_30K_SpeedUP will be tested with ensembles LSVM$_3^{PSO}$_km_30K_SpeedUP and LSVM$_5^{PSO}$_km_30K_SpeedUP. For that purpose, there is a need to select respectively three and five training datasets from each CV by applying k-Means clustering and PSO tuning on each selected dataset. Results proceeded by PSO tuning on each dataset are presented in Appendices B in Table B.10 for sentiment140 dataset and in Table B.11 for AmazonTest.

Table 4.25 gives averaged results of the proposed method with different settings are set on and also a classical LSVM with PSO tuning applied on it.

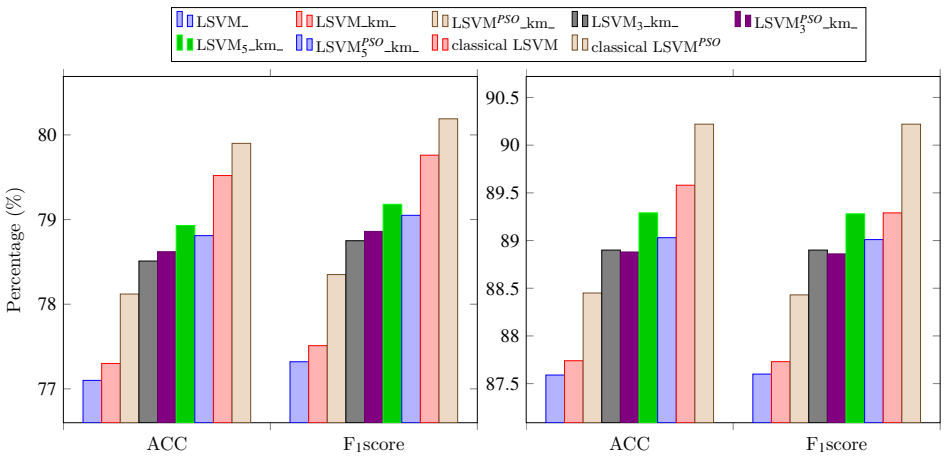
The distribution of the results of the accuracy of the proposed method with different settings is depicted in Figure 4.25.

The results clearly showed that the accuracy of LSVM_30K_SpeedUP was increased up to 1.02%, when it was used with ensembles of classifiers – up to 1.83% on sentiment140. The difference with classical LSVM also decreased – only 0.59% in the case of LSVM$_5$_km_30K_SpeedUP. The accuracy of classical LSVM increased by 0.38%, when PSO tuning was applied on it. The results on AmazonTest dataset also indicated the increased accuracy of ML_30K_SpeedUP. Accuracy increased up to 0.86%, when it was used with ensembles of classifiers up to 1.7%. The accuracy of baseline was increased up to 0.64%, when PSO tuning was applied on it.

Table 4.25. Averaged effectiveness metrics of the proposed hybrid method

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
sentiment140 dataset							
LSVM_	77.10%	76.60%	77.62%	78.05%	76.16%	77.32%	85.36%
LSVM_km_	77.30%	76.78%	77.83%	78.26%	76.33%	77.51%	85.55%
$LSVM^{PSO}$_km_	**78.12%**	**77.64%**	**78.62%**	**79.99%**	**77.24%**	**78.31%**	**85.95%**
$LSVM_3$_km_	78.51%	77.87%	79.18%	79.65%	77.36%	78.75%	**86.84%**
$LSVM_3^{PSO}$_km_	**78.62%**	**77.98%**	**79.29%**	**79.76%**	**77.48%**	**78.86%**	86.41%
$LSVM_5$_km_	**78.93%**	**78.25%**	**79.65%**	**80.14%**	**77.72%**	**79.18%**	**87.14%**
$LSVM_5^{PSO}$_km_	78.81%	78.16%	79.49%	79.96%	77.66%	79.05%	86.55%
classical LSVM	79.52%	78.83%	80.24%	80.71%	78.32%	79.76%	87.59%
classical $LSVM^{PSO}$	**79.90%**	**79.02%**	**80.82%**	**81.40%**	**78.39%**	**80.19%**	**87.82%**
AmazonTest dataset							
LSVM_	87.59%	87.50%	87.68%	87.71%	87.47%	87.60%	95.04%
LSVM_km_	87.74%	87.75%	87.72%	87.71%	87.76%	87.73%	95.11%
$LSVM^{PSO}$_km_	**88.45%**	**88.57%**	**88.33%**	**88.29%**	**88.61%**	**88.43%**	**95.24%**
$LSVM_3$_km_	**88.90%**	**88.90%**	**88.89%**	**88.89%**	**88.90%**	**88.90%**	**95.78%**
$LSVM_3^{PSO}$_km_	88.88%	89.02%	88.74%	88.70%	89.06%	88.86%	95.49%
$LSVM_5$_km_	**89.29%**	**89.30%**	**89.27%**	**89.27%**	**89.30%**	**89.28%**	**95.93%**
$LSVM_5^{PSO}$_km_	89.03%	89.17%	88.89%	88.85%	89.21%	89.01%	95.56%
classical LSVM	89.58%	91.77%	87.60%	86.95%	92.20%	89.29%	96.33%
classical $LSVM^{PSO}$	**90.22%**	**90.20%**	**90.24%**	**90.24%**	**90.19%**	**90.22%**	**96.43%**

Underscore "_" means that 30K_SpeedUP should be added at the end.



(a) sentiment140 (b) AmazonTest

Fig. 4.25. Averaged effectiveness metrics of the full proposed hybrid method

Before applying Welch's t-test it was checked whether variables fol-

low a standard normal distribution (See Table B.16 in App. B).

Table 4.26. Accuracy comparison between the proposed hybrid method with different parts enabled/disabled using Welch's t-test

Method	LSVM_ LSVM^{PSO}_km_	LSVM$_3$_km_ LSVM$_3^{PSO}$_km_	LSVM$_5$_km_ LSVM$_5^{PSO}$_km_	LSVM LSVM^{PSO}
sentiment140 dataset				
Statistics	$\lvert\overline{d}\rvert = 1.01$ $SE = 0.021$ $\lvert T\rvert = 48.13$ $t_{table} = 2.145$	$\lvert\overline{d}\rvert = 0.11$ $SE = 0.031$ $\lvert T\rvert = 3.52$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.12$ $SE = 0.028$ $\lvert T\rvert = 4.34$ $t_{table} = 2.145$	$\lvert\overline{d}\rvert = 0.38$ $SE = 0.023$ $\lvert T\rvert = 16.63$ $t_{table} = 2.11$
p-value	$< .00001$	0.0026	0.0007	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	rejected rejected	rejected rejected	rejected rejected
AmazonTest dataset				
Statistics	$\lvert\overline{d}\rvert = 0.86$ $SE = 0.007$ $\lvert T\rvert = 126.19$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 0.02$ $SE = 0.015$ $\lvert T\rvert = 1.22$ $t_{table} = 2.12$	$\lvert\overline{d}\rvert = 0.25$ $SE = 0.011$ $\lvert T\rvert = 24.36$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.64$ $SE = 0.033$ $\lvert T\rvert = 19.50$ $t_{table} = 2.228$
p-value	$< .00001$	0.2402	$< .00001$	$< .00001$
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert > t_{table}$ $p < \alpha$
H_0	rejected rejected	not rejected not rejected	rejected rejected	rejected rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$). Underscore "_" means that 30K_SpeedUP should be added at the end.

The calculation was performed based on the accuracy of each CV fold achieved by relevant methods (See Table B.13 in Appendices B). The results in Table 4.26 showed that the biggest difference between accuracy average ($\overline{d}$) of the two methods was 1.01% on the sentiment140 and 0.86% on AmazonTest, when LSVM^{PSO}_km_30K_SpeedUP compared with LSVM_30K_ SpeedUP was used. In the case of LSVM and LSVM^{PSO} difference between accuracy average of the two methods increased by 0.38% on the sentiment140 and by 0.64% on AmazonTest. Considering Welch's t-test and p-value, the achieved results were significant in all cases (H_0 hypothesis

was rejected), except in the case between LSVM_3^{PSO}_km_30K_SpeedUP and LSVM_3_km_30K_SpeedUP where the H_0 hypothesis was not rejected (the hypotheses are defined in Subsection 3.3.3).

The results showed that ensembles from weaker classifiers (without PSO tuning applied) could give better results than stronger ones. Part of the results of this experiment cycle are published in [A2], [A3] and [A7]. They are slightly different because different CV folds were used.

Hybrid method for textual data sentiment classification

The aim of this dissertation was to propose a hybrid method for textual data sentiment classification with the recommend set of the parameters for it. Considering the achieved results, the recommended parameters for proposed method are presented in expression 4.1 and 4.2.

For standalone use:

$$\textbf{SpeedUP}(ml = \text{'LSVM'},\ kmeans = 1,\ ensemble = 0,\ pso = 1,\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2) \tag{4.1}$$

For using in ensemble:

$$\textbf{SpeedUP}(ml = \text{'LSVM'},\ kmeans = 1,\ ensemble = 5,\ pso = 0,\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2) \tag{4.2}$$

The main advantage of the proposed method that it is a synergy of the individual methods, which aim to combine their best properties, thereby compensating for the shortcomings of them. As presented in Table 4.27 the authors mostly used one or two methods, while the proposed method combine them all.

Further a comparison of the proposed method with other authors' work is made. The label for the proposed method will be as follows: LSVM^{PSO}_30K_SpeedUP – for standalone method, LSVM_3^{PSO}_30K_SpeedUP – for ensemble and LSVM^{PSO} – for classical LSVM with PSO tuning applied on it. It is assumed not to use k-Means clustering to show the effectiveness of the proposed method on large datasets, when only 70K instances for training are selected and all other data is used for testing.

Table 4.27. Advantage of proposed method compared to existing methods for research problem

Method	Training dataset reduction		Hyperparameter tuning	Ensemble
	k-Means	random selection		
Zheng et al. [131]	√			
Qian et al. [160]			√	
Zhang et al. [161]			√	
Mao et al. [127]		√		√
Pedregosa [150]			√	
Osman et al. [94]			√	
Catal and Nangir [170]			√	√
Gu et al. [159]			√	
Tang et al. [132]	√			
Sadhasivam and Kalivaradhan [171]				√
Alrehili and Albalawi [173]				√
Proposed method	√	√	√	√

4.1.5 Experiment cycle of the comparison of the results

4.1.5.1 Experimental settings

For the comparison of the proposed method with other authors' research, datasets and their settings presented in Table 4.28. For this cycle parameters in the SpeedUP method are set as follows:

For the first case LSVM^{PSO}_30K_SpeedUP will be used:

$$\textbf{SpeedUP}(ml = \text{'LSVM'},\ kmeans = 0,\ ensemble = 0,\ pso = 1,\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

For the second case LSVM$_3^{PSO}$_30K_SpeedUP will be used:

$$\textbf{SpeedUP}(ml = \text{'LSVM'},\ kmeans = 0,\ ensemble = 3,\ pso = 1,\ Subset_{size} = 30\text{K},\ D_{text},\ num_{class} = 2)$$

It was decided to additionally perform PSO tuning when it is applied on classical LSVM. It will be labeled LSVM^{PSO}. The goal is to check the effectiveness of PSO tuning. Table 4.29 shows the sizes of training and testing datasets for LSVM^{PSO} with split ratio 70/30.

Although the Amazon reviews come in 5-star rating, the aforementioned authors used only two classes of presented datasets. According to them, 3-star ratings are considered as neutral reviews (meaning neither positive nor negative), hence instances with this class were discarded from

datasets. The remaining classes were converted into binary as follows: reviews with 1 or 2 star rating were labeled as '0', whereas reviews with 4 and 5 stars received a label '1'.

Table 4.28. Training and testing data sizes of the experiment cycle of the comparison of the results

Dataset	Testing data size (TDs)	Subset size (Ss)	Subsets quantity (SQ) $trunc(TDs/Ss)$	Remainder $TDs-(Ss*SQ)$	Training data depending on Ss $Ss*Split_{ratio}$
Books	20,037,414	30,000	667	27,414	70,000
Electronics	7,117,716	30,000	237	7,716	70,000
KindleStore	2,828,300	30,000	94	8,300	70,000
Phones&Accessories	3,025,090	30,000	100	25,090	70,000

To show the effectiveness of the proposed method on large datasets it was decided not to use k-Means clustering for training data selection; instead of it training data was randomly selected from the whole dataset and the remaining part was used for testing. All experiments were performed ten times to get more accurate results and the average was taken as the final result.

Table 4.29. Training and testing data sizes for the comparison when a classical LSVM with PSO tuning is used

Dataset	Training data (70%)	Testing data (30%)
Electronics	5,031,400	2,156,316
Cell Phones and Accessories	2,166,563	928,527

4.1.5.2 Results

In this cycle comparison of our proposed method with other authors' work was performed. It is difficult to explicitly compare the results obtained with results in other papers due to discrepancy in implementations, parameters or tasks. Therefore, Table 4.30 presents the results of comparative analysis based on accuracy, when the proposed method was applied to the same datasets and contains the same number of classes. The results demonstrated that sufficient accuracy can be obtained using a smaller training subset.

Table 4.30. Comparison of the results with other authors' research

Authors	Dataset	ML method	Accuracy
Rain C. [210] (2013)	Books	NB	84.50%
Shaikh and Deshpande [211] (2016)		NB	80.00%
Proposed method		LSVM^{PSO}_30K_SpeedUP	**89.50%**
		LSVM$_3^{PSO}$_30K_SpeedUP	**89.86%**
Rain C. [210] (2013)	KindleStore	NB	87.33%
Proposed method		LSVM^{PSO}_30K_SpeedUP	**91.27%**
		LSVM$_3^{PSO}$_30K_SpeedUP	**91.50%**
Haque et al. [212] (2018)	Phones&	LSVM	**93.57%**
	Accessories	MNB	90.28%
		SGD	91.88%
		RF	92.72%
		LR	88.20%
		DT	91.45%
Wang et al. [88] (2018)		CNN	85.9%
		CFM	83.5%
		PFM	84.2%
		LR-BoW	83.8%
		SVM-BoW	83.7%
		SVM-Poly	82.30%
		LR-WE	76.7%
		SVM-WE	76.7%
		CNN	75.4%
		FM	78.6%
Proposed method		LSVM^{PSO}_30K_SpeedUP	90.57%
		LSVM$_3^{PSO}$_30K_SpeedUP	90.83%
		LSVM^{PSO}	93.22%
Haque et al. [212] (2018)	Electronics	LSVM	**93.52%**
		MNB	89.36%
		SGD	92.61%
		RF	92.89%
		LR	88.96%
		DT	91.57%
Proposed method		LSVM^{PSO}_30K_SpeedUP	90.14%
		LSVM$_3^{PSO}$_30K_SpeedUP	90.52%
		LSVM^{PSO}	93.17%

Table 4.30 showed that LSVM^{PSO}_30K_SpeedUP and LSVM$_3^{PSO}$_30K_SpeedUP resulted in higher accuracy compared with [210] and [211], when applied on the largest Books dataset and KindleStore dataset [210].

The proposed method and its ensembles also outperformed CNN, CFM and PFM, when they were applied on Phones&Accessories dataset compared with [88]. However, it performed slightly worse when compared with [212], where the proposed method was slightly surpassed by LSVM on the Electronics and Phones&Accessories datasets. The results showed that the proposed method proved to be competitive when compared with the research of other authors. Part of the results of this experiment cycle are published in [A3].

4.1.6 Experiment cycle with real-world data

4.1.6.1 Experimental settings

For this cycle the data was collected from the Internet. These two topics were selected: "A public person" (Person) and "The event" (Event). The questions for public opinion research were: "How do you appreciate this person: positively or negatively?" and "Do you agree with this event: Yes or No?". Both topics were related to Lithuania, so the comments should also be selected in Lithuanian. There are a lot of tools created for downloading data, but because the Lithuanian language was selected for this research, web scrapping (it is used to collect information from websites) tool was written by using Python programming language. The time range of both collected datasets is from 01.01.2019 to 01.04.2019. Social networks and social media served as sources.

Experiments with LSVM^{PSO}_s_SpeedUP are performed here. It means that parameters in the SpeedUP method are set as follows:

$$\textbf{SpeedUP}(ml = \text{'LSVM'},\ kmeans = 0,\ ensemble = 0,\ pso = 1,$$
$$Subset_{size} = \{287,\ 300\},\ D_{text},\ num_{class} = 2)$$

where $Subset_{size} = 287$, when Person dataset is used and $Subset_{size} = 300$, when in the case of Event.

The aim is to present practical usage of the method proposed in this dissertation. Figure 4.26 presents the diagram of the experiment cycle with real-world data. The dashed line in the diagram represents steps for additional calculations executed before a concrete step is joined to it.

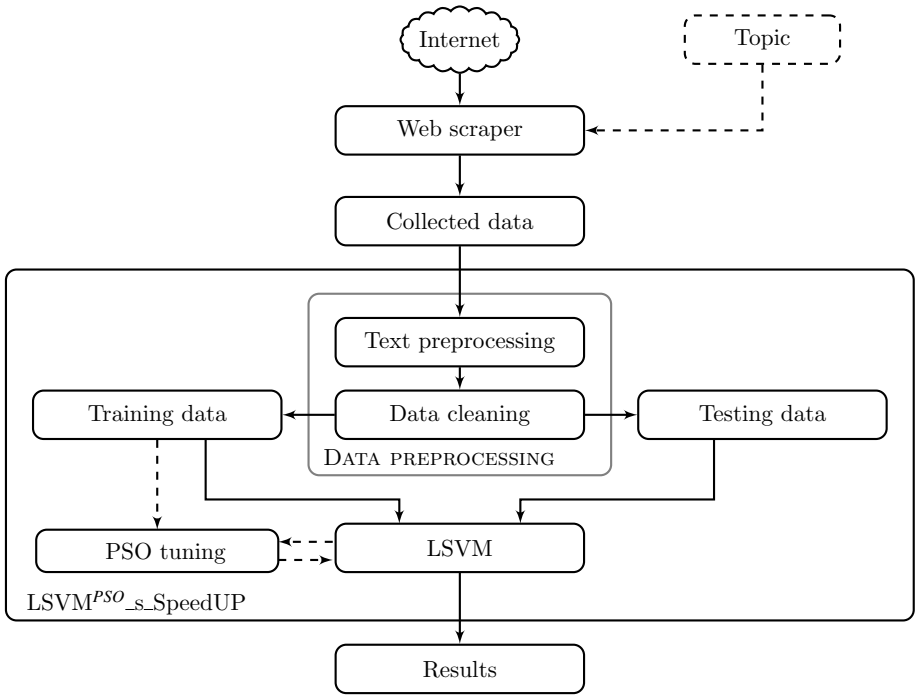


Fig. 4.26. Diagram of the experiment cycle with real-world data

The diagram consists of the following steps:

- The diagram starts with collecting data. Data was collected according to the selected topics.
- Data preprocessing is performed on collected data. This step contains two actions: text preprocessing and data cleaning. Text preprocessing includes actions such as converting to lowercase, removing redundant tokens such as hashtag, symbols @, numbers, "http" for links, punctuation symbols, usernames etc. Later, in the data cleaning part, the dataset is checked for empty strings and if any exist they are removed.
- After that data is split into training and testing data. In particular, data is split into ten training and testing data folds by using Stratified ShuffleSplit cross-validator (StratifiedShuffleSplit) from scikit-learn library for more objective results. The most common split ratio (70/30) into training and testing data is used. The description of datasets is shown in table 4.31, while sizes of training and testing data are presented in Table 4.32.
- Training and testing data are passed to LSVM^{PSO}_s_SpeedUP.
- PSO tuning performs selecting the C parameter for LSVM.

- Later LSVM executes and provides sentiment classification results. However, the results are provided from 10 CV folds, and they are averaged as final results.

Table 4.31. The description of datasets

Dataset	Num. of comments	Positive comment	Negative comment	Num. of classes
Person	954	304	650	2
Event	1000	346	654	2

The data for both datasets is in a CSV file format. Comments are in Lithuanian. Data file format has 2 fields:

0 – the polarity of a comment (0 = negative, 1 = positive)
1 – the text of the comment

It is important to mention that all data was labeled manually by the author, therefore the datasets contain this amount of data.

Table 4.32. Training and testing data sizes

Dataset	Training data (70%)	Testing data (30%)
Person	667	287
Event	700	300

Person dataset contains 954 sentiments in total, which consists of two classes with 304 positive sentiments in the first class and 650 negative sentiments in the other; Event is composed of 1,000 sentiments in total, which contains 2 classes with 346 positive sentiments in one and 654 negative in the other.

The aim is to compare the results achieved by LSVM^{PSO}_s_SpeedUP with classical LSVM, classical LSVM with random search (LSVM$_{RS}$) and Bayesian optimization (LSVM$_{Bopt}$) hyperparameter tuning methods integrated into it, and the real results presented by two institutions of public opinion and market research: Vilmorus ltd. (Source1) and Baltic Surveys (Source2).

4.1.6.2 Results

The aim of this cycle of experiments was to present practical usage of the hybrid method proposed in this dissertation. For comparison classical LSVM, LSVM$_{RS}$, LSVM$_{Bopt}$ and LSVM^{PSO}_s_SpeedUP were used. Table 4.33 shows the results of the aforementioned methods.

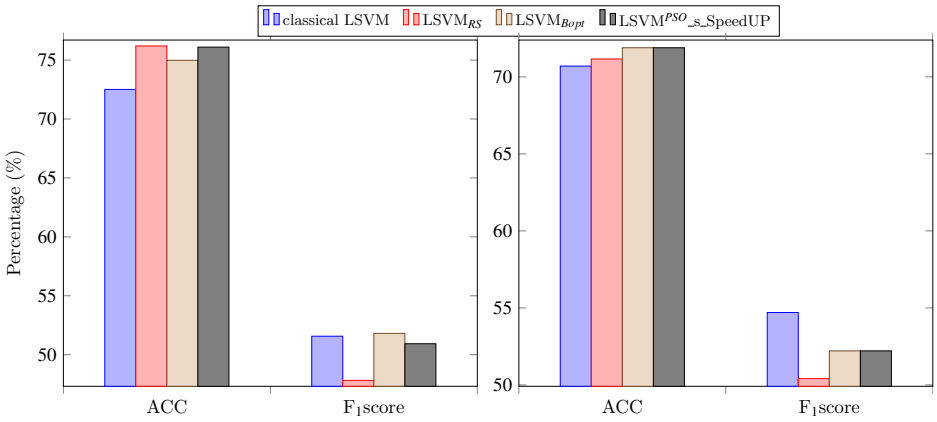
The results showed that LSVM$_{RS}$ outperformed classical LSVM in terms of ACC, PPV and TNR on Person dataset, while there was a slight loss in terms of NPV, TPR and F_1score. LSVM$_{RS}$ also slightly exceeded LSVM^{PSO}_287_SpeedUP, while difference in terms of ACC was only 0.1%; in the case of LSVM$_{Bopt}$ it was 1.22%. The quality measure of the model's predictions AUC also showed the superiority of the LSVM$_{RS}$ on Person dataset. However, LSVM$_{RS}$ lost to LSVM^{PSO}_300_SpeedUP and LSVM$_{Bopt}$ on Event dataset, where the data contained more ambiguity. The difference in terms of ACC was 0.73% in the case of LSVM^{PSO}_300_SpeedUP and LSVM$_{Bopt}$. The results showed that LSVM^{PSO}_s_SpeedUP also outperformed classical LSVM on both datasets with 3.59% on Person and 1.19% on Event. The quality measure of the model's predictions AUC showed the superiority of the LSVM$_{Bopt}$.

Table 4.33. Averaged effectiveness metrics of LSVM^{PSO}_s_SpeedUP, LSVM$_{RS}$, LSVM$_{Bopt}$ and classical LSVM in the experiment cycle with real-world data

Method	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
Person dataset							
classical LSVM	72.51%	59.30%	76.90%	**45.76%**	85.13%	51.57%	74.43%
LSVM$_{RS}$	**76.20%**	**81.00%**	75.57%	34.13%	**96.05%**	47.82%	**76.21%**
LSVM$_{Bopt}$	74.98%	69.82%	**77.05%**	42.72%	90.21%	**51.81%**	75.71%
LSVM^{PSO}_287_	76.10%	76.68%	76.39%	38.59%	93.79%	50.93%	75.91%
Event dataset							
classical LSVM	70.70%	58.83%	**76.04%**	**51.54%**	80.81%	**54.70%**	73.97%
LSVM$_{RS}$	71.16%	62.53%	74.39%	43.56%	85.74%	50.41%	74.41%
LSVM$_{Bopt}$	**71.89%**	63.21%	75.26%	46.15%	85.48%	52.55%	**74.55%**
LSVM^{PSO}_300_	**71.89%**	**63.78%**	75.06%	45.38%	**85.89%**	52.21%	74.49%

Underscore "_" means that SpeedUP should be added at the end.

The distribution of the results of the accuracy of LSVM^{PSO}_s_SpeedUP, LSVM$_{RS}$, LSVM$_{Bopt}$ and classical LSVM methods is depicted in Figure 4.27.



(a) Person dataset (b) Event dataset

Fig. 4.27. Accuracy of LSVM^{PSO}_s_SpeedUP, LSVM$_{RS}$, LSVM$_{Bopt}$ and classical LSVM in the experiment cycle with real-world data

Depending on the results, the accuracy of Event dataset was smaller than Person. This happened because the second dataset contained more ambiguity. For example, in the case of Person dataset, when it was checking the appreciation of a person, the answer was very simple: positive or negative, while on Event dataset an opinion was positive, when it was related directly to this event e.g. "yes, this event should happen", "I'm waiting for it" etc.; if an opinion was not related directly to the first event e.g. "the other proposed event is much better", "other event is fantastic" etc. it was negative.

Before applying Welch's t-test it was checked whether variables follow a standard normal distribution (See Table B.16 in App. B). The results presented in Table 4.34 were calculated based on the accuracy of each CV fold achieved by relevant methods (See Table B.15 in App. B). They showed that the biggest difference between accuracy average ($\overline{d}$) of the two methods LSVM^{PSO}_s_SpeedUP and classical LSVM was 3.59%, on Person dataset. The achieved results were statistically significant, because the H_0 hypothesis was rejected (the hypotheses are defined in Subsection 3.3.3). The comparison of the results between LSVM^{PSO}_s_SpeedUP and LSVM$_{RS}$, and also between LSVM^{PSO}_s_SpeedUP and LSVM$_{Bopt}$ were not significant, because the H_0 hypothesis was not rejected. This led to the conclusion that the proposed method could be competitive with such methods for tuning hyperparameters as random search and Bayesian optimization.

Table 4.34. Accuracy comparison between LSVM^{PSO}_s_SpeedUP, LSVM$_{RS}$, LSVM$_{Bopt}$ and classical LSVM in the experiment cycle with real-world data using Welch's t-test

Method	LSVM LSVM^{PSO}_	LSVM$_{Bopt}$ LSVM^{PSO}_	LSVM$_{RS}$ LSVM^{PSO}_	LSVM LSVM^{PSO}_	LSVM$_{RS}$ LSVM^{PSO}_
	A public person			The event	
Statistics	$\lvert\overline{d}\rvert = 3.59$ $SE = 1.135$ $\lvert T\rvert = 3.16$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 1.12$ $SE = 1.162$ $\lvert T\rvert = 0.96$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.10$ $SE = 0.925$ $\lvert T\rvert = 0.11$ $t_{table} = 2.131$	$\lvert\overline{d}\rvert = 1.20$ $SE = 1.038$ $\lvert T\rvert = 1.15$ $t_{table} = 2.11$	$\lvert\overline{d}\rvert = 0.73$ $SE = 1.025$ $\lvert T\rvert = 0.71$ $t_{table} = 2.11$
p-value	0.0057	0.3508	0.9112	0.2654	0.4854
Results	$\lvert T\rvert > t_{table}$ $p < \alpha$	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert < t_{table}$ $p > \alpha$	$\lvert T\rvert < t_{table}$ $p > \alpha$
H_0	rejected rejected	not rejected not rejected	not rejected not rejected	not rejected not rejected	not rejected not rejected

$\overline{d}$ – mean accuracy between the two methods, SE – standard error of difference, T – Welch's test statistic, t_{table} – value from t-distribution table, α – significance level ($\alpha = 0.05$). Underscore "_" means that s_SpeedUP should be added at the end.

Table 4.35. Result comparison between the original sources, manually labeled data and ML methods

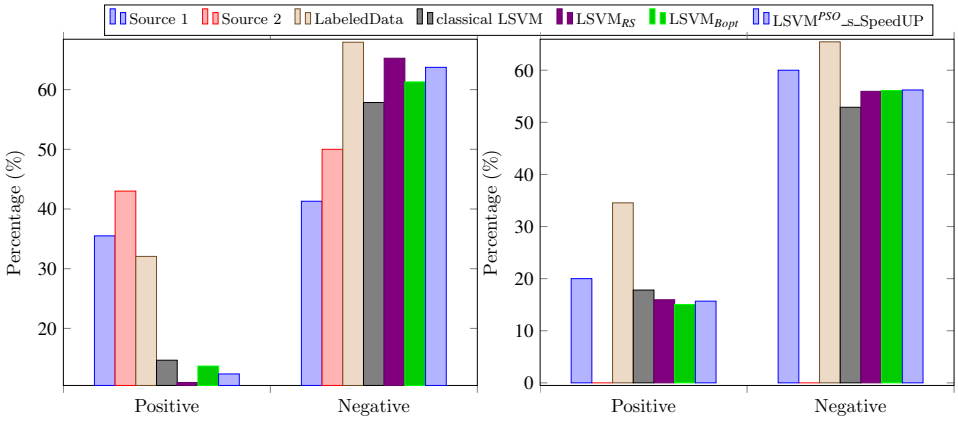
Dataset	Person		Event	
	positive	negative	positive	negative
The results presented by Vilmorus ltd. and Baltic Surveys				
Source1	35.5%	41.3%	20%	60%
Source2	43%	50%	–	–
The results obtained from manually selected and labeled data				
LabeledData	32.06%	67.94%	34.55%	65.45%
The results obtained by ML methods				
classical LSVM	**14.67%**	57.84%	**17.81%**	52.89%
LSVM$_{RS}$	10.94%	**65.26%**	15.04%	56.11%
LSVM$_{Bopt}$	13.69%	61.29%	15.95%	55.95%
LSVM^{PSO}_s_SpeedUP	12.37%	63.73%	15.68%	**56.21%**

Finally, the results of the aforementioned methods were expressed in the percentage of positive and negative opinions. Table 4.35 shows the results which were presented by Source1, Source2, LabeledData (opinions

which were collected from the Internet, manually labeled and the percentage expression of positive and negative opinions was calculated), classical LSVM, LSVM$_{RS}$, LSVM$_{Bopt}$ and LSVM^{PSO}_s_SpeedUP.

LabeledData on Person dataset is not very different from Source1 and Source2, especially in the case of positive opinions. The difference in the case of negative opinions could be explained by missing data on Source1 and Source2, when participants in the surveys did not have any opinion. Classical LSVM outperformed all methods on Person in the case of positive opinions; LSVM^{PSO}_s_SpeedUP exceeded LSVM$_{RS}$, while it lost to LSVM$_{Bopt}$; in the case of negative opinions the results of LSVM$_{RS}$ were best, while LSVM^{PSO}_s_SpeedUP outperformed LSVM$_{Bopt}$. The results of classical LSVM were best on Event dataset in the case of positive opinions, while the results of LSVM^{PSO}_s_SpeedUP – in the case of negative opinions.

The distribution of the results is depicted in Figure 4.28.



(a) Person (b) Event

Fig. 4.28. Percentage results of "positive" and "negative" opinions obtained by the sources and ML methods in the experiment cycle with real-world data

4.2 Conclusions of Chapter 4

In this chapter, experiments and experimental settings for experiments with the proposed hybrid method for textual data sentiment analysis are described. The experiments were performed by turning on/off settings of the proposed method. The classical algorithms – MNB, LR, LSVM, DT and RF – were compared and baselines for the hybrid method were set according to

the obtained results. The experimental results showed the following:

1. The comparison between classical MNB, LR, LSVM, RF and DT algorithms showed that the best execution time was achieved by MNB on both experimental datasets – 3.86s on sentiment140 and 12.02s on AmazonTest datasets, while LR performed better in terms of ACC (79.96% and 90.20%), NPV (80.63% and 90.33%), TPR (81.06% and 90.36%), F_1score (80.17% and 90.21%) and AUC (87.84% and 96.38%) on both datasets; however, there was a slight loss in the case of PPV (79.44%) and TNR (81.71%) to RF on the sentiment140 and to LSVM in terms of PPV (91.77%) and TNR (92.20%) on AmazonTest. Average ranking showed the superiority of LR and LSVM compared with others classifiers.
2. ML_30K_SpeedUP showed the reduced execution time in the case of LR (up to 4.7x), LSVM (up to 50.7x), RF (up to 72.3x) and especially of DT (up to 634.8x). In the case of MNB it resulted in increased execution time (up to 5.6x) compared with the classical algorithms. However, all classifiers reported slightly lower accuracy. ML_km_30K_SpeedUP achieved higher accuracy compared with ML_30K_SpeedUP: in the case of MNB (0.07%-0.19%), LR (0.05%-0.09%), LSVM (0.15%-0.20%), RF (0.43%-1.16%), DT (0.36%-1.31%). In the case of ensembles, classifiers showed significantly higher results compared to standalone methods: ML_3_km_30K_SpeedUP achieved the higher accuracy in the case of MNB (0.39%-0.74%), LR (0.27%-0.54%), LSVM (1.16%-1.21%), RF (1.80%-3.13%), DT (3.37%-4.11%); ML_5_km_30K_SpeedUP in the case of MNB (0.44%-0.96%), LR (0.36%-0.70%), LSVM (1.55%-1.63%), RF (2.53%-4.08%), DT (4.94%-5.70%). The results also showed that MNB_3_km_30K_SpeedUP outperformed classical MNB on AmazonTest dataset by 0.17%; RF_5_km_30K_SpeedUP outperformed classical RF on both datasets (0.69%-2.16%), while DT_5_km_30K_SpeedUP methods exceeded classical DT (2.00%-2.97%). However, they resulted in less accuracy compared with LR and LSVM in all cases. The average ranking showed the superiority of LR and LSVM in all experiments. PSO_tuning – ML^{PSO}_km_30K_SpeedUP – in the case of the LSVM showed the significantly higher results compared with LSVM_km_30K_SpeedUP (0.71%-0.82%), while in the case of LR it achieved slightly lower results (0.03%). However, in the case of ensembles $LSVM_3^{PSO}$_km_30K_SpeedUP outperformed $LSVM_3$_km_30K_

SpeedUP (0.11%) on the sentiment140, while it lost on AmazonTest (0.02%); in the case of $LSVM_5^{PSO}$_km_30K_SpeedUP it lost to $LSVM_5$_km_30K_SpeedUP (0.12%-0.26%) on both datasets. The average ranking showed the superiority of LSVM compared with LR.

3. According to ranks, LR and LSVM were the best classifiers, but LSVM was the best choice for the proposed hybrid method, because it fit better. The accuracy of LSVM_30K_SpeedUP by applying the full proposed method increased 1.70%-1.83%, while in the case of LR_30K_SpeedUP only by 0.41%-0.79%. Two sets of parameters were recommended for the proposed method. The recommendation is to keep PSO_tuning turned off in the case of ensembles and turned on, when standalone method is used. The comparison with other work showed that the proposed method could be competitive compared with classical (LSVM, LR, MNB, RF, DT) and state-of-the-art (CNN, CFM, PFM) methods which are used by other authors.
4. The experimental results performed on real-world data with the hybrid method showed that $LSVM^{PSO}$_s_SpeedUP outperformed classical LSVM in terms of accuracy by 1.19%-3.59% on both datasets; LSVM with Bayesian optimization for tuning of parameters by 1.12% on Person dataset, while on Event they performed with equal accuracy of 71.89%. LSVM with random search for tuning of parameters performed better on Person – by 0.10%, while it lost on Event dataset by 0.73%. Statistically this difference was not significant, so it led to the conclusion that the proposed hybrid method with PSO tuning could be competitive with such popular methods as RS and Bopt.

GENERAL CONCLUSIONS

1. The proposed hybrid method for textual data sentiment analysis in large scale datasets can increase classification speed up to 4.7x-634.8x of classical machine learning algorithms such as LSVM, LR, DT and RF, while losing in terms of accuracy is 0.29%-4.06%. The method includes the following:
    - **SpeedUP** – the main part of the proposed hybrid method, whose aim is to increase the classification speed of classical machine learning algorithms. The speed increased: LSVM (up to 50.7x), LR (up to 4.7x), DT (up to 634.8x) and RF (up to 72.3x). Accuracy loses compared with classical ML algorithms were: LSVM (1.99%-2.42%), LR (1.91%-1.97%), DT (3.28%-4.06%) and RF (2.35%-3.00%).
    - **k-Means clustering** – responsible for training data selection. Accuracy loses compared with classical ML algorithms were: LSVM (1.84%-2.22%), LR (1.82%-1.92%), DT (1.97%-3.70%) and RF (1.84%-1.92%).
    - **PSO tuning** – performs the tuning of hyperparameters. Accuracy loses compared with classical LSVM were 1.13%-1.40%.
    - **Ensemble** – performs combination and voting of the machine learning algorithms. Accuracy loses compared with classical ML algorithms were: LSVM (0.29%-0.59%), LR (1.12%-1.56%), DT (outperformed classical DT by 2.0%-2.97%) and RF (outperformed classical RF by 0.69%-2.16%).
2. Two recommended sets of parameters for the hybrid method were proposed according to the experimental results: the first for the standalone classifier and the second for ensemble. The best results were achieved, when LSVM was used with the first set (accuracy increased by 0.96%-1.02%) and the second set (accuracy increased by 1.70%-1.83%) of parameters. It is recommended to turn off PSO tuning if the ensemble method is enabled, since ensemble works better with weaker classifiers – it outperformed ensemble with PSO tuning by 0.02%-0.26%.
3. According to the results achieved during the comparison with other authors' work, the proposed hybrid method could be used as an alter-

native method on large scale textual datasets for classical (LSVM, LR, MNB, RF, DT) (the proposed method lost by 0.35%-3.00% and outperformed it by 0.28%-9.86%) and state-of-the-art methods (CNN, CFM, PFM) (the proposed method outperformed by 6.77%-17.82%), which are used by other authors on not very powerful computers. Moreover, PSO tuning applied on LSVM could be an alternative to such popular methods as random search (the proposed method lost by 0.10% and outperformed by 0.73%) and Bayesian optimization (the proposed method outperformed by 1.12%) on the real-world data classification tasks in public opinion research.

REFERENCES

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[2] S. Das and M. Chen, "Yahoo! for Amazon: Extracting market sentiment from stock message boards," in *Proceedings of the Asia Pacific finance association annual conference (APFA)*, vol. 35. Bangkok, Thailand, 2001, p. 43.

[3] R. M. Tong, "An operational system for detecting and tracking opinions in on-line discussion," in *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*, vol. 1, no. 6, 2001.

[4] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 417–424.

[5] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 79–86.

[6] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 519–528.

[7] B. Liu, *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, 2015.

[8] D. Maynard and A. Funk, "Automatic detection of political opinions in tweets," in *Extended Semantic Web Conference*. Springer, 2011, pp. 88–99.

[9] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[10] A. Giachanou and F. Crestani, "Like it or not: A survey of twitter sentiment analysis methods," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–41, 2016.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[12] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[13] J. Khairnar and M. Kinikar, "Machine learning algorithms for opinion mining and sentiment classification," *International Journal of Scientific and Research Publications*, vol. 3, no. 6, pp. 1–6, 2013.

[14] H. Tang, S. Tan, and X. Cheng, "A survey on sentiment detection of reviews," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 760–10 773, 2009.

[15] E. Cambria, B. Schuller, Y. Xia, and C. Havasi, "New avenues in opinion mining and sentiment analysis," *IEEE Intelligent systems*, vol. 28, no. 2, pp. 15–21, 2013.

[16] A. Yousif, Z. Niu, J. K. Tarus, and A. Ahmad, "A survey on sentiment analysis of scientific citations," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1805–1838, 2019.

[17] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[18] A. Qazi, R. G. Raj, G. Hardaker, and C. Standing, "A systematic literature review on opinion types and sentiment analysis techniques," *Internet Research*, 2017.

[19] C. Dhaoui, C. M. Webster, and L. P. Tan, "Social media sentiment analysis: lexicon versus machine learning," *Journal of Consumer Marketing*, 2017.

[20] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, 2009.

[21] V. Kharde, P. Sonawane *et al.*, "Sentiment analysis of twitter data: a survey of techniques," *arXiv preprint arXiv:1601.06971*, 2016.

[22] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys," in *Coling 2010: Posters*, 2010, pp. 241–249.

[23] J. Kapočiūtė-Dzikienė, A. Krupavičius, and T. Krilavičius, "A comparison of approaches for sentiment classification on lithuanian internet comments," in *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, 2013, pp. 2–11.

[24] B. Le and H. Nguyen, "Twitter sentiment analysis using machine learning techniques," in *Advanced Computational Methods for Knowledge Engineering*. Springer, 2015, pp. 279–289.

[25] G. Gautam and D. Yadav, "Sentiment analysis of twitter data using machine learning approaches and semantic analysis," in *Contemporary computing (IC3), 2014 seventh international conference on*. IEEE, 2014, pp. 437–442.

[26] O. Kolchyna, T. T. Souza, P. Treleaven, and T. Aste, "Twitter sentiment analysis: Lexicon method, machine learning method and their combination," *arXiv preprint arXiv:1507.00955*, 2015.

[27] M. Kanakaraj and R. M. R. Guddeti, "NLP based sentiment analysis on Twitter data using ensemble classifiers," in *Signal processing, communication and networking (ICSCN), 2015 3rd international conference on*. IEEE, 2015, pp. 1–5.

[28] Y. Wan and Q. Gao, "An ensemble sentiment classification system of twitter data for airline services analysis," in *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1318–1325.

[29] A. Amolik, N. Jivane, M. Bhandari, and M. Venkatesan, "Twitter sentiment analysis of movie reviews using machine learning techniques,"

International Journal of Engineering and Technology, vol. 7, no. 6, pp. 1–7, 2016.

[30] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems with Applications*, vol. 57, pp. 117–126, 2016.

[31] S. Bhuta, A. Doshi, U. Doshi, and M. Narvekar, "A review of techniques for sentiment analysis Of Twitter data," in *2014 International conference on issues and challenges in intelligent computing techniques (ICICT)*. IEEE, 2014, pp. 583–591.

[32] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[33] M. Ahmad, S. Aftab, S. S. Muhammad, and S. Ahmad, "Machine Learning Techniques for Sentiment Analysis: A Review," *Int. J. Multidiscip. Sci. Eng*, vol. 8, no. 3, pp. 27–32, 2017.

[34] T. Pranckevičius and V. Marcinkevičius, "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," *Baltic Journal of Modern Computing*, vol. 5, no. 2, p. 221, 2017.

[35] A. S. Rathor, A. Agarwal, and P. Dimri, "Comparative Study of Machine Learning Approaches for Amazon Reviews," *Procedia Computer Science*, vol. 132, pp. 1552–1561, 2018.

[36] R. Manikandan and R. Sivakumar, "Machine learning algorithms for text-documents classification: A review," *Machine learning*, vol. 3, no. 2, 2018.

[37] R. V. A. Ogutu, R. Rimiru, and C. Otieno, "Target Sentiment Analysis Model with Naïve Bayes and Support Vector Machine for Product Review Classification," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 7, 2019.

[38] I. A. Kandhro, M. A. Chhajro, K. Kumar, H. N. Lashari, and U. Khan, "Student Feedback Sentiment Analysis Model using Various Machine Learning Schemes: A Review," *Indian Journal of Science and Technology*, vol. 12, p. 14, 2019.

[39] J. Kapočiūtė-Dzikienė, R. Damaševičius, and M. Woźniak, "Sentiment analysis of lithuanian texts using traditional and deep learning approaches," *Computers*, vol. 8, no. 1, p. 4, 2019.

[40] R. B. Shamantha, S. M. Shetty, and P. Rai, "Sentiment Analysis Using Machine Learning Classifiers: Evaluation of Performance," in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2019, pp. 21–25.

[41] A. Gupta, A. Singh, I. Pandita, and H. Parashar, "Sentiment Analysis of Twitter Posts using Machine Learning Algorithms," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2019, pp. 980–983.

[42] S. Kumar, M. Gahalawat, P. P. Roy, D. P. Dogra, and B.-G. Kim, "Exploring Impact of Age and Gender on Sentiment Analysis Using Machine Learning," *Electronics*, vol. 9, no. 2, p. 374, 2020.

[43] N. C. Dang, M. N. Moreno-García, and F. De la Prieta, "Sentiment Analysis Based on Deep Learning: A Comparative Study," *Electronics*, vol. 9, no. 3, p. 483, 2020.

[44] A. Kumar and A. Jaiswal, "Systematic literature review of sentiment analysis on Twitter using soft computing techniques," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 1, p. e5107, 2020.

[45] S. Huang, L. Bao, Y. Cao, Z. Chen, C.-Y. Lin, C. R. Ponath, J.-T. Sun, M. Zhou, and J. Wang, "Smart Sentiment Classifier for Product Reviews," Oct. 9 2008, US Patent App. 11/950,512.

[46] C. Xia, "Social media sentiment analysis method adopting machine learning," 2017, Patent App. CN106776982 (A).

[47] S. Li, X. Zhang, and G. Zhou, "Text sentiment classification method and system," 2012, Patent App. CN102682130 (A).

[48] Z. Wang, J. C. V. TONG, and S.-B. HO, "Method And System Of Intelligent Sentiment And Emotion Sensing With Adaptive Learning," 2018, Patent App. WO 2018/182501 Al.

[49] M. Chatterjee, R. Turan, B. Lue, A. Agrawal, and K. Perillo, "Hybrid human machine learning system and method," Oct. 18 2016, US Patent 9,471,883.

[50] R. R. Nair, J. Mathew, V. Muraleedharan, and S. D. Kanmani, "Study of Machine Learning Techniques for Sentiment Analysis," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2019, pp. 978–984.

[51] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 616–623.

[52] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2004, pp. 488–499.

[53] J. Su, J. S. Shirab, and S. Matwin, "Large scale text classification using semi-supervised multinomial naive bayes," in *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 97–104.

[54] L. Jiang, S. Wang, C. Li, and L. Zhang, "Structure extended multinomial naive Bayes," *Information Sciences*, vol. 329, pp. 346–356, 2016.

[55] M. Mowafy, A. Rezk, and H. El-Bakry, "An efficient classification model for unstructured text document," *American Journal of Computer Science and Information Technology*, vol. 6, no. 1, p. 16, 2018.

[56] M. Abbas, K. A. Memon, A. A. Jamali, S. Memon, and A. Ahmed, "Multinomial Naive Bayes Classification Model for Sentiment Analysis," *IJCSNS*, vol. 19, no. 3, p. 62, 2019.

[57] J. Chen, Z. Dai, J. Duan, H. Matzinger, and I. Popescu, "Improved Naive Bayes with optimal correlation factor for text classification," *SN Applied Sciences*, vol. 1, no. 9, p. 1129, 2019.

[58] S. Ruan, H. Li, C. Li, and K. Song, "Class-Specific Deep Feature Weighting for Naïve Bayes Text Classifiers," *IEEE Access*, vol. 8, pp. 20 151–20 159, 2020.

[59] V. Balakrishnan and W. Kaur, "String-based Multinomial Naïve Bayes for Emotion Detection among Facebook Diabetes Community," *Procedia Computer Science*, vol. 159, pp. 30–37, 2019.

[60] A. A. Farisi, Y. Sibaroni, and S. Al Faraby, "Sentiment analysis on hotel reviews using Multinomial Naïve Bayes classifier," in *Journal of Physics: Conference Series*, vol. 1192, no. 1. IOP Publishing, 2019, pp. 12–24.

[61] H. Zhang, "The Optimality of Naive Bayes," *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, vol. 2, 2004.

[62] H. Hamdan, P. Bellot, and F. Bechet, "Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis," in *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, 2015, pp. 753–758.

[63] E. Byrne and P. Schniter, "Sparse multinomial logistic regression via approximate message passing," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5485–5498, 2016.

[64] R. Rafeek and R. Remya, "Detecting contextual word polarity using aspect based sentiment analysis and logistic regression," in *2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*. IEEE, 2017, pp. 102–107.

[65] H. Wang, R. Zhu, and P. Ma, "Optimal subsampling for large sample logistic regression," *Journal of the American Statistical Association*, vol. 113, no. 522, pp. 829–844, 2018.

[66] H. Zhang, Z. Li, H. Shahriar, L. Tao, P. Bhattacharya, and Y. Qian, "Improving Prediction Accuracy for Logistic Regression on Imbalanced Datasets," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 918–919.

[67] C. Zhu, C. U. Idemudia, and W. Feng, "Improved logistic regression model for diabetes prediction by integrating PCA and K-means techniques," *Informatics in Medicine Unlocked*, vol. 17, p. 100179, 2019.

[68] X. Mai, Z. Liao, and R. Couillet, "A large scale analysis of logistic regression: Asymptotic performance and new insights," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3357–3361.

[69] C.-X. Zhang, S. Xu, and J.-S. Zhang, "A novel variational Bayesian method for variable selection in logistic regression models," *Computational Statistics & Data Analysis*, vol. 133, pp. 1–19, 2019.

[70] M. Okabe, J. Tsuchida, and H. Yadohisa, "F-measure Maximizing Logistic Regression," *arXiv preprint arXiv:1905.02535*, 2019.

[71] C. Robert, "Machine learning, a probabilistic perspective," 2014.

[72] H. Abdi and N. J. Salkind, "Encyclopedia of measurement and statistics," *Thousand Oaks, CA: Sage. Agresti, A.(1990) Categorical data analysis., New York: Wiley. Agresti, A.(1992) A survey of exact inference for contingency tables. Statist Sci*, vol. 7, pp. 131–153, 2007.

[73] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The journal of educational research*, vol. 96, no. 1, pp. 3–14, 2002.

[74] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[75] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[76] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 2000.

[77] R. Damaševičius, "Optimization of SVM parameters for recognition of regulatory DNA sequences," *Top*, vol. 18, no. 2, pp. 339–353, 2010.

[78] I. Martišius, R. Damaševičius, V. Jusas, and D. Birvinskas, "Using higher order nonlinear operators for SVM classification of EEG data," *Elektronika ir Elektrotechnika*, vol. 119, no. 3, pp. 99–102, 2012.

[79] J. Tichonov, O. Kurasova, and E. Filatovas, "Vaizdų klasifikavimas pagal suspaudimo algoritmo poveikį jų kokybei." *Informacijos Mokslai/Information Sciences*, vol. 73, 2015.

[80] D. K. Jain, S. B. Dubey, R. K. Choubey, A. Sinhal, S. K. Arjaria, A. Jain, and H. Wang, "An approach for hyperspectral image classification by optimizing SVM using self organizing map," *Journal of Computational Science*, vol. 25, pp. 252–259, 2018.

[81] P. Danėnas and G. Garšva, "Selection of support vector machines based classifiers for credit risk domain," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3194–3204, 2015.

[82] F. Deng, S. Guo, R. Zhou, and J. Chen, "Sensor multifault diagnosis with improved support vector machines," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1053–1063, 2015.

[83] T. T. Hoang, M.-Y. Cho, M. N. Alam, and Q. T. Vu, "A novel differential particle swarm optimization for parameter selection of support vector machines for monitoring metal-oxide surge arrester conditions," *Swarm and Evolutionary Computation*, vol. 38, pp. 120–126, 2018.

[84] R. Morisi, D. N. Manners, G. Gnecco, N. Lanconelli, C. Testa, S. Evangelisti, L. Talozzi, L. L. Gramegna, C. Bianchini, G. Calandra-Buonaura *et al.*, "Multi-class parkinsonian disorders classification with quantitative MR markers and graph-based features using support vector machines," *Parkinsonism & related disorders*, vol. 47, pp. 64–70, 2018.

[85] R. Ren, D. D. Wu, and T. Liu, "Forecasting stock market movement direction using sentiment analysis and support vector machine," *IEEE Systems Journal*, vol. 13, no. 1, pp. 760–770, 2018.

[86] S. Liu and I. Lee, "Email Sentiment Analysis Through k-Means Labeling and Support Vector Machine Classification," *Cybernetics and Systems*, vol. 49, no. 3, pp. 181–199, 2018.

[87] Y. Chen and Z. Zhang, "Research on text sentiment analysis based on CNNs and SVM," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2018, pp. 2731–2734.

[88] S. Wang, M. Zhou, G. Fei, Y. Chang, and B. Liu, "Contextual and position-aware factorization machines for sentiment classification," *arXiv preprint arXiv:1801.06172*, 2018.

[89] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[90] F. Colas and P. Brazdil, "Comparison of SVM and some older classification algorithms in text classification tasks," in *IFIP International Conference on Artificial Intelligence in Theory and Practice*. Springer, 2006, pp. 169–178.

[91] G. Fei and B. Liu, "Social media text classification under negative covariate shift," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 2347–2356.

[92] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[93] Z. A. Sunkad *et al.*, "Feature selection and hyperparameter optimization of SVM for human activity recognition," in *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*. IEEE, 2016, pp. 104–109.

[94] H. Osman, M. Ghafari, and O. Nierstrasz, "Hyperparameter optimization to improve bug prediction accuracy," in *Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE Workshop on*. IEEE, 2017, pp. 33–38.

[95] J. Liu and E. Zio, "SVM hyperparameters tuning for recursive multi-step-ahead prediction," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3749–3763, 2017.

[96] H. Han, X. Cui, Y. Fan, L. Xu, and H. Wu, "Product sale prediction method based on support vector machine model with parameter optimization," 2018, patent App. CN108305103 (A).

[97] S. Lu, M. Li, M. Zhang, and N. Zhong, "Method For Optimizing Support Vector Machine On Basis Of Particle Swarm Optimization Algorithm," 2018, Patent App. WO2018072351 (A1).

[98] M. Asif, A. Ishtiaq, H. Ahmad, H. Aljuaid, and J. Shah, "Sentiment Analysis of Extremism in Social Media from Textual Information," *Telematics and Informatics*, p. 101345, 2020.

[99] C. Meng, J. Zhang, X. Ye, F. Guo, and Q. Zou, "Review and comparative analysis of machine learning-based phage virion protein identification methods," *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics*, p. 140406, 2020.

[100] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[101] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[102] K. Kim, "A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree," *Pattern Recognition*, vol. 60, pp. 157–163, 2016.

[103] C.-C. Wu, Y.-L. Chen, Y.-H. Liu, and X.-Y. Yang, "Decision tree induction with a constrained number of leaf nodes," *Applied Intelligence*, vol. 45, no. 3, pp. 673–685, 2016.

[104] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, 2017.

[105] S. Y. Ooi, S. C. Tan, and W. P. Cheah, "Temporal Sleuth Machine with decision tree for temporal classification," *Soft Computing*, vol. 22, no. 24, pp. 8077–8095, 2018.

[106] M. Saremi and F. Yaghmaee, "Improving evolutionary decision tree induction with multi-interval discretization," *Computational Intelligence*, vol. 34, no. 2, pp. 495–514, 2018.

[107] S. Nor, S. Heryati, S. Ismail, and B. W. Yap, "Personal bankruptcy prediction using decision tree model," *Journal of Economics, Finance and Administrative Science*, vol. 24, no. 47, pp. 157–170, 2019.

[108] Y. Cai, H. Zhang, Q. He, and S. Sun, "New classification technique: fuzzy oblique decision tree," *Transactions of the Institute of Measurement and Control*, vol. 41, no. 8, pp. 2185–2195, 2019.

[109] R. Primartha, B. A. Tama, A. Arliansyah, and K. J. Miraswan, "Decision tree combined with PSO-based feature selection for sentiment

analysis," in *Journal of Physics: Conference Series*, vol. 1196, no. 1. IOP Publishing, 2019, p. 012018.

[110] V. Y. Kulkarni, P. K. Sinha, and M. C. Petare, "Weighted hybrid decision tree model for random forest classifier," *Journal of The Institution of Engineers (India): Series B*, vol. 97, no. 2, pp. 209–217, 2016.

[111] A. Chaudhary, S. Kolhe, and R. Kamal, "An improved random forest classifier for multi-class classification," *Information Processing in Agriculture*, vol. 3, no. 4, pp. 215–222, 2016.

[112] M. Z. Alam, M. S. Rahman, and M. S. Rahman, "A Random Forest based predictor for medical data classification using feature ranking," *Informatics in Medicine Unlocked*, vol. 15, p. 100180, 2019.

[113] V. A. Fitri, R. Andreswari, and M. A. Hasibuan, "Sentiment Analysis of Social Media Twitter with Case of Anti-LGBT Campaign in Indonesia using Naïve Bayes, Decision Tree, and Random Forest Algorithm," *Procedia Computer Science*, vol. 161, pp. 765–772, 2019.

[114] G. Khanvilkar and D. Vora, "Smart Recommendation System Based on Product Reviews Using Random Forest," in *2019 International Conference on Nascent Technologies in Engineering (ICNTE)*. IEEE, 2019, pp. 1–9.

[115] M. Z. Ansari, M. Aziz, M. Siddiqui, H. Mehra, and K. Singh, "Analysis of Political Sentiment Orientations on Twitter," *Procedia Computer Science*, vol. 167, pp. 1821–1828, 2020.

[116] R. Kumar and J. Kaur, "Random Forest-Based Sarcastic Tweet Classification Using Multiple Feature Collection," in *Multimedia Big Data Computing for IoT Applications*. Springer, 2020, pp. 131–160.

[117] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[118] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[119] W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, and K. Drammey, "Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 154–161.

[120] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 2001, pp. 1–17.

[121] H. Lei and V. Govindaraju, "Speeding up multi-class SVM evaluation by PCA and feature selection," *Feature Selection for Data Mining*, vol. 72, 2005.

[122] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," in *Advances in neural information processing systems*, 2005, pp. 521–528.

[123] O. Meyer, B. Bischl, and C. Weihs, "Support vector machines on large data sets: Simple parallel approaches," in *Data Analysis, Machine Learning and Knowledge Discovery*. Springer, 2014, pp. 87–95.

[124] M. Nandan, P. P. Khargonekar, and S. S. Talathi, "Fast SVM training using approximate extreme points," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 59–98, 2014.

[125] S. Wang, Z. Li, C. Liu, X. Zhang, and H. Zhang, "Training data reduction to speed up SVM training," *Applied intelligence*, vol. 41, no. 2, pp. 405–420, 2014.

[126] L. Guo and S. Boukir, "Fast data selection for SVM training using ensemble margin," *Pattern Recognition Letters*, vol. 51, pp. 112–119, 2015.

[127] X. Mao, Z. Fu, O. Wu, and W. Hu, "Fast kernel SVM training via support vector identification," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1554–1559.

[128] S. Mourad, A. Tewfik, and H. Vikalo, "Data subset selection for efficient SVM training," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 833–837.

[129] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[130] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis and Applications*, vol. 13, no. 2, pp. 131–141, 2010.

[131] B. Zheng, S. W. Yoon, and S. S. Lam, "Breast cancer diagnosis based on feature extraction using a hybrid of K-means and support vector machine algorithms," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1476–1482, 2014.

[132] T. Tang, S. Chen, M. Zhao, W. Huang, and J. Luo, "Very large-scale data classification based on K-means clustering and multi-kernel SVM," *Soft Computing*, vol. 23, no. 11, pp. 3793–3801, 2019.

[133] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[134] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[135] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.

[136] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 481–488.

[137] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Springer, 2019, pp. 3–33.

[138] I. Ilhan and G. Tezel, "A genetic algorithm–support vector machine method with parameter optimization for selecting the tag SNPs," *Journal of biomedical informatics*, vol. 46, no. 2, pp. 328–340, 2013.

[139] J.-S. Chou, M.-Y. Cheng, Y.-W. Wu, and A.-D. Pham, "Optimizing parameters of support vector machine using fast messy genetic algorithm for dispute classification," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3955–3964, 2014.

[140] C. Yongqi, "LS_SVM parameters selection based on hybrid complex particle swarm optimization," *Energy Procedia*, vol. 17, pp. 706–710, 2012.

[141] G. Garšva and P. Danėnas, "Particle swarm optimization for linear support vector machines based classifier selection," *Nonlinear Analysis: Modelling and Control*, vol. 19, no. 1, pp. 26–42, 2014.

[142] C.-L. Huang, "ACO-based hybrid classification system with feature subset selection and model parameters optimization," *Neurocomputing*, vol. 73, no. 1-3, pp. 438–448, 2009.

[143] X. Zhang, X. Chen, and Z. He, "An ACO-based algorithm for parameter optimization of support vector machines," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6618–6628, 2010.

[144] M. Ahmad, S. Aftab, M. S. Bashir, N. Hameed, I. Ali, and Z. Nawaz, "SVM optimization for sentiment analysis," *International Journal of Advanced Computer Science & Applications*, vol. 9, no. 4, pp. 393–398, 2018.

[145] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[146] R. G. Mantovani, A. L. Rossi, J. Vanschoren, B. Bischl, and A. C. De Carvalho, "Effectiveness of random search in SVM hyper-parameter tuning," in *2015 International Joint Conference on Neural Networks (IJCNN)*. Ieee, 2015, pp. 1–8.

[147] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[148] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[149] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*, 2013, pp. 115–123.

[150] F. Pedregosa, "Hyperparameter optimization with approximate gradient," *arXiv preprint arXiv:1602.02355*, 2016.

[151] O. Y. Bakhteev and V. V. Strijov, "Comprehensive analysis of gradient-based hyperparameter optimization algorithms," *Annals of Operations Research*, pp. 1–15, 2019.

[152] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[153] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, "Hyperparameter Optimization Using a Genetic Algorithm Considering Verification Time in a Convolutional Neural Network," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 721–726, 2020.

[154] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[155] O. Gustafsson, M. Villani, and P. Stockhammar, "Bayesian Optimization of Hyperparameters when the Marginal Likelihood is Estimated by MCMC," *arXiv preprint arXiv:2004.10092*, 2020.

[156] I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," *arXiv preprint arXiv:1604.07269*, 2016.

[157] C.-L. Huang and J.-F. Dun, "A distributed PSO–SVM hybrid system with feature selection and parameter optimization," *Applied soft computing*, vol. 8, no. 4, pp. 1381–1391, 2008.

[158] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert systems with applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[159] X. Gu, T. Li, Y. Wang, L. Zhang, Y. Wang, and J. Yao, "Traffic fatalities prediction using support vector machine with hybrid particle swarm optimization," *Journal of Algorithms & Computational Technology*, vol. 12, no. 1, pp. 20–29, 2018.

[160] Q. Qian, H. Gao, and B. Wang, "A SVM method trained by improved particle swarm optimization for image classification," in *Chinese Conference on Pattern Recognition*. Springer, 2014, pp. 263–272.

[161] X. Zhang, D. Qiu, and F. Chen, "Support vector machine with parameter optimization by a novel hybrid method and its application to fault diagnosis," *Neurocomputing*, vol. 149, pp. 641–651, 2015.

[162] Y. Maali and A. Al-Jumaily, "A novel partially connected cooperative parallel PSO-SVM algorithm: Study based on sleep apnea detection," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[163] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, 1995, pp. 39–43.

[164] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[165] H. Zhenya, W. Chengjian, Y. Luxi, G. Xiqi, Y. Susu, R. C. Eberhart, and Y. Shi, "Extracting rules from fuzzy neural network by particle swarm optimisation," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 74–77.

[166] N. F. Da Silva, E. R. Hruschka, and E. R. Hruschka Jr, "Tweet sentiment analysis with classifier ensembles," *Decision Support Systems*, vol. 66, pp. 170–179, 2014.

[167] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Systems with Applications*, vol. 62, pp. 1–16, 2016.

[168] I. Perikos and I. Hatzilygeroudis, "Recognizing emotions in text using ensemble of classifiers," *Engineering Applications of Artificial Intelligence*, vol. 51, pp. 191–201, 2016.

[169] I. Perikos and I. Hatzilygeroudis, "A classifier ensemble approach to detect emotions polarity in social media," in *Special Session on Social Recommendation in Information Systems*, vol. 2. SCITEPRESS, 2016, pp. 363–370.

[170] C. Catal and M. Nangir, "A sentiment classification model based on multiple classifiers," *Applied Soft Computing*, vol. 50, pp. 135–141, 2017.

[171] J. Sadhasivam and R. B. Kalivaradhan, "Sentiment Analysis of Amazon Products Using Ensemble Machine Learning Algorithm," *International Journal of Mathematical, Engineering and Management Sciences*, vol. 4, no. 2, pp. 508–520, 2019.

[172] Z. Tan, Y. Zhang, C. Zhang, R. Huang, P. Lei, and X. Duan, "Research on The Text Emotion of Multinomial Naïve Bayes Integration Algorithm," in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, 2019, pp. 107–111.

[173] A. Alrehili and K. Albalawi, "Sentiment Analysis of Customer Reviews Using Ensemble Method," in *2019 International Conference on Computer and Information Sciences (ICCIS)*. IEEE, 2019, pp. 1–6.

[174] E. D. Liddy, "Natural language processing," 2001.

[175] S. Sun, C. Luo, and J. Chen, "A review of natural language processing techniques for opinion mining systems," *Information fusion*, vol. 36, pp. 10–25, 2017.

[176] A. Tommasel and D. Godoy, "Short-text feature construction and selection in social media data: a survey," *Artificial Intelligence Review*, vol. 49, no. 3, pp. 301–338, 2018.

[177] B.-K. Wang, Y.-F. Huang, W.-X. Yang, and X. Li, "Short text classification based on strong feature thesaurus," *Journal of Zhejiang University SCIENCE C*, vol. 13, no. 9, pp. 649–659, 2012.

[178] Z. Liu, W. Yu, W. Chen, S. Wang, and F. Wu, "Short text feature selection for micro-blog mining," in *2010 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010, pp. 1–4.

[179] G. Tang, Y. Xia, W. Wang, R. Lau, and F. Zheng, "Clustering tweets using Wikipedia concepts." in *LREC*. Citeseer, 2014, pp. 2262–2267.

[180] S. Verma, S. Vieweg, W. J. Corvey, L. Palen, J. H. Martin, M. Palmer, A. Schram, and K. M. Anderson, "Natural language processing to the rescue? extracting" situational awareness" tweets during mass emergency," in *Fifth international AAAI conference on weblogs and social media*, 2011.

[181] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 1555–1565.

[182] O. Ozdikis, P. Senkul, and H. Oguztuzun, "Semantic expansion of tweet contents for enhanced event detection in twitter," in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2012, pp. 20–24.

[183] X. Hu, N. Sun, C. Zhang, and T.-S. Chua, "Exploiting internal and external semantics for the clustering of short texts using world knowledge," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 919–928.

[184] J. Tang, X. Hu, H. Gao, and H. Liu, "Unsupervised feature selection for multi-view data in social media," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 270–278.

[185] Y. Fang, H. Zhang, Y. Ye, and X. Li, "Detecting hot topics from Twitter: A multiview approach," *Journal of Information Science*, vol. 40, no. 5, pp. 578–593, 2014.

[186] J. Tang and H. Liu, "Feature selection with linked data in social media," in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 118–128.

[187] H. Saif, Y. He, and H. Alani, "Alleviating data sparsity for twitter sentiment analysis." CEUR Workshop Proceedings (CEUR-WS. org), 2012.

[188] S. Amir, M. B. Almeida, B. Martins, J. Filgueiras, and M. J. Silva, "Tugas: Exploiting unlabelled data for twitter sentiment analysis," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 673–677.

[189] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 151–160.

[190] C. Li, A. Sun, and A. Datta, "Twevent: segment-based event detection from tweets," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 155–164.

[191] M. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. IEEE, 2013, pp. 1–5.

[192] A. Yousefpour, R. Ibrahim, and H. N. A. Hamed, "Ordinal-based and frequency-based integration of feature selection methods for sentiment analysis," *Expert Systems with Applications*, vol. 75, pp. 80–93, 2017.

[193] A. Abdi, S. M. Shamsuddin, S. Hasan, and J. Piran, "Machine learning-based multi-documents sentiment-oriented summarization using linguistic treatment," *Expert Systems with Applications*, vol. 109, pp. 66–85, 2018.

[194] D. Q. Nguyen, D. Q. Nguyen, T. Vu, and S. B. Pham, "Sentiment classification on polarity reviews: an empirical study using rating-based features," in *Proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis*, 2014, pp. 128–135.

[195] B. Agarwal, N. Mittal *et al.*, *Prominent feature extraction for sentiment analysis*. Springer, 2016.

[196] A. Ortigosa, J. M. Martín, and R. M. Carro, "Sentiment analysis in Facebook and its application to e-learning," *Computers in human behavior*, vol. 31, pp. 527–541, 2014.

[197] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining." in *LREc*, vol. 10, no. 2010, 2010, pp. 1320–1326.

[198] E. Boiy and M.-F. Moens, "A machine learning approach to sentiment analysis in multilingual Web texts," *Information retrieval*, vol. 12, no. 5, pp. 526–558, 2009.

[199] K. K. Bharti and P. K. Singh, "Hybrid dimension reduction by integrating feature selection with feature extraction method for text clustering," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3105–3114, 2015.

[200] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of research and development*, vol. 1, no. 4, pp. 309–317, 1957.

[201] T. Tokunaga and I. Makoto, "Text categorization based on weighted inverse document frequency," in *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ)*. Citeseer, 1994.

[202] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[203] G. Salton and C.-S. Yang, "On the specification of term values in automatic indexing," Cornell University, Tech. Rep., 1973.

[204] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[205] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 43–52.

[206] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[207] P. B. Brazdil and C. Soares, "A comparison of ranking methods for classification algorithm selection," in *European conference on machine learning*. Springer, 2000, pp. 63–75.

[208] H. R. Neave and P. L. Worthington, *Distribution-free tests*. Unwin Hyman, 1988.

[209] C. Heumann, M. Schomaker *et al.*, *Introduction to statistics and data analysis*. Springer, 2016.

[210] C. Rain, "Sentiment Analysis in Amazon Reviews Using Probabilistic Machine Learning," *Swarthmore College*, 2013.

[211] T. Shaikh and D. Deshpande, "Feature Selection Methods in Sentiment Analysis and Sentiment Classification of Amazon Product Reviews," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 36, no. 4, pp. 225–230, 2016.

[212] T. U. Haque, N. N. Saber, and F. M. Shah, "Sentiment analysis on large scale Amazon product reviews," in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*. IEEE, 2018, pp. 1–6.

Appendix A. Default parameters of machine learning algorithms

Multinomial naïve Bayes[9]:

- **alpha: float, optional (default=1.0)**. Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).
- **fit_prior: boolean, optional (default=True)**. Whether to learn class prior probabilities or not. If false, a uniform prior will be used.
- **class_prior: array-like, size (n_classes), optional (default=None)**. Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

Linear support vector machines[9]:

- **C: float, optional (default=1.0)**. Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.
- **penalty: str, 'l1' or 'l2' (default='l2')**. Specifies the norm used in the penalization. The 'l2' penalty is the standard used in SVC. The 'l1' leads to coef_ vectors that are sparse.
- **dual: bool, (default=True)**. Select the algorithm to either solve the dual or primal optimization problem. Prefer dual=False when n_samples > n_features.
- **tol: float, optional (default=1e-4)**. Tolerance for stopping criteria.
- **loss: str, 'hinge' or 'squared_hinge' (default='squared_hinge')**. Specifies the loss function. 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss.
- **multi_class: str, 'ovr' or 'crammer_singer' (default='ovr')**. Determines the multi-class strategy if y contains more than two classes. 'ovr' trains n_classes one-vs-rest classifiers, while 'crammer_singer' optimizes a joint objective over all classes.
- **fit_intercept: bool, optional (default=True)**. Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (i.e. data is expected to be already centered).
- **class_weight: {dict, 'balanced'}, optional (default=None)**. Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one.

[9] `https://scikit-learn.org/`

- **verbose: int, (default=0)**. Enable verbose output.
- **random_state: int, RandomState instance or None, optional (default=None)**. The seed of the pseudo random number generator to use when shuffling the data for the dual coordinate descent (if dual=True). When dual=False the underlying implementation of LinearSVC is not random and random_state has no effect on the results. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
- **max_iterint, (default=1000)**. The maximum number of iterations to be run.

Logistic regression (aka logit, MaxEnt)[9]:

- **C: float (default: 1.0)**. Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- **penalty: {'l1', 'l2', 'elasticnet', 'none'} (default='l2')**. Used to specify the norm used in the penalization.
- **dual: bool (default=False)**. Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n_samples > n_features.
- **tol: float (default=1e-4)**. Tolerance for stopping criteria.
- **fit_intercept: bool (default=True)**. Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.
- **class_weight: dict or 'balanced' (default=None)**. Weights associated with classes in the form class_label: weight. If not given, all classes are supposed to have weight one.
- **random_state: int, RandomState instance (default=None)**. The seed of the pseudo random number generator to use when shuffling the data.
- **solver: {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'} (default='lbfgs')**. Algorithm to use in the optimization problem.
- **max_iter: int (default=100)**. Maximum number of iterations taken for the solvers to converge.
- **multi_class: {'auto', 'ovr', 'multinomial'} (default='auto')**. If the option chosen is 'ovr', then a binary problem is fit for each label.

For 'multinomial' the loss minimised is the multinomial loss fit across the entire probability distribution, even when the data is binary.

- **verbose: int (default=0)**. For the liblinear and lbfgs solvers set verbose to any positive number for verbosity.
- **warm_start: bool (default=False)**. When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution.
- **n_jobs: int (default=None)**. Number of CPU cores used when parallelizing over classes if multi_class='ovr'.

Decision tree[9]:

- **criterion: {'gini', 'entropy'} (default='gini')**. The function to measure the quality of a split. Supported criteria are 'gini' for the Gini impurity and 'entropy' for the information gain.
- **splitter: {'best', 'random'} (default='best')**. The strategy used to choose the split at each node. Supported strategies are 'best' to choose the best split and 'random' to choose the best random split.
- **max_depth: int (default=None)**. The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split: int or float (default=2)**. The minimum number of samples required to split an internal node.
- **min_samples_leaf: int or float (default=1)**. The minimum number of samples required to be at a leaf node.
- **min_weight_fraction_leaf: float (default=0.0)**. The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features: int, float or {'auto', 'sqrt', 'log2'} (default=None)**. The number of features to consider when looking for the best split.
- **random_state: int or RandomState (default=None)**. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
- **max_leaf_nodes: int (default=None)**. Grow a tree with

max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

- **min_impurity_decreasefloat (default=0.0)**. A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- **min_impurity_split: float (default=1e-7)**. Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.
- **class_weight: dict, list of dict or 'balanced' (default=None)**. Weights associated with classes in the form class_label: weight. If None, all classes are supposed to have weight one.
- **ccp_alpha: non-negative float (default=0.0)**. Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen.

Random forest[9]:

- **n_estimators: integer, optional (default=100)**. The number of trees in the forest.
- **criterion: string, optional (default = 'gini')**. The function to measure the quality of a split. Supported criteria are 'gini' for the Gini impurity and 'entropy' for the information gain. Note: this parameter is tree-specific.
- **max_depth: integer or None, optional (default=None)**. The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split: int, float, optional (default=2)**. The minimum number of samples required to split an internal node.
- **min_samples_leaf: int, float, optional (default=1)**. The minimum number of samples required to be at a leaf node.
- **min_weight_fraction_leaf: float, optional (default=0)**. The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features: int, float, string or None, optional (default = 'auto')**. The number of features to consider when looking for the best split.

- **max_leaf_nodes: int or None, optional (default=None)**. Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
- **min_impurity_decrease: float, optional (default=0)**. A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- **min_impurity_split: float, (default=1e-7)**. Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.
- **bootstrap: boolean, optional (default=True)**. Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- **oob_score: bool (default=False)**. Whether to use out-of-bag samples to estimate the generalization accuracy.
- **n_jobs: int or None, optional (default=None)**. The number of jobs to run in parallel.
- **random_state: int, RandomState instance or None, optional (default=None)**. Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap=True) and the sampling of the features to consider when looking for the best split at each node (if max_features < n_features).
- **verbose: int, optional (default=0)**. Controls the verbosity when fitting and predicting.
- **warm_start: bool, optional (default=False)**. When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest.
- **class_weight: dict, list of dicts, 'balanced', 'balanced_subsample' or None, optional (default=None)**. Weights associated with classes in the form class_label: weight. If not given, all classes are supposed to have weight one.
- **ccp_alpha: non-negative float, optional (default=0.0)**. Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed.
- **max_samples: int or float (default=None)**. If bootstrap is True, the number of samples to draw from X to train each base estimator.

Appendix B. Classification results

Table B.1. ML_km_30K_SpeedUP ranking results of the experiment cycle with k-Means clustering

Method	ACC	PPV	NPV	TPR	TNR	F_1score	AUC
sentiment140 dataset							
MNB_km_	76.19%	76.69%	75.70%	75.24%	77.14%	75.96%	84.58%
rank	3	4	3	3	3	3	3
LR_km_	78.14%	77.66%	78.64%	79.02%	77.26%	78.33%	85.98%
rank	1	1	1	1	2	1	1
LSVM_km_	77.30%	76.78%	77.83%	78.26%	76.33%	77.51%	85.55%
rank	2	3	2	2	4	2	2
RF_km_	74.34%	77.18%	72.05%	69.13%	79.56%	72.93%	83.60%
rank	4	2	4	5	1	4	4
DT_km_	70.12%	70.00%	70.25%	70.43%	69.81%	70.21%	77.04%
rank	5	5	5	4	5	5	5
$\overline{rank}_{LR_1} = 1.14$ | $\overline{rank}_{LSVM_1} = 2.43$ | $\overline{rank}_{MNB_1} = 3.14$ | $\overline{rank}_{RF_1} = 3.43$ | $\overline{rank}_{DT_1} = 4.86$							
AmazonTest dataset							
MNB_km_	84.20%	85.10%	83.34%	82.91%	85.48%	83.99%	92.23%
rank	3	3	3	3	4	3	3
LR_km_	88.28%	88.43%	88.13%	88.08%	88.47%	88.25%	95.01%
rank	1	1	1	1	1	1	2
LSVM_km_	87.74%	87.75%	87.72%	87.71%	87.76%	87.73%	95.11%
rank	2	2	2	2	2	2	1
RF_km_	78.45%	83.45%	74.76%	70.98%	85.92%	76.71%	90.34%
rank	4	4	4	5	3	4	4
DT_km_	73.61%	73.68%	73.54%	73.46%	73.75%	73.57%	82.23%
rank	5	5	5	4	5	5	5
$\overline{rank}_{LR_2} = 1.14$ | $\overline{rank}_{LSVM_2} = 1.86$ | $\overline{rank}_{MNB_2} = 3.14$ | $\overline{rank}_{RF_2} = 4$ | $\overline{rank}_{DT_2} = 4.86$							
$\overline{Rank}_{LR} = 1.14$ | $\overline{Rank}_{LSVM} = 2.14$ | $\overline{Rank}_{MNB} = 3.14$ | $\overline{Rank}_{RF} = 3.71$ | $\overline{Rank}_{DT} = 4.86$							

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.2. Accuracy of ML_30K_SpeedUP and ML_km_30K_SpeedUP in each CV fold of the experiment cycle with k-Means clustering

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
MNB_ (%)	76.05	75.99	75.95	76.01	76.03	76.07	75.93	76.04	75.98	75.96
MNB_km_ (%)	76.10	76.20	76.10	76.27	76.19	76.30	76.13	76.17	76.23	76.18
LR_ (%)	78.07	78.08	77.99	78.11	78.06	78.11	77.97	78.05	77.99	78.05
LR_km_ (%)	78.10	78.16	78.09	78.19	78.14	78.24	78.07	78.18	78.09	78.16
LSVM_ (%)	77.12	77.09	77.08	77.15	77.10	77.16	77.05	77.10	77.09	77.11
LSVM_km_ (%)	77.28	77.33	77.24	77.31	77.30	77.37	77.25	77.26	77.31	77.32
RF_ (%)	73.18	73.25	73.12	73.19	73.22	73.19	73.03	73.31	73.21	73.13
RF_km_ (%)	74.35	74.37	74.25	74.35	74.44	74.40	74.31	74.21	74.35	74.40
DT_ (%)	68.82	68.91	68.79	68.86	68.78	68.84	68.77	68.68	68.78	68.88
DT_km_ (%)	70.14	70.13	70.09	70.19	70.11	70.12	70.19	70.01	70.12	70.10
AmazonTest dataset										
MNB_ (%)	84.17	84.12	84.12	84.08	84.12	84.08	84.15	84.18	84.15	84.12
MNB_km_ (%)	84.13	84.14	84.23	84.12	84.15	84.18	84.24	84.25	84.26	84.25
LR_ (%)	88.23	88.25	88.22	88.21	88.24	88.20	88.24	88.25	88.23	88.21
LR_km_ (%)	88.28	88.28	88.30	88.25	88.28	88.26	88.29	88.29	88.28	88.26
LSVM_ (%)	87.59	87.60	87.59	87.58	87.59	87.58	87.61	87.58	87.60	87.57
LSVM_km (%)	87.77	87.73	87.73	87.73	87.74	87.75	87.76	87.74	87.72	87.70
RF_ (%)	77.95	78.14	77.96	78.05	78.15	77.85	78.02	78.17	77.96	77.91
RF_km_ (%)	78.57	78.27	78.57	78.42	78.46	78.42	78.46	78.42	78.46	78.45
DT_ (%)	73.26	73.29	73.22	73.21	73.25	73.20	73.38	73.18	73.17	73.34
DT_km_ (%)	73.53	73.74	73.56	73.51	73.62	73.60	73.61	73.62	73.62	73.65

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.3. ML_3_km_30K_SpeedUP ranking results of the experiment cycle with the full proposed hybrid method

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
sentiment140 dataset							
MNB_3_km_	76.93%	77.39%	76.49%	76.10%	77.77%	76.74%	85.38%
rank	3	4	3	3	2	3	3
LR_3_km_	78.68%	78.15%	79.23%	79.62%	77.75%	78.88%	86.44%
rank	1	2	1	2	3	1	2
$LSVM_3$_km_	78.51%	77.87%	79.18%	79.65%	77.36%	78.75%	86.84%
rank	2	3	2	1	4	2	1
RF_3_km_	76.14%	79.51%	73.47%	70.45%	81.84%	74.70%	85.25%
rank	4	1	5	5	1	4	4
DT_3_km_	73.49%	73.45%	73.52%	73.57%	73.41%	73.51%	82.16%
rank	5	5	4	4	5	5	5

Continued on next page

Table B.3 continued from the previous page.

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
$\overline{rank}_{LR_1} = 1.71$ | $\overline{rank}_{LSVM_1} = 2.14$ | $\overline{rank}_{MNB_1} = 3$ | $\overline{rank}_{RF_1} = 3.43$ | $\overline{rank}_{DT_1} = 4.71$							
AmazonTest dataset							
MNB_3_km_	84.59%	85.51%	83.72%	83.30%	85.88%	84.39%	92.56%
rank	3	4	3	3	4	3	3
LR_3_km_	88.55%	88.70%	88.40%	88.35%	88.75%	88.53%	95.18%
rank	2	2	2	2	3	2	2
$LSVM_3$_km_	88.90%	88.90%	88.89%	88.89%	88.90%	88.90%	95.78%
rank	1	1	1	1	2	1	1
RF_3_km_	81.58%	87.56%	77.24%	73.61%	89.54%	79.98%	92.45%
rank	4	3	5	5	1	4	4
DT_3_km_	77.72%	77.97%	77.48%	77.28%	78.16%	77.62%	87.49%
rank	5	5	4	4	5	5	5
$\overline{rank}_{LSVM_2} = 1.14$ | $\overline{rank}_{LR_2} = 2.14$ | $\overline{rank}_{MNB_2} = 3.29$ | $\overline{rank}_{RF_2} = 3.71$ | $\overline{rank}_{DT_2} = 4.71$							
$\overline{Rank}_{LSVM} = 1.64$ | $\overline{Rank}_{LR} = 1.93$ | $\overline{Rank}_{MNB} = 3.14$ | $\overline{Rank}_{RF} = 3.57$ | $\overline{Rank}_{DT} = 4.71$							

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.4. ML_5_km_30K_SpeedUP ranking results of the experiment cycle with the full proposed hybrid method

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
sentiment140 dataset							
MNB_5_km_	77.15%	77.60%	76.72%	76.34%	77.97%	76.97%	85.57%
rank	3	4	3	3	2	3	4
LR_5_km_	78.84%	78.28%	79.42%	79.82%	77.86%	79.04%	86.53%
rank	2	2	2	2	3	2	2
$LSVM_5$_km_	78.93%	78.25%	79.65%	80.14%	77.72%	79.18%	87.14%
rank	1	3	1	1	4	1	1
RF_5_km_	76.87%	80.44%	74.06%	71.02%	82.73%	75.44%	85.79%
rank	4	1	5	5	1	4	3
DT_5_km_	75.06%	75.08%	75.05%	75.03%	75.10%	75.05%	83.64%
rank	5	5	4	4	5	5	5
$\overline{rank}_{LSVM_1} = 1.71$ | $\overline{rank}_{LR_1} = 2.14$ | $\overline{rank}_{MNB_1} = 3.14$ | $\overline{rank}_{RF_1} = 3.29$ | $\overline{rank}_{DT_1} = 4.71$							
AmazonTest dataset							
MNB_5_km_	84.64%	85.54%	83.78%	83.37%	85.91%	84.44%	92.59%
rank	3	4	3	3	4	4	4
LR_5_km_	88.64%	88.80%	88.48%	88.43%	88.85%	88.62%	95.22%
rank	2	3	2	2	3	3	2
$LSVM_5$_km_	89.29%	89.30%	89.27%	89.27%	89.30%	89.28%	95.93%
rank	1	1	1	1	2	2	1

Continued on next page

Table B.4 continued from the previous page.

Method	ACC	PPV	NPV	TPR	TNR	$F_1 score$	AUC
RF_5_km_	82.53%	88.95%	77.93%	74.29%	90.77%	80.96%	92.89%
rank	4	2	5	5	1	5	3
DT_5_km_	79.31%	79.65%	78.98%	78.73%	79.89%	79.19%	88.75%
rank	5	5	4	4	5	1	5
$\overline{rank}_{LSVM_2} = 1.29$ | $\overline{rank}_{LR_2} = 2.43$ | $\overline{rank}_{MNB_2} = 3.57$ | $\overline{rank}_{RF_2} = 3.57$ | $\overline{rank}_{DT_2} = 4.14$							
$\overline{Rank}_{LSVM} = 1.5$ | $\overline{Rank}_{LR} = 2.29$ | $\overline{Rank}_{MNB} = 3.36$ | $\overline{Rank}_{RF} = 3.43$ | $\overline{Rank}_{DT} = 4.43$							

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.5. Accuracy of ML_km_30K_SpeedUP and ML_3_km_30K_SpeedUP in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
MNB_km (%)	76.10	76.20	76.10	76.27	76.19	76.30	76.13	76.17	76.23	76.18
MNB_3_km_ (%)	76.76	76.92	76.87	76.98	76.95	77.10	76.86	76.96	76.98	76.93
LR_km (%)	78.10	78.16	78.09	78.19	78.14	78.24	78.07	78.18	78.09	78.16
LR_3_km_ (%)	78.67	78.67	78.64	78.72	78.67	78.76	78.67	78.71	78.59	78.71
LSVM_km_ (%)	77.28	77.33	77.24	77.31	77.30	77.37	77.25	77.26	77.31	77.32
$LSVM_3$_km_ (%)	78.46	78.54	78.42	78.50	78.56	78.67	78.47	78.44	78.49	78.53
RF_km_ (%)	74.35	74.37	74.25	74.35	74.44	74.40	74.31	74.21	74.35	74.40
RF_3_km_ (%)	76.22	76.17	76.00	76.09	76.20	76.19	76.10	76.13	76.13	76.22
DT_km_ (%)	70.14	70.13	70.09	70.19	70.11	70.12	70.19	70.01	70.12	70.10
DT_3_km_ (%)	73.49	73.56	73.50	73.62	73.44	73.40	73.54	73.34	73.53	73.45
AmazonTest dataset										
MNB_km_ (%)	84.13	84.14	84.23	84.12	84.15	84.18	84.24	84.25	84.26	84.25
MNB_3_km_ (%)	84.49	84.56	84.60	84.48	84.57	84.59	84.61	84.70	84.66	84.65
LR_km_ (%)	88.28	88.28	88.30	88.25	88.28	88.26	88.29	88.29	88.28	88.26
LR_3_km_ (%)	88.54	88.56	88.59	88.51	88.54	88.55	88.55	88.57	88.54	88.55
LSVM_km_ (%)	87.77	87.73	87.73	87.73	87.74	87.75	87.76	87.74	87.72	87.70
$LSVM_3$_km_ (%)	88.92	88.92	88.87	88.89	88.89	88.95	88.91	88.87	88.88	88.86
RF_km_ (%)	78.57	78.27	78.57	78.42	78.46	78.42	78.51	78.42	78.46	78.45
RF_3_km_ (%)	81.64	81.41	81.69	81.46	81.74	81.63	81.63	81.45	81.53	81.59
DT_km_ (%)	73.53	73.74	73.56	73.51	73.62	73.60	73.61	73.62	73.62	73.65
DT_3_km_ (%)	77.60	77.89	77.69	77.56	77.74	77.73	77.67	77.70	77.84	77.79

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.6. Accuracy of ML_3_km_30K_SpeedUP and ML_5_km_30K_SpeedUP in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
MNB_3_km_ (%)	76.76	76.92	76.87	76.98	76.95	77.10	76.86	76.96	76.98	76.93
MNB_5_km_ (%)	77.03	77.16	77.07	77.21	77.19	77.24	77.13	77.17	77.13	77.21
LR_3_km_ (%)	78.67	78.67	78.64	78.72	78.67	78.76	78.67	78.71	78.59	78.71
LR_5_km_ (%)	78.82	78.87	78.77	78.90	78.83	78.96	78.77	78.85	78.76	78.87
$LSVM_3$_km_ (%)	78.46	78.54	78.42	78.50	78.56	78.67	78.47	78.44	78.49	78.53
$LSVM_5$_km_ (%)	78.91	78.97	78.84	78.97	78.94	78.99	78.91	78.91	78.90	78.97
RF_3_km_ (%)	76.22	76.17	76.00	76.09	76.20	76.19	76.10	76.13	76.13	76.22
RF_5_km_ (%)	76.92	76.89	76.80	76.82	76.95	76.95	76.82	76.88	76.82	76.91
DT_3_km_ (%)	73.49	73.56	73.50	73.62	73.44	73.40	73.54	73.34	73.53	73.45
DT_5_km_ (%)	75.06	75.18	75.03	75.16	75.02	75.07	75.03	74.98	75.04	75.07
AmazonTest dataset										
MNB_3_km_ (%)	84.49	84.56	84.60	84.48	84.57	84.59	84.61	84.70	84.66	84.65
MNB_5_km_ (%)	84.57	84.58	84.68	84.55	84.61	84.61	84.69	84.71	84.70	84.71
LR_3_km_ (%)	88.54	88.56	88.59	88.51	88.54	88.55	88.55	88.57	88.54	88.55
LR_5_km_ (%)	88.64	88.65	88.66	88.61	88.63	88.65	88.64	88.64	88.64	88.64
$LSVM_3$_km_ (%)	88.92	88.92	88.87	88.89	88.89	88.95	88.91	88.87	88.88	88.86
$LSVM_5$_km_ (%)	89.31	89.31	89.29	89.27	89.27	89.31	89.29	89.29	89.26	89.25
RF_3_km_ (%)	81.64	81.41	81.69	81.46	81.74	81.63	81.63	81.45	81.53	81.59
RF_5_km_ (%)	82.65	82.42	82.58	82.51	82.61	82.51	82.54	82.52	82.40	82.58
DT_3_km_ (%)	77.60	77.89	77.69	77.56	77.74	77.73	77.67	77.70	77.84	77.79
DT_5_km_ (%)	79.22	79.42	79.31	79.14	79.33	79.31	79.34	79.31	79.33	79.38

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.7. Accuracy of $LSVM_5$_km_30K_SpeedUP and LR_5_km_30K_SpeedUP in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
LR_5_km_ (%)	78.82	78.87	78.77	78.90	78.83	78.96	78.77	78.85	78.76	78.87
$LSVM_5$_km_ (%)	78.91	78.97	78.84	78.97	78.94	78.99	78.91	78.91	78.90	78.97
AmazonTest dataset										
LR_5_km_ (%)	88.64	88.65	88.66	88.61	88.63	88.65	88.64	88.64	88.64	88.64
$LSVM_5$_km_ (%)	89.31	89.31	89.29	89.27	89.27	89.31	89.29	89.29	89.26	89.25

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.8. Accuracy of ML_5_km_30K_SpeedUP and classical ML algorithm in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
RF_5_km_ (%)	76.92	76.89	76.80	76.82	76.95	76.95	76.82	76.88	76.82	76.91
classical RF (%)	76.11	76.29	76.22	76.30	76.17	76.19	76.05	76.18	76.15	76.16
DT_5_km_ (%)	75.06	75.18	75.03	75.16	75.02	75.07	75.03	74.98	75.04	75.07
classical DT (%)	72.07	72.11	72.07	72.07	72.14	72.06	72.12	72.13	72.00	72.11
AmazonTest dataset										
MNB_5_km_ (%)	84.57	84.58	84.68	84.55	84.61	84.61	84.69	84.71	84.70	84.71
classical MNB (%)	84.45	84.47	84.45	84.48	84.47	84.48	84.52	84.48	84.47	84.43
RF_5_km_ (%)	82.65	82.42	82.58	82.51	82.61	82.51	82.54	82.52	82.40	82.58
classical RF (%)	80.42	80.31	80.44	80.12	80.09	80.88	80.39	80.56	80.14	80.31
DT_5_km_ (%)	79.22	79.42	79.31	79.14	79.33	79.31	79.34	79.31	79.33	79.38
classical DT (%)	77.31	77.42	77.20	77.30	77.35	77.15	77.38	77.25	77.40	77.34

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.9. Accuracy of ML_km_30K_SpeedUP and ML^{PSO}_km_30K_SpeedUP in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
LR_km_ (%)	78.10	78.16	78.09	78.19	78.14	78.24	78.07	78.18	78.09	78.16
LR^{PSO}_km_ (%)	78.11	78.17	78.07	78.16	78.12	78.16	78.07	78.14	78.10	78.14
LSVM_km_ (%)	77.28	77.33	77.24	77.31	77.30	77.37	77.25	77.26	77.31	77.32
$LSVM^{PSO}$_km_ (%)	78.04	78.14	78.08	78.19	78.13	78.21	78.04	78.15	78.08	78.13

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.10. Results of PSO tuning on sentiment140 dataset performed with LSVM

CV	Data	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_1 ($acc_2 - acc_1$)	C_{tune}	Δ_2 ($acc_3 - acc_1$)	C_{PSO}
CV1	1st	77.04%	78.10%	78.22%	1.06%	0.1	1.18%	0.06246
	2nd	77.18%	78.25%	78.26%	1.07%	0.1	1.08%	0.10079
	3rd	76.97%	78.20%	78.29%	1.23%	0.1	1.32%	0.07326
	4th	77.08%	77.94%	78.02%	0.86%	0.1	0.94%	0.14623
	5th	76.75%	78.03%	78.05%	1.28%	0.1	1.30%	0.10826
	1st	77.14%	78.05%	78.19%	0.91%	0.1	1.05%	0.07333
	2nd	76.96%	77.94%	77.97%	0.98%	0.1	1.01%	0.12298

Continued on next page

Table B.10 continued from the previous page.

CV	Data	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_1 ($acc_2 - acc_1$)	C_{tune}	Δ_2 ($acc_3 - acc_1$)	C_{PSO}
CV2	3rd	76.89%	78.15%	78.18%	1.26%	0.1	1.29%	0.08048
	4th	77.02%	78.25%	78.40%	1.23%	0.1	1.38%	0.07676
	5th	76.96%	78.15%	78.20%	1.19%	0.1	1.24%	0.11223
CV3	1st	76.91%	78.17%	78.22%	1.26%	0.1	1.31%	0.08546
	2nd	77.22%	78.30%	78.34%	1.08%	0.1	1.12%	0.09296
	3rd	76.96%	78.09%	78.14%	1.13%	0.1	1.18%	0.07326
	4th	76.66%	78.11%	78.23%	1.45%	0.1	1.57%	0.08243
	5th	77.00%	78.17%	78.21%	1.17%	0.1	1.21%	0.11362
CV4	1st	77.37%	78.35%	78.38%	0.98%	0.1	1.01%	0.13370
	2nd	76.97%	77.96%	78.05%	0.99%	0.1	1.08%	0.13907
	3rd	76.64%	77.62%	77.74%	0.98%	0.1	1.10%	0.11705
	4th	77.27%	78.36%	78.41%	1.09%	0.1	1.14%	0.11806
	5th	76.75%	78.04%	78.05%	1.29%	0.1	1.30%	0.09315
CV5	1st	77.08%	78.21%	78.25%	1.13%	0.1	1.17%	0.08416
	2nd	76.77%	77.94%	78.12%	1.17%	0.1	1.35%	0.05658
	3rd	77.26%	78.07%	78.17%	0.81%	0.1	0.91%	0.12122
	4th	77.05%	77.91%	77.98%	0.86%	0.1	0.93%	0.16853
	5th	77.50%	78.46%	78.56%	0.96%	0.2	1.06%	0.13961
CV6	1st	77.05%	78.27%	78.28%	1.22%	0.1	1.23%	0.10142
	2nd	76.73%	77.99%	78.16%	1.26%	0.1	1.43%	0.06470
	3rd	76.77%	77.86%	78.02%	1.09%	0.1	1.25%	0.04829
	4th	76.78%	77.89%	77.96%	1.11%	0.1	1.18%	0.15426
	5th	76.59%	77.87%	77.93%	1.28%	0.1	1.34%	0.08601
CV7	1st	77.15%	78.45%	78.53%	1.30%	0.1	1.38%	0.06151
	2nd	76.97%	78.14%	78.25%	1.17%	0.1	1.28%	0.12156
	3rd	77.20%	77.98%	78.07%	0.78%	0.1	0.87%	0.13970
	4th	77.19%	78.33%	78.44%	1.13%	0.1	1.25%	0.07078
	5th	77.31%	78.50%	78.54%	1.19%	0.1	1.23%	0.08743
CV8	1st	77.07%	78.16%	78.19%	1.09%	0.1	1.12%	0.10652
	2nd	76.85%	77.89%	77.97%	1.04%	0.1	1.12%	0.09225
	3rd	77.07%	78.19%	78.33%	1.12%	0.1	1.26%	0.07209
	4th	77.11%	78.56%	78.59%	1.45%	0.1	1.48%	0.08413
	5th	76.96%	77.96%	78.05%	1.0%	0.1	1.09%	0.14011
CV9	1st	77.11%	78.24%	78.26%	1.13%	0.1	1.15%	0.10998
	2nd	77.14%	78.35%	78.35%	1.21%	0.1	1.21%	0.09963
	3rd	77.04%	78.00%	78.09%	0.96%	0.1	1.05%	0.07189
	4th	77.09%	78.19%	78.26%	1.10%	0.1	1.17%	0.06221
	5th	77.03%	78.06%	78.15%	1.03%	0.1	1.12%	0.07794
CV10	1st	77.06%	78.17%	78.33%	1.11%	0.1	1.27%	0.13223
	2nd	77.20%	78.43%	78.55%	1.23%	0.1	1.35%	0.08645
	3rd	77.42%	78.26%	78.41%	0.84%	0.1	0.99%	0.11247

Continued on next page

Table B.10 continued from the previous page.

CV	Data	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_1 ($acc_2 - acc_1$)	C_{tune}	Δ_2 ($acc_3 - acc_1$)	C_{PSO}
	4th	77.33%	78.26%	78.36%	0.93%	0.1	1.03%	0.14616
	5th	77.10%	78.13%	78.25%	1.03%	0.1	1.15%	0.06572

Table B.11. Results of PSO tuning on AmazonTest dataset performed with LSVM

CV	Data	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_1 ($acc_2 - acc_1$)	C_{tune}	Δ_2 ($acc_3 - acc_1$)	C_{PSO}
	1st	87.56%	88.31%	88.40%	0.75%	0.2	0.84%	0.14700
	2nd	87.40%	88.35%	88.47%	0.95%	0.2	1.07%	0.15276
CV1	3rd	87.58%	88.32%	88.42%	0.74%	0.2	0.84%	0.14713
	4th	87.46%	88.36%	88.37%	0.90%	0.2	0.91%	0.19511
	5th	87.40%	88.37%	88.41%	0.97%	0.2	1.01%	0.21929
	1st	87.26%	88.09%	88.20%	0.83%	0.2	0.94%	0.13333
	2nd	87.33%	88.34%	88.38%	1.01%	0.2	1.05%	0.17476
CV2	3rd	87.68%	88.57%	88.61%	0.89%	0.3	0.94%	0.15670
	4th	87.39%	88.30%	88.33%	0.91%	0.2	0.94%	0.20296
	5th	87.26%	88.18%	88.25%	0.92%	0.2	0.99%	0.13678
	1st	87.11%	88.13%	88.22%	1.02%	0.1	1.11%	0.10908
	2nd	87.56%	88.47%	88.54%	0.91%	0.2	0.98%	0.17474
CV3	3rd	87.29%	88.23%	88.29%	0.94%	0.1	1.0%	0.13681
	4th	87.10%	88.13%	88.17%	1.03%	0.2	1.07%	0.18266
	5th	87.49%	88.30%	88.38%	0.81%	0.3	0.89%	0.24559
	1st	87.92%	88.77%	88.91%	0.85%	0.2	0.99%	0.16078
	2nd	87.68%	88.38%	88.42%	0.70%	0.2	0.74%	0.15654
CV4	3rd	87.81%	88.66%	88.69%	0.85%	0.2	0.88%	0.23245
	4th	87.22%	88.31%	88.37%	1.09%	0.2	1.15%	0.13103
	5th	87.11%	88.06%	88.15%	0.95%	0.2	1.04%	0.17856
	1st	87.24%	88.15%	88.21%	0.91%	0.2	0.97%	0.18523
	2nd	87.45%	88.40%	88.41%	0.95%	0.2	0.96%	0.19971
CV5	3rd	87.55%	88.40%	88.49%	0.85%	0.2	0.94%	0.14573
	4th	87.35%	88.12%	88.14%	0.77%	0.2	0.79%	0.17251
	5th	87.06%	88.22%	88.27%	1.16%	0.2	1.21%	0.15860
	1st	87.26%	88.14%	88.16%	0.88%	0.2	0.90%	0.17073
	2nd	87.28%	88.27%	88.33%	0.99%	0.2	1.05%	0.18628
CV6	3rd	87.45%	88.41%	88.45%	0.96%	0.2	1.0%	0.21723
	4th	87.32%	88.28%	88.33%	0.96%	0.2	1.01%	0.17768
	5th	87.74%	88.41%	88.49%	0.67%	0.2	0.75%	0.13177
	1st	87.70%	88.53%	88.56%	0.83%	0.1	0.86%	0.12571
	2nd	87.11%	88.05%	88.14%	0.94%	0.2	1.03%	0.16029
CV7	3rd	87.54%	88.48%	88.52%	0.94%	0.2	0.98%	0.21895

Continued on next page

Table B.11 continued from the previous page.

CV	Data	ACC_d (acc_1)	ACC_{tune} (acc_2)	ACC_{PSO} (acc_3)	Δ_1 ($acc_2 - acc_1$)	C_{tune}	Δ_2 ($acc_3 - acc_1$)	C_{PSO}
	4th	87.45%	88.49%	88.56%	1.04%	0.2	1.11%	0.14771
	5th	87.13%	88.10%	88.10%	0.97%	0.2	0.97%	0.19679
	1st	87.42%	88.28%	88.31%	0.86%	0.1	0.89%	0.10955
	2nd	87.46%	88.54%	88.59%	1.08%	0.2	1.13%	0.21088
CV8	3rd	87.70%	88.47%	88.52%	0.77%	0.2	0.82%	0.13807
	4th	87.81%	88.67%	88.71%	0.86%	0.2	0.90%	0.22941
	5th	87.24%	88.10%	88.15%	0.86%	0.1	0.91%	0.18526
	1st	87.50%	88.32%	88.39%	0.82%	0.2	0.89%	0.17923
	2nd	87.46%	88.46%	88.52%	1.0%	0.3	1.06%	0.28866
CV9	3rd	87.57%	88.41%	88.45%	0.84%	0.2	0.88%	0.12109
	4th	87.57%	88.51%	88.56%	0.94%	0.2	0.99%	0.11533
	5th	87.99%	88.79%	88.82%	0.80%	0.2	0.83%	0.24146
	1st	87.35%	88.31%	88.38%	0.96%	0.2	1.03%	0.17733
	2nd	87.50%	88.39%	88.44%	0.89%	0.2	0.94%	0.17063
CV10	3rd	87.39%	88.63%	88.64%	1.24%	0.2	1.25%	0.19760
	4th	87.79%	88.54%	88.59%	0.75%	0.2	0.80%	0.16918
	5th	87.42%	88.51%	88.57%	1.09%	0.2	1.15%	0.18131

Table B.12. Results of PSO tuning performed for classical LSVM

CV	ACC_d (acc_1)	ACC_{PSO} (acc_2)	Δ ($acc_2 - acc_1$)	C_{PSO}
sentiment140 dataset				
CV1	77.40%	78.23%	0.83%	0.07706
CV2	77.10%	78.23%	1.23%	0.12161
CV3	77.33%	78.43%	1.10%	0.08602
CV4	77.19%	78.25%	1.06%	0.05472
CV5	77.12%	78.09%	0.97%	0.10539
CV6	77.51%	78.32%	0.81%	0.09774
CV7	76.99%	78.11%	1.12%	0.07228
CV8	76.77%	78.17%	1.40%	0.07039
CV9	77.30%	78.41%	1.11%	0.07723
CV10	77.18%	78.30%	1.12%	0.09720
AmazonTest dataset				
CV1	87.80%	88.54%	0.74%	0.18855
CV2	87.81%	88.54%	0.73%	0.14663
CV3	87.85%	88.70%	0.85%	0.13294
CV4	87.81%	88.67%	0.86%	0.18163

Continued on next page

Table B.12 continued from the previous page.

CV	ACC_d (acc_1)	ACC_{PSO} (acc_2)	Δ ($acc_2 - acc_1$)	C_{PSO}
CV5	87.79%	88.57%	0.78%	0.22259
CV6	87.73%	88.56%	0.83%	0.21497
CV7	87.75%	88.70%	0.95%	0.16057
CV8	87.80%	88.60%	0.80%	0.20158
CV9	87.68%	88.57%	0.89%	0.15131
CV10	88.00%	88.82%	0.82%	0.21164

Table B.13. Accuracy of the proposed hybrid method in each CV fold of the experiment cycle with the full proposed hybrid method

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
sentiment140 dataset										
LSVM_ (%)	77.12	77.09	77.08	77.15	77.10	77.16	77.05	77.10	77.09	77.11
LSVM^{PSO}_km_ (%)	78.04	78.14	78.08	78.19	78.13	78.21	78.04	78.15	78.08	78.13
LSVM$_3$_km_ (%)	78.46	78.54	78.42	78.50	78.56	78.67	78.47	78.44	78.49	78.53
LSVM$_3^{PSO}$_km_ (%)	78.56	78.70	78.51	78.71	78.64	78.62	78.56	78.61	78.58	78.70
LSVM$_5$_km_ (%)	78.91	78.97	78.84	78.97	78.94	78.99	78.91	78.91	78.90	78.97
LSVM$_5^{PSO}$_km_ (%)	78.78	78.86	78.71	78.95	78.87	78.79	78.75	78.82	78.71	78.85
LSVM (%)	79.50	79.54	79.42	79.53	79.55	79.62	79.47	79.52	79.49	79.52
LSVM^{PSO} (%)	79.88	79.90	79.82	79.92	79.92	79.98	79.82	79.91	79.87	79.92
AmazonTest dataset										
LSVM_ (%)	87.59	87.60	87.59	87.58	87.59	87.58	87.61	87.58	87.60	87.57
LSVM^{PSO}_km_ (%)	88.47	88.44	88.43	88.44	88.46	88.48	88.46	88.45	88.43	88.43
LSVM$_3$_km_ (%)	88.92	88.92	88.87	88.89	88.89	88.95	88.91	88.87	88.88	88.86
LSVM$_3^{PSO}$_km_ (%)	88.86	88.85	88.82	88.88	88.90	88.95	88.89	88.84	88.90	88.89
LSVM$_5$_km_ (%)	89.31	89.31	89.29	89.27	89.27	89.31	89.29	89.29	89.26	89.25
LSVM$_5^{PSO}$_km_ (%)	89.03	89.00	89.00	89.00	89.01	89.07	89.05	89.04	89.05	89.04
LSVM (%)	89.38	89.58	89.59	89.65	89.44	89.56	89.69	89.56	89.70	89.59
LSVM^{PSO} (%)	90.25	90.20	90.23	90.24	90.20	90.18	90.26	90.20	90.21	90.20

Underscore "_" means that 30K_SpeedUP should be added at the end.

Table B.14. C values obtained by PSO tuning, random search and Bayesian optimization of the experiment cycle with real-world data

CV	C_{PSO}	C_{RS}	C_{Bopt}	C_{PSO}	C_{RS}	C_{Bopt}
	Person dataset			Event dataset		
CV1	0.16395	0.18788	0.26478	0.35264	0.17426	0.37357
CV2	0.5	0.23655	0.09187	0.32594	0.23655	0.48067
CV3	0.20836	0.17426	0.25998	0.3	0.87517	0.68186
CV4	0.2	0.14207	0.36230	0.17209	0.82932	0.15582
CV5	0.14461	0.14207	0.81331	0.3	0.19420	0.80211
CV6	0.14885	0.17426	0.98462	0.21387	0.56561	0.18215
CV7	0.4	0.27637	0.41633	1.9	0.28671	0.46562
CV8	0.39414	0.17426	1.39327	1.11668	0.32262	1.29922
CV9	0.34895	0.12045	2.05769	0.27715	0.14207	0.39434
CV10	0.3	0.18788	0.42478	0.32752	0.50658	0.43190

Table B.15. Accuracy of ML in each CV fold of the experiment cycle with real-world data

Method	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
Person dataset										
LSVM (%)	73.17	67.60	74.22	70.38	73.52	77.70	73.17	72.82	71.43	71.08
LSVM^{PSO}_287_ (%)	77.70	72.13	77.00	76.66	79.09	79.44	75.61	74.91	72.82	75.61
LSVM$_{RS}$ (%)	74.56	74.91	78.75	76.31	78.75	77.70	75.26	74.91	74.56	76.31
LSVM$_{Bopt}$ (%)	74.56	74.91	79.09	76.66	78.40	74.56	73.52	73.17	69.34	75.61
Event dataset										
LSVM (%)	71.76	68.44	75.08	68.44	72.76	70.76	73.42	68.11	69.77	68.44
LSVM^{PSO}_300_ (%)	71.43	69.77	76.41	70.43	73.09	72.76	73.09	68.77	70.76	72.43
LSVM$_{RS}$ (%)	70.10	70.43	75.42	68.44	70.43	71.43	75.42	68.77	70.43	70.76
LSVM$_{Bopt}$ (%)	71.43	69.77	76.74	70.10	73.09	71.76	74.09	68.77	71.10	72.09

Underscore "_" means that SpeedUP should be added at the end.

Table B.16. Normal distribution

	z-score	p-value	results	H_0
Method	sentiment140 dataset			
classical MNB	0.197	0.906	$p > \alpha$	not rejected
MNB_30K_SpeedUP	1.077	0.584	$p > \alpha$	not rejected
MNB_km_30K_SpeedUP	0.324	0.85	$p > \alpha$	not rejected

Continued on next page

Table B.16 continued from the previous page.

	z-score	p-value	Results	H_0
MNB_3_km_30K_SpeedUP	1.303	0.521	$p > \alpha$	not rejected
MNB_5_km_30K_SpeedUP	1.195	0.55	$p > \alpha$	not rejected
classical LR	0.418	0.812	$p > \alpha$	not rejected
LR_30K_SpeedUP	1.058	0.584	$p > \alpha$	not rejected
LR_km_30K_SpeedUP	0.418	0.811	$p > \alpha$	not rejected
LR_3_km_30K_SpeedUP	1.295	0.523	$p > \alpha$	not rejected
LR_5_km_30K_SpeedUP	0.406	0.816	$p > \alpha$	not rejected
LR^{PSO}_km_30K_SpeedUP	1.182	0.554	$p > \alpha$	not rejected
classical LSVM	1.495	0.473	$p > \alpha$	not rejected
LSVM_30K_SpeedUP	0.323	0.851	$p > \alpha$	not rejected
LSVM_km_30K_SpeedUP	0.104	0.949	$p > \alpha$	not rejected
$LSVM_3$_km_30K_SpeedUP	5.053	0.08	$p > \alpha$	not rejected
$LSVM_5$_km_30K_SpeedUP	0.988	0.61	$p > \alpha$	not rejected
$LSVM^{PSO}$_km_30K_SpeedUP	0.502	0.778	$p > \alpha$	not rejected
$LSVM_3^{PSO}$_km_30K_SpeedUP	0.864	0.649	$p > \alpha$	not rejected
$LSVM_5^{PSO}$_km_30K_SpeedUP	0.234	0.89	$p > \alpha$	not rejected
classical RF	0.038	0.981	$p > \alpha$	not rejected
RF_30K_SpeedUP	1.161	0.56	$p > \alpha$	not rejected
RF_km_30K_SpeedUP	1.396	0.498	$p > \alpha$	not rejected
RF_3_km_30K_SpeedUP	2.25	0.325	$p > \alpha$	not rejected
RF_5_km_30K_SpeedUP	3.01	0.222	$p > \alpha$	not rejected
classical DT	2.062	0.357	$p > \alpha$	not rejected
DT_30K_SpeedUP	0.856	0.652	$p > \alpha$	not rejected
DT_km_30K_SpeedUP	2.764	0.251	$p > \alpha$	not rejected
DT_3_km_30K_SpeedUP	0.165	0.921	$p > \alpha$	not rejected
DT_5_km_30K_SpeedUP	2.32	0.314	$p > \alpha$	not rejected
	AmazonTest dataset			
classical MNB	2.086	0.352	$p > \alpha$	not rejected
MNB_30K_SpeedUP	0.234	0.89	$p > \alpha$	not rejected
MNB_km_30K_SpeedUP	5.093	0.078	$p > \alpha$	not rejected
MNB_3_km_30K_SpeedUP	0.173	0.917	$p > \alpha$	not rejected
MNB_5_km_30K_SpeedUP	4.427	0.109	$p > \alpha$	not rejected
classical LR	0.627	0.731	$p > \alpha$	not rejected
LR_30K_SpeedUP	1.193	0.551	$p > \alpha$	not rejected
LR_km_30K_SpeedUP	0.586	0.746	$p > \alpha$	not rejected
LR_3_km_30K_SpeedUP	1.444	0.486	$p > \alpha$	not rejected
LR_5_km_30K_SpeedUP	5.593	0.061	$p > \alpha$	not rejected
classical LSVM	1.499	0.473	$p > \alpha$	not rejected

Continued on next page

Table B.16 continued from the previous page.

	z-score	p-value	Results	H_0
LSVM_30K_SpeedUP	0.136	0.934	$p > \alpha$	not rejected
LSVM_km_30K_SpeedUP	0.245	0.885	$p > \alpha$	not rejected
$LSVM_3$_km_30K_SpeedUP	0.86	0.651	$p > \alpha$	not rejected
$LSVM_5$_km_30K_SpeedUP	1.216	0.545	$p > \alpha$	not rejected
$LSVM^{PSO}$_km_30K_SpeedUP	1.172	0.556	$p > \alpha$	not rejected
$LSVM_3^{PSO}$_km_30K_SpeedUP	0.54	0.763	$p > \alpha$	not rejected
$LSVM_5^{PSO}$_km_30K_SpeedUP	1.377	0.502	$p > \alpha$	not rejected
classical RF	3.479	0.176	$p > \alpha$	not rejected
RF_30K_SpeedUP	1.084	0.582	$p > \alpha$	not rejected
RF_km_30K_SpeedUP	2.611	0.271	$p > \alpha$	not rejected
RF_3_km_30K_SpeedUP	1.013	0.603	$p > \alpha$	not rejected
RF_5_km_30K_SpeedUP	0.361	0.835	$p > \alpha$	not rejected
classical DT	1.106	0.575	$p > \alpha$	not rejected
DT_30K_SpeedUP	1.254	0.534	$p > \alpha$	not rejected
DT_km_30K_SpeedUP	1.778	0.411	$p > \alpha$	not rejected
DT_3_km_30K_SpeedUP	0.03	0.985	$p > \alpha$	not rejected
DT_5_km_30K_SpeedUP	4.084	0.13	$p > \alpha$	not rejected
	Person dataset			
classical LSVM	1.417	0.492	$p > \alpha$	not rejected
$LSVM^{PSO}$_287_SpeedUP	0.309	0.857	$p > \alpha$	not rejected
$LSVM_{RS}$	1.974	0.373	$p > \alpha$	not rejected
$LSVM_{Bopt}$	1.337	0.512	$p > \alpha$	not rejected
	Event dataset			
classical LSVM	1.306	0.521	$p > \alpha$	not rejected
$LSVM^{PSO}$_300_SpeedUP	1.945	0.378	$p > \alpha$	not rejected
$LSVM_{RS}$	3.205	0.201	$p > \alpha$	not rejected
$LSVM_{Bopt}$	2.573	0.276	$p > \alpha$	not rejected

Konstantinas Korovkinas

HYBRID METHOD FOR TEXTUAL DATA SENTIMENT ANALYSIS

Doctoral Dissertation
Natural Sciences
Informatics (N 009)
Editor Zuzana Šiušaitė

Vilnius University Press
9 Saulėtekio Ave., Building III, LT-10222 Vilnius
Email: info@leidykla.vu.lt, www.leidykla.vu.lt
Print run copies 20