VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's thesis

Estimating the Proportion of E-commerce Enterprises in Lithuania Using Machine Learning Methods

E-komercijos įmonių proporcijos vertinimas Lietuvoje naudojant mašininio mokymosi metodus

Rokas Steponavičius

Supervisor : Dr. Andrius Čiginas

Reviewer : Prof., Dr. (HP) Marijus Radavičius

Vilnius
2025

Summary

The goal of this thesis is to estimate the proportion of e-commerce enterprises in Lithuania using machine learning methods. In the process, enterprises were manually classified into e-commerce and non-e-commerce to have actual values of enterprises' e-commerce status for models training and performance testing. Companies' websites were scraped to collect the text from them. Machine learning algorithms were combined with NLP methods for classification task. Logistic regression, Naïve Bayes, Support Vector Machines, Extreme Gradient Boosting, and BERT models were used to classify enterprises as e-commerce and non-e-commerce based on the extracted text from their websites. The inverse probability weighting estimator was applied to estimate the proportion of e-commerce enterprises in Lithuania. The estimated proportion of e-commerce enterprises in Lithuania is 0.25. The BERT model achieved the best performance classifying enterprises.

Keywords: e-commerce, machine learning, text classification, natural language processing, enterprises classification.

Santrauka

Šio darbo tikslas yra įvertinti e-komercijos įmonių proporciją Lietuvoje, naudojant mašininio mokymosi metodus. Proceso metu įmonės buvo rankiniu būdu klasifikuotos į e-komercijos ir ne e-komercijos įmones, kad turėti realius duomenis apie įmonių e-komercijos statusą modelių mokymui ir testavimui. Įmonių interneto svetainės buvo nuskaitytos ir jose esantis teksas buvo surinktas. Mašininio mokymosi algoritmai buvo derinami su natūralios kalbos apdorojimo metodais klasifikavimo užduočiai atlikti. Logistinės regresijos, Naiviojo Bajeso klasifikatoriaus, atraminių vektorių mašinų, ekstremalaus gradientų didinimo ir BERT modeliai buvo naudojami klasifikuoti įmones į e-komercijos ir ne e-komercijos įmones, remiantis jų interneto svetainėse esančiais tekstais. Atvirkštinis tikimybinis svertinis įvertinys buvo taikomas, siekiant įvertinti e-komercijos įmonių proporciją Lietuvoje. Apskaičiuota, kad e-komercijos įmonių proporcijos Lietuvoje įvertis yra 0.25. Geriausią rezultatą klasifikuojant įmones pasiekė BERT modelis.

Raktiniai žodžiai: e-komercija, mašininis mokymasis, teksto klasifikavimas, natūralios kalbos apdorojimas, įmonių klasifikavimas.

List of Figures

1 figure. The proportion of E-commerce enterprises in Lithuania and EU 2014-2023. Source: Eurostat 11
2 figure. Optimal SVM classifier [4] 23
3 figure. Extreme gradient boosting model schema 25
4 figure. The structure of BERT model [33] 27
5 figure. Confusion matrix 29
6 figure. Propensity scores 35

List of Tables

1 table. Optimal hyperparameters 32
2 table. Performance of classification models 33
3 table. Logistic regression models for propensity scores model 34
4 table. Performance of logistic regression models for propensity scores model 35

Contents

Summary . . . 2

Santrauka . . . 3

List of Figures . . . 4

List of Tables . . . 5

List of abbreviations . . . 7

Introduction . . . 8

1 Literature review . . . 10
1.1 E-commerce description . . . 10
1.2 E-commerce in Lithuania . . . 10
1.3 Related works . . . 11

2 Estimation of the proportion of e-commerce enterprises . . . 16
2.1 Data . . . 16
2.2 Data preprocessing . . . 17
2.2.1 Text normalization . . . 17
2.2.2 Tokenization . . . 18
2.2.3 Term Frequency-Inverse Document Frequency . . . 19
2.3 Models for websites classification . . . 20
2.3.1 Logistic Regression . . . 20
2.3.2 Naïve Bayes . . . 21
2.3.3 Support Vector Machines . . . 22
2.3.4 Extreme Gradient Boosting . . . 24
2.3.5 BERT model . . . 26
2.4 Performance evaluation metrics . . . 28
2.5 The Inverse Probability Weighting Estimator . . . 30

3 Results . . . 32
3.1 Hyperparameters . . . 32
3.2 Performance of classification models . . . 33
3.3 Propensity scores . . . 34
3.4 The proportion of e-commerce enterprises . . . 35

4 Conclusions and recommendations . . . 37

Appendix 1. Python code . . . 41

Appendix 2. Usage of AI . . . 50

List of abbreviations

URL	Uniform Resource Locator
HTML	Hypertext Markup Language
XML	Extensible Markup Language
TF-IDF	Term Frequency-Inverse Document Frequency
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting
IPW	The Inverse Probability Weighting estimator

Introduction

In recent years, e-commerce has become a significant sector of the global economy. The online shopping is transforming the commerce sector and the global economy. In Lithuania, as in many other countries, the number of e-commerce enterprises has grown rapidly in recent years due to technological advances, the expansion of digital payment systems, increased internet access, and evolving consumer preferences toward online shopping. Due to the growing sales and the number of e-commerce enterprises, the government needs accurate and efficient methods to distinguish e-commerce enterprises from other enterprises. This is beneficial for policymakers and businesses because they need reliable statistics to make data-driven decisions. Traditional methods of identifying e-commerce enterprises are based on sample surveys, industry reports and financial data. These methods are costly and time-consuming and may not always provide up-to-date information. These methods fail to keep the pace with the changing landscape of online commerce. It is challenging to develop a consistent and scalable classification system because physical commerce companies tomorrow can become e-commerce companies. An automated approach based on machine learning application is an innovative solution to this problem. Machine learning combined with Natural Language Processing (NLP) methods is a powerful tool for text classification. By analyzing patterns, keywords, and linguistic features found in the text on companies' websites, machine learning models can be trained to classify companies as e-shops or non-e-shops with high accuracy. It is a scalable, data-driven solution for estimating the proportion of E-commerce enterprises in Lithuania.

The main goal of this thesis is to estimate the proportion of e-commerce enterprises in Lithuania using machine learning methods. This study aims to classify companies based on textual patterns extracted from their websites by applying various machine learning models, including Logistic regression, Naïve Bayes, Support Vector Machines, Extreme Gradient Boosting and deep learning BERT model. Natural Language Processing (NLP) techniques will be utilized to preprocess and analyze the textual data, enabling the models to classify companies as e-commerce or non-e-commerce. As data of scraped websites is a non-probability sample of the survey population, the proportion of e-commerce enterprises in Lithuania will be estimated by applying the inverse probability weighting estimator. The estimated proportion will be compared with statistics by the survey of usage of information technologies in enterprises, which is conducted by the State Data Agency. The objectives of this thesis are:

- Conduct the literature review.
- Train machine learning models to classify enterprises as e-commerce and non-e-commerce based on the text extracted from their websites.
- Compare the performance of machine learning models classifying enterprises as e-commerce and non-e-commerce.
- Estimate the proportion of e-commerce enterprises in Lithuania.

The findings of this research have the potential to significantly contribute to the understanding of Lithuania's E-commerce sector. This study can contribute to the academic literature on applying machine learning to real-world classification problems by estimating the proportion of e-commerce enterprises. Additionally, the approach developed in this thesis could be applied to other countries or industries to classify businesses as e-shops or not e-shops based on the content of their websites.

The thesis is structured as follows: a comprehensive review of the literature is presented in Section 1. Section 2 describes the methodology. The results are presented in Section 3. Finally, conclusions and recommendations for further work are presented in Section 4.

1 Literature review

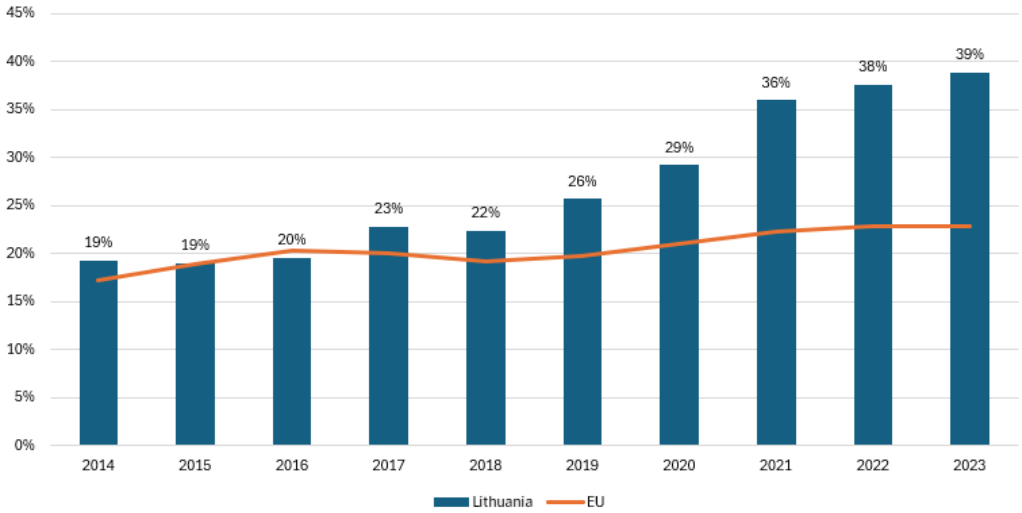
In this section, a definition of e-commerce will be described, and the trends of the e-commerce sector in Lithuania will be analyzed by reports of global institutions. Related works about websites and text classification using machine learning methods will be reviewed in this section.

1.1 E-commerce description

According to Eurostat [10], an e-commerce company is defined as an enterprise that sells goods or services electronically via the internet or other online platforms. Specifically, Eurostat refers to an e-commerce company as one that conducts transactions through a website, mobile app, or automated data exchange, where orders are made electronically but payments and delivery can occur offline. In its surveys and reports, Eurostat [11] measures e-commerce activity based on the percentage of businesses that have made online sales during a given period. Typically, e-commerce companies sell to individuals (B2C), other businesses (B2B) and public institutions through electronic channels. Eurostat's e-commerce statistics helps to track markets trends and notice changes and differences in e-commerce sector in European Union. According to Eurostat's methodology [10], e-commerce sales are classified based on whether they occur through websites or apps (web sales) or Electronic Data Interchange (EDI) for more structured and automated business transactions. Companies are considered e-commerce businesses if they generate revenue through these digital sales.

1.2 E-commerce in Lithuania

According to the Eurostat database, in 2023, 39% of businesses were engaged in e-commerce, i.e. selling goods or services over computer networks, in Lithuania. Lithuania has the highest number of e-commerce enterprises in European Union (EU) countries in 2023, higher than Sweden (38%) and Denmark (37%). As Figure 1 shows, the proportion of e-commerce companies in Lithuania increased from 19% in 2014 to 39% in 2023. According to the Figure 1, the percentage of e-commerce companies in Lithuania is steadily higher than in the EU. For example, in 2023, 39% of businesses were engaged in e-commerce in Lithuania, while in the EU, only 29% of companies were e-commerce.



1 figure. *The proportion of E-commerce enterprises in Lithuania and EU 2014-2023. Source: Eurostat*

According to the Portal for Official Statistics [31], 61% of large enterprises, 46% of medium-sized enterprises and 35% of small enterprises were engaged in e-commerce in Lithuania in 2023. The largest share of e-commerce enterprises was in information and communication (50%), trade (48%) and professional, scientific and technical activities (39%).

The rise of e-commerce enterprises in recent years has transformed customers purchasing behaviour. According to a report by Eurostat [9], in 2023, 70% of individuals aged 16 to 74 in the European Union (EU) reported ordering goods or services online in the previous 12 months. This metric increased from 60% in 2019 to 70% in 2023 in the EU. Most of the population is online shoppers in the Netherlands, Denmark, and Sweden, where over 89% of individuals in these countries make online purchases. Lithuania has also experienced significant e-commerce growth. The percentage of individuals making online purchases in Lithuania increased from 48% in 2019 to 61% in 2023. The main reasons for this growth are improved digital infrastructure, including e-logistics and digital payment systems and the COVID-19 pandemic's impact on purchasing products online.

According to the report by Mordor Intelligence [24], the future of e-commerce in Lithuania is very optimistic. The e-commerce market is forecasted to grow significantly in Lithuania. According to the forecast, the compound annual growth rate (CAGR) will be 18% in 2024-2029. This growth is driven by increased internet and smartphone usage, improved 5G infrastructure, and a more significant consumer shift towards online shopping. Lithuania is positioned as a leading e-commerce market in the Baltic region. In the retail market, the online sales share is 7.2% and will increase by 8.7% (pessimistic scenario) or 10.1% (optimistic scenario) by 2029 in Lithuania. According to the forecast, the e-commerce sector will become a more important part of retail and Lithuania's economy.

1.3 Related works

Enterprises classification into e-commerce and non-e-commerce based on their websites is a challenging and essential task for policymakers and national statistics departments. The article by De Fausti, Pugliese and Zardetto [7] from the Italian National Institute of Statistics shows how deep learning can help accurately identify e-commerce websites from their web content. The study is es-

pecially relevant to this master's thesis as it aims to differentiate e-commerce businesses from other companies by analysing web-scraped text. The authors describe the challenges of using unstructured data for text classification tasks. Unstructured data does not have a predefined manner. Unlike structured data, which fits easily into tables or relational databases, unstructured data is more complex due to its lack of a specific format or structure. Scraped data from websites is unstructured data. Unstructured data often contains detailed information that is unavailable through structured data. Unstructured data can potentially have very beneficial subtle details. For example, terms like 'shopping cart,' 'checkout,' and 'online store' could indicate that the company is e-commerce. The authors developed a sophisticated deep learning model that uses convolutional neural networks (CNNs) with word embeddings to process and classify large volumes of textual data extracted from companies' websites. Typically, web-scraped texts are enormous and have a very low signal-to-noise ratio. The researchers used an innovative False Positive Reduction (FPR) framework to increase model accuracy by systematically increasing the signal-to-noise ratio. The authors constructed a multi-layer processing pipeline that begins with transforming text into a grayscale image format using word embeddings method. According to the results, the most accurate model achieved F1 score of 0.72, precision of 0.73, recall of 0.72, and an accuracy of 0.89. This CNN model outperformed all the alternative machine learning models already tested in the Italian National Institute of Statistics for the same enterprises classification task. The researches emphasize that websites scrapping and machine learning application to classify companies as e-commerce and non-e-commerce is a scalable and autonomous method. Websites scrapping and classification algorithms can be repeated more frequently than the surveys about companies could be done. Therefore, it allows the national statistics departments to have more reliable and up-to-date statistics, enabling policymakers to make data-driven decisions.

Before De Fausti, Pugliese and Zardetto's publication, the other researchers, Barcaroli et al. [1] from the Italian National Institute of Statistics, published an article in 2016 about companies classification as e-commerce and non-e-commerce using web scraping and machine learning models. This study focuses on the dual aspects of improving data collection and processing. The methodology section outlines the combination of web scraping and machine learning processes to convert unstructured web data into structured formats that would be appropriate for classification models. In this research, companies were included which answered the Italian National Institute of Statistics' survey about e-commerce status. Websites of these companies were scraped and textual data was used to train classification models. It is a significant step forward for web scraping and machine learning applications to get needed information instead of yearly surveys of companies. Logistic regression, Naïve Bayes, Random Forest and SVM models were trained to classify companies as e-commerce and non-e-commerce in this research. According to the results, SVM showed the best performance with accuracy of 84% and F1 score of 59%. SVM was followed by Random Forest, which achieved an 83% accuracy and an F1 score of 55%.

Mirończuk and Protasiewicz [23] provide an extensive overview of text classification methodologies using machine learning algorithms. The authors describe the steps of the text classification pipeline, including data acquisition, data preprocessing, feature extraction, model training, and performance evaluation. They discuss various feature extraction techniques, such as bag-of-words and

TF-IDF. These methods are essential for relevant features extraction, which can significantly influence the accuracy of classification models. The review also explores several machine learning methods for text classification, such as Support Vector Machines, Decision Trees, and Neural Networks. The researchers compare these techniques and evaluate their suitability for different text classification tasks. Mirończuk and Protasiewicz describe various performance evaluation metrics for text classification, such as accuracy, precision, recall, and F1 score. They suggest a nuanced model evaluation method for text classification task, which includes the micro and macro averaging methods. Using this approach, scientists can get a more comprehensive understanding of model performance across multiple classes. The authors also highlight several challenges facing the field of text classification, such as the need for models to adapt dynamically to new, unseen data and the integration of domain-specific knowledge into the classification process.

In the comprehensive study by Somesha et al. [29], the authors focused on the application of deep learning techniques for effectively classifying websites as either phishing or non-phishing. Three types of deep learning models were utilised in the study: Deep Neural Network (DNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). Each model was chosen for their specific capabilities in handling different data processing and pattern recognition aspects. The DNN model was designed with multiple hidden layers. As a result of multiple hidden layers, the model is able to learn from the complex and non-linear relationships in the data. The main feature of LSTM networks is capturing temporal dependencies and retaining information over long periods. It was applied to recognise patterns in sequential data. The CNN model was creatively used to analyse website text and structure. By viewing website data like an image, where text and HTML tags form visual-like patterns, CNNs use their ability to process spatial data hierarchies to detect subtle signs of phishing. According to the performance results, the DNN model achieved an accuracy of 99.52%, LSTM reached 99.57%, and CNN reached 99.43%. The researchers emphasize the efficiency of using deep learning in phishing detection and state that their trained models outperform many traditional machine learning algorithms.

Matoševic et al. [22] explore machine learning techniques to classify web pages based on their search engine optimization (SEO) levels. The research shows how learning algorithms can effectively assist in the SEO process, which is critical for improving web page visibility and ranking in search engines. Traditional machine learning models, such as Logistic Regression, Decision Trees, Support Vector Machines, Naïve Bayes, and K-nearest Neighbours (kNN), were trained and tested on a dataset of 6000 web pages. SEO experts manually classified these web pages into three categories: low, medium, and high level. It is a multi-class supervised classification task. The aim of this research is to create a scalable SEO evaluation tool based on machine learning models. The features used for training included both on-page elements like meta tags, keyword density, and structural HTML features, which are essential in determining a page's SEO score. According to the results, the accuracy of models is between 55% and 70%. KNN model achieved the highest accuracy at 70%, followed by Decision Trees at 68%.

Shaffi and Muthulakshmi [28] analyze machine learning techniques to classify web pages based on their SEO levels. The authors manually categorized web pages into three groups – low, medium,

and high – based on SEO levels. The researchers combined expert knowledge with machine learning algorithms to solve a multi-label supervised classification task. Decision Trees, Naive Bayes, KNN, SVM, and Logistic Regression models were used in the research. The authors used the ensemble approach, combining these models to maximize their strengths while reducing their weaknesses. For example, the main advantage of decision trees is transparency, SVM offers robustness, Naïve Bayes offers speed, as a result, combining them ensures that the classification system is accurate and efficient. The results showed that the accuracy of models is between 57% and 68%.

Markkandeyan and Indra Devi [21] explore advanced feature selection methods to achieve high accuracy and efficiency of machine learning algorithms for web page classification. The authors developed a technique which effectively reduces the negative impact of high dimensionality of web page data. Web scraped data is usually high dimensional, and its preprocessing is a challenging task. The authors evaluate several machine learning models, their advantages and their capabilities for web page classification. Support Vector Machines, K-Nearest Neighbors, Naïve Bayes and Decision Trees models were trained to classify web pages. This paper provides a comprehensive methodology for applying machine learning to solve real-world problems in web classification. The advanced feature selection and machine learning techniques explored in this study directly apply to classifying e-commerce enterprises. This methodological approach improves the speed and accuracy of the classification and offers scalable solutions that can adapt to the dynamic nature of e-commerce content on the web.

Gupta and Bhatia [13] describe a detailed design of an ensemble model that combines the strengths of advanced neural network models to classify web pages. The authors introduced an innovative model that combines the Bidirectional Encoder Representations from Transformers (BERT) model and a deep inception network with residual connections. This advanced architecture is created to maximize the performance of supervised text classification task. The BERT model is selected for its capability to understand the context of every word in a sentence by looking at the words that come before and after it. This ability is beneficial for categorizing web pages, where the context of words is essential. BERT is the first layer in this ensemble model, processing the text to a vector and transforming the text into a more appropriate data type for classification. After the BERT model, the inception network with residual connections is used to extract complex features. This network works simultaneously on multiple scales, extracting minor and significant features, which is essential for analyzing web pages with different layouts and structures. The training process involved fine-tuning the BERT layer with web-specific data to adapt its pre-trained general language understanding to the specialized supervised task of web page classification. The model was optimized using regularization and batch normalization to improve training dynamics and minimize the risk of overfitting. According to the results, the accuracy of the ensemble model is 95%, while standalone deep learning models like CNNs and traditional machine learning models like SVM and Random Forest achieve 86-90% accuracy. In conclusion, this ensemble model, constructed of the BERT model and deep inception network with residual connections, significantly outperformed traditional machine learning models and standalone deep learning models.

González-Carvajal and Garrido-Merchán [12] analyze the effectiveness of the BERT model com-

pared to traditional machine learning methods for text classification. The authors focused on sentiment analysis, classifying consumers' reviews into positive, negative, and neutral categories. The dataset of 50000 reviews was used in the sentiment analysis, where 80% of cases were split for training and 20% for testing. The researchers used a standard data preprocessing methodology, where all text was converted to lowercase, and punctuation marks were removed. TF-IDF method was applied for features extraction. In the sentiment analysis, Bert model and traditional machine learning models, such as Logistic regression, Naïve Bayes and Support Vector Machines were trained to classify customers' reviews. According to the results, the BERT model achievied 94% accuracy and significantly outperformed traditional machine learning models across various metrics, including accuracy, precision, recall, and F1 score. The researchers emphasize that the main reason why BERT model achieved better performance resuls is BERT model's ability to understand complex language nuances better than traditional methods, which is a crucial advantage for sentiment analysis where the context can significantly influences classification accuracy.

Roumeliotis et al. [27], in their article, solve binary supervised text classification task by categorizing articles as genuine and fake news. The dataset, used in a research, contains 72134 news articles, where 48% of them are labeled as real and 52% as fake. 80% of articles were split for training and 20% for testing. Data preprocessing includes text normalization, stemming, lemmatization and tokenization. The advanced machine learning models were applied for classification task, such as Convolutional Neural Networks (CNN), BERT, and Generative Pre-trained Transformers (GPT). Each model was selected for specific features. According to the results, the BERT model showed the best performance in classifying articles into genuine and fake news. Pre-trained and fine-tuned for this task, the BERT model achieved higher results in accuracy, precision, recall, and F1 score than CNNs and GPT models.

2 Estimation of the proportion of e-commerce enterprises

The main goal of the analytical part is to train machine learning models to classify companies as e-commerce and non-e-commerce based on the text extracted from companies' websites and estimate the proportion of e-commerce enterprises in Lithuania. The objectives of the analytical part are:

1. Manually classify enterprises as e-commerce and non-e-commerce enterprises.
2. Web scrape enterprises' websites to extract the text from their main pages of websites.
3. Train machine learning models to classify enterprises as e-commerce and non-e-commerce enterprises based on the text extracted from their websites.
4. Compare the performance of machine learning models.
5. Estimate the proportion of e-commerce enterprises in Lithuania.

2.1 Data

In this section, the structure of the data and the data collection process are described, including the web scrapping process and manual companies' classification as e-commerce and non-e-commerce companies based on their websites. The State Data Agency (Statistics Lithuania) provided a list of 10830 companies which operate in Lithuania and have websites.

All enterprises from the list were manually classified as e-commerce and non-e-commerce to have actual values (or labels) of e-commerce status. Enterprises were manually classified to know the real proportion of e-commerce enterprises in a population of scraped websites. Actual values of e-commerce status are also needed to train machine learning models because supervised learning models must know which enterprises are e-commerce and which are non-e-commerce enterprises. In that way, models try to learn different patterns from the text, which are attributes of e-commerce or non-e-commerce companies. Actual values of companies' e-commerce status are also needed for testing models to check how accurately models predict which companies are e-commerce companies and which are not. Companies has been classified according to Eurostat e-commerce enterprise definition that e-commerce enterprise is defined as an enterprise that sells goods or services electronically via the internet or other online platforms. The manual classification involved systematically examining each enterprise's website to determine whether it engaged in e-commerce according to the definition provided. Each website was visited directly to determine whether it offers an online ordering system. Key indicators of an e-commerce company are a shopping cart, product catalogs and a checkout.

Web scrapping methods were used to extract text from companies' websites. As it is defined by Barcaroli et al. [2], "Web scraping is the process of automatically collecting information from the World Wide Web based on tools (called scrapers, internet robots, bots, etc.) that navigate and extract a website's content and store scraped data in local databases." Websites were scrapped using the

Python programming language and the Beautiful Soup library, which is dedicated to web scrapping. This Python library was selected because of its ability to parse hypertext markup language (HTML) and extensible markup language (XML) documents. 7381 websites were scrapped and the text from their main pages was collected. According to Barcaroli et al. [1], the main reasons for not scraping all the websites are:

- Website system restrictions on web scraping: websites implement anti-scraping tools, such as the completely automated public turing test to tell computers and humans apart (CAPTCHA), IP bans, or user-agent verification, which block automated scraping tools.
- The incorrect URLs: If the URLs in a list were incorrect or outdated, the scraper did not scrape these websites.
- Errors with website servers: the most frequently occurring issues are temporary server issues, slow website response time, server overloads, connectivity issues, or incorrect request formatting.
- Technologies not supported by the parser: If a website uses technologies like ADOBE Flash technology, which is not typically supported by standard HTML parsers, a scrapping tool cannot scrape such a website.

After web scraping and manual websites classification, a dataset contains the text from a website and a label of e-commerce status. Each row represents a particular company in the dataset. As a result, the dataset contains information of 7381 scraped companies. According to the manual inspection, 15% of the companies are e-commerce companies, and 85% are non-e-commerce companies. Data was split into training and testing sets: 80% of enterprises were split into training set and 20% of them into testing set.

2.2 Data preprocessing

Text pre-processing and feature extraction are crucial steps for text classification. Web-scraped text from websites is noisy, and the data is unstructured. Most extracted text has many spaces, empty lines, and unnecessary words such as stop words, misspellings, slang, etc. Noise and unnecessary features can negatively impact the performance of text classification models. In this section, the data preprocessing is described.

2.2.1 Text normalization

Web-scraped text data comes from different sources and contains inconsistent capitalization. All characters in a text were converted to lowercase. This standardization method is performed to reduce the complexity of the data and to treat words with the same alphabetic characters as identical regardless of how they appear in the original text [19]. For example, the words ”Checkout”, “CHECKOUT” and ”checkout” would all be converted to ”checkout”.

Removing stop words and punctuation is a critical preprocessing step in text analytics. Stop words are common words that carry minimal useful information for text classification. Stop words in the Lithuanian language typically are pronouns and prepositions, such as: and, or, but, if, that, because. In many related works about text classification, pronouns and prepositions are removed because they appear frequently in the language but do not significantly contribute to the understanding of the text's content [19]. Punctuation does not have significant information by itself and is just noise for text classification algorithms. Therefore, all punctuation marks, such as commas, periods, exclamation marks, and question marks, are removed from the text [2].

The literature on text classification using NLP methods recommends removing numbers from the text. Removing numbers from text data in the context of classifying websites as e-commerce and non-e-commerce can be both useful and not. There are arguments for numbers removing and against it. The main reason to keep numbers is that numbers could have the important information, for example, prices and quantities, which could be highly relevant for distinguishing e-commerce from non-e-commerce sites. The main reasons to remove numbers are:

- Noise and dimensionality reduction: On e-shop websites, there are many different prices, but the prices are different between e-shops. Every unique price would be treated as a unique token. Therefore, numbers would increase the dimensionality of the feature space. This would lead to a sparse matrix with many features without significant value [16].
- Faster computations: removing numbers from the text reduces the number of features, which leads to faster computation. Computation time and the computational resources needed to accomplish a task are crucial when dealing with large datasets [26].
- Higher accuracy of an experiment: an experiment was accomplished with a smaller sample. Machine learning models were trained to classify websites as e-commerce or non-e-commerce on two data samples, excluding and including numbers in a text. The experiment results showed higher accuracy and F1 score of models trained on the data sample with removed numbers.

For these reasons, the final decision was to remove numbers from the data. This reduces noise and dimensionality and improves the speed of computations.

2.2.2 Tokenization

After removing stop words, numbers, and punctuation, the other step in text preprocessing is tokenization. Unstructured text data, such as text on a website, lacks a predefined structure that machines can easily interpret. Tokenization breaks down the text into smaller units called tokens [17]. For this classification task, tokens are words, and the text is split into words. For example, the text from a website is "products categories discounts recommendations cart". In this case, the tokens are as follows:

$$\{\text{"products"}, \text{"categories"}, \text{"discounts"}, \text{"recommendations"}, \text{"cart"}\}$$

One primary reason for tokenization is to convert textual data into a numerical representation that can be processed by machine learning algorithms. Tokens serve as numeric representations of text and are used as features in machine learning pipelines. These features capture important linguistic information [17]. For example, in text classification, the presence or absence of specific tokens can influence the prediction of a particular class, e-commerce or non-e-commerce.

2.2.3 Term Frequency-Inverse Document Frequency

The Term Frequency-Inverse Document Frequency (TF-IDF) method is one of the most widely used techniques in NLP for converting textual data into numerical features. It transforms text into a format that machine learning models can effectively use, making it particularly useful for text classification tasks [20].

The TF-IDF weighting method has two main components: Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how often a term t appears in a document d in equation (1). TF value increases with the frequency of the term t within a document d, emphasizing commonly used words. IDF measures the frequency of a term t across a collection of documents D. It is calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term in equation (2). The IDF value decreases for common terms across many documents, giving higher weight to less frequent but more meaningful words. The TF-IDF weight for a term t in a document d is calculated by multiplying TF and IDF scores, as in equation (3) [20].

$$\text{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{1}$$

$$\text{IDF}(t,D) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right) \tag{2}$$

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D) \tag{3}$$

TF-IDF is the essential method for text classification tasks because it helps to identify terms that are more relevant to individual documents while filtering out common words that are less informative. The TF-IDF method, which weights terms based on their importance, improves the performance of machine learning models, making them more accurate and efficient in processing large amounts of textual data.

In data preprocessing, the TF-IDF method transforms the text from company websites into numerical features. TF-IDF method is beneficial for features extraction. The text of each company website, such as product descriptions and general business information, is analyzed to calculate TF-IDF scores for each word. For example, words like “shopping cart,” “checkout,” and “delivery” might have high TF-IDF scores in e-commerce websites but low scores in non-e-commerce sites [20]. Words that are common across all websites, like “about” or “contact,” will have low IDF values and thus receive lower weights. This helps in reducing the noise in the dataset and ensures that the model focuses on terms that are more relevant to classifying websites as e-commerce or non-e-commerce [17]). The TF-IDF scores are stored in a sparse matrix format, where each row corresponds to a single document

(In this case, a text from a website) and each column corresponds to unique term (word) from the vocabulary of all scrapped websites. The values in the matrix are the TF-IDF scores for each word in each document. These scores indicate the importance of each word within a particular document relative to all documents. This sparse matrix is used for training machine learning models.

2.3 Models for websites classification

A task to classify companies as e-commerce and non-e-commerce based on scraped text from their websites is a supervised binary text classification task. As Jurafsky and Martin [17] wrote: "in supervised machine learning, there is a data set of input observations, each associated with some correct output (a 'supervision signal'). The goal of the algorithm is to learn how to map from a new observation to a correct output." In this section, the methodology of applied machine learning methods are provided.

2.3.1 Logistic Regression

Logistic regression is one of the simplest algorithms used for text classification. Despite this fact, it is one of the most widely used methods in related works about text classification. Logistic regression is a probabilistic model used for supervised machine learning. Logistic regression estimates the probability of a given input belonging to one of two classes, based on a set of features extracted from the input data [17].

According to the methodology of James et al. [16], the independent variables are linearly related to the logarithmic of odds (log odds), which is the dependent variable of logistic regression, as defined in the following equation:

$$\log(\text{odds}) = \log\left(\frac{P}{1-P}\right) \tag{4}$$

The linear relationship between independent variables and log odds can be expanded and defined as:

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{5}$$

where P is the probability of specific event happening, β_0 is the intercept, β_i is the coefficient associated with the independent variable x_i. By exponentiating both sides of the equation (5), we obtain:

$$\left(\frac{P}{1-P}\right) = e^{\beta_0+\beta_1 x_1+\cdots+\beta_n x_n} \tag{6}$$

By converting odds to a simple probability function, finally, the logistic regression is defined as follows:

$$P = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_n x_n}} \tag{7}$$

The maximum likelihood estimation is usually used to estimate logistic regression coefficients. The likelihood function is defined as follows:

$$l(\beta_0, \beta_i) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{8}$$

where x_i is the value of independent variable, y_i is the binary outcome for observation i, n is a number of total observations, and p(x) is the probability of the dependent variable. The estimates of coefficients are chosen to maximize this likelihood function.

According to James et al. [16], logistic regression, for text classification task, estimates the probability that a target variable belongs to a particular category. For binary classification, such as distinguishing e-commerce from non-e-commerce enterprises, the model predicts the probability $P(Y = 1|X)$ where Y is a dependent variable and X are independent variables.

The final classification decision [17] using logistic regression is made according to algorithm in equation (9). The threshold is set at 0.5. For a given x, the predicted class is 1 (e-commerce) if the probability $P(y = 1|x)$ is more than 0.5, and class 0 (non-e-commerce) otherwise.

$$\text{decision}(X) = \begin{cases} 1 & \text{if } P(Y = 1|X) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Logistic regression algorithm is discussed in more detail by Jurafsky and Martin [17], James et al. [16]. The features derived from TF-IDF were used for the logistic regression classification model. Python's sklearn library has been used for the logistic regression implementation.

2.3.2 Naïve Bayes

The Naïve Bayes classifier is a fundamental statistical approach in machine learning. The Naïve Bayes classifier is a probabilistic model based on Bayes' theorem. This method is effective for text categorization tasks due to its simplicity and effectiveness, even with large feature spaces in text data [20].

According to a methodology by Jurafsky and Martin [17], for text classification using Naïve Bayes algorithm, a class is defined as c, instead of y as output variable and d is defined as document instead of x as input variable. In this case, document d_i is a vector of website i from TF-IDF. There is a training set of N documents that each is with a class $\{(d_1, c_1), \ldots, (d_N, c_N)\}$. Naïve Bayes [17] is a probabilistic classifier, meaning that for a document d, the classifier returns the class $\hat{c}$ which has the maximum posterior probability given the document. In equation (10) $\hat{c}$ means the estimate of class and arg max mean an operation that selects class c that maximizes a function (in this case the probability $P(c \mid d)$).

$$\hat{c} = \arg\max_{c \in C} P(c \mid d) \tag{10}$$

The framework of Naïve Bayes classification method [17] is to use Bayes' rule to transform equation

(10) into other probabilities that have some useful properties. By applying Bayes' theorem, equation (10) can be defined as in equation (11):

$$\hat{c} = \underset{c \in C}{\arg\max}\, P(c \mid d) = \underset{c \in C}{\arg\max}\, \frac{P(d \mid c)P(c)}{P(d)} \tag{11}$$

According to Jurafsky and Martin [17], equation (11) can be simplified by dropping the denominator $P(d)$. This is possible because we will be computing $\frac{P(d|c)P(c)}{P(d)}$ for each possible class. But $P(d)$ does not change for each class. Therefore, we can choose the class that maximizes this simpler formula:

$$\hat{c} = \underset{c \in C}{\arg\max}\, P(c \mid d) = \underset{c \in C}{\arg\max}\, P(d \mid c)P(c) \tag{12}$$

As Jurafsky and Martin described [17], we compute the most probable class $\hat{c}$ given some document d by choosing the class which has the highest product of two probabilities: the prior probability of the class $P(c)$ and the likelihood of the document $P(d \mid c)$, as in the following equation:

$$\hat{c} = \underset{c \in C}{\arg\max}\, \underbrace{P(d \mid c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}} \tag{13}$$

To apply the Naïve Bayes classifier to text [17], we use each word in the documents as a feature. The features derived from TF-IDF were used for the classification model. Naïve Bayes algorithm is discussed in more detail by Jurafsky and Martin [17]. Python's sklearn library has been used for the Naïve Bayes implementation.

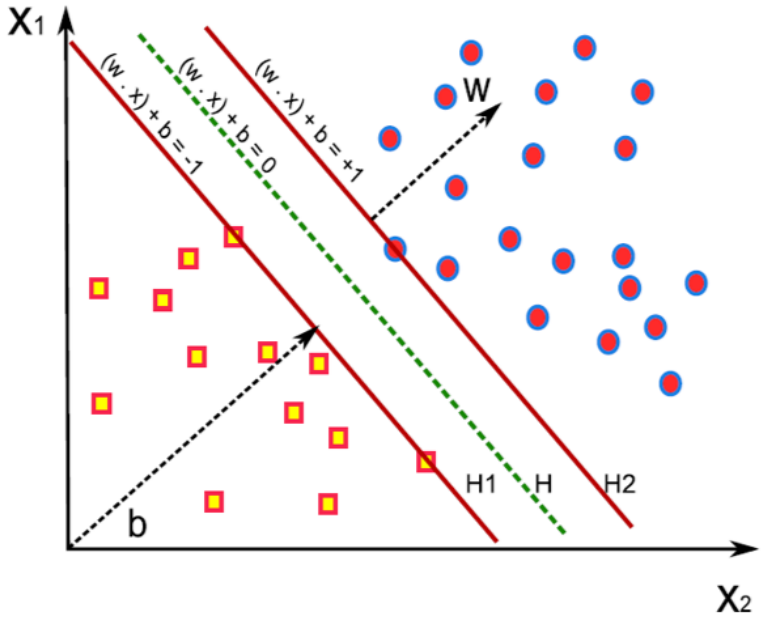
2.3.3 Support Vector Machines

Support Vector Machines (SVM) are a powerful and robust supervised classification algorithm. Vapnik [34] introduced SVM as a kernel-based machine learning model. SVM algorithm is based on finding an optimal line called hyperplane that maximizes the distance between classes.

According to a methodology by Cervantes et al. [4], the hyperplane that separates classes with maximum distance between them is defined in equation.

$$w^T x_i + b = 0 \tag{14}$$

where w is the weight vector, x is the vector of features, and b is the bias. In an example of linear classification, as 2 figure. shows, there are data points of positive class (in this case e-commerce companies, depicted as red circles) and negative class (in this case non-e-commerce companies, depicted as yellow cubes). The hyperplane with the maximum margin between classes is depicted as H in 2 figure. The lines that are adjacent to the optimal hyperplane (H1 and H2 in 2 figure.) are known as support vectors, as these vectors run through the data points that determine the maximal margin [14]. The distance between the hyperplane and closest data points from different classes to the hyperplane is called margin.



2 figure. *Optimal SVM classifier [4]*

According to a methodology by Cervantes et al. [4], the geometric margin of x^+ and x^- is defined as in following equation:

$$\gamma_i = \frac{1}{2}\left(\frac{\langle w, x^+\rangle}{\|w\|} - \frac{\langle w, x^-\rangle}{\|w\|}\right) \tag{15}$$

where:

- x^+ refers to the data point on the positive side of the hyperplane, which is closest to the hyperplane.
- x^- refers to the data point on the negative side of the hyperplane, which is closest to the hyperplane.

Simplifying this expression, we have:

$$\gamma_i = \frac{1}{2\|w\|}\left(\langle w, x^+\rangle - \langle w, x^-\rangle\right) \tag{16}$$

Thus, the margin can be expressed as:

$$\gamma_i = \frac{1}{\|w\|} \tag{17}$$

According to a methodology by Cervantes et al. [4], the objective of the SVM is to maximize the geometric margin while ensuring that all samples are correctly classified. This leads to the following optimization problem:

$$\begin{aligned} &\min_{w,b} \|w\|^2 \\ &\text{subject to:} \\ &y_i((w, x_i) + b) \geq 1, \quad \forall i \end{aligned} \tag{18}$$

where y_i is the actual value of class. We change (18) to the problem using the Lagrange formulation. The conditions given in equation (18) will be replaced by Lagrange multipliers, which are much easier to handle [4]. In this way, the Lagrangian is defined as following:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i((w \cdot x_i) + b - 1) \tag{19}$$

where α_i are the Lagrange's multipliers.

The SVM algorithm can be applied to both linear and nonlinear classification tasks. Much of the data in real-world scenarios is not linearly separable, and that is where nonlinear SVMs are useful. When data is not linearly separable, kernel functions are used. The kernel helps to reduce the complexity, making the computation more efficient [14]. When data is not linearly separable, kernel functions transform it to a higher-dimensional space to enable linear separation [4]. In this case, classifying companies as e-commerce and non-e-commerce, the algorithm tries to find the hyperplane in N-dimensional space. Several different kernels can be applied to classify data, such as the linear kernel, the Polynomial kernel, the Radial Basis Function (RBF) kernel and the Sigmoid kernel. The best accuracy results of website classification were achieved using the RBF kernel, which is defined as:

$$K(x, x') = \exp\left(-\gamma\|x - x'\|^2\right) \tag{20}$$

where the kernel's parameter γ is responsible for overfitting. When the value of γ increases, the probability of overfitting increases and vice versa. In this case, SVM uses the feature vectors derived from the TF-IDF representation of website texts to classify enterprises' websites into e-commerce and non-e-commerce. The SVM model's ability to handle large and sparse datasets and its effectiveness in high-dimensional spaces makes it suitable for websites classification task. The kernels allows to SVM to reduce the complexity, which provides flexibility and robustness to the model.

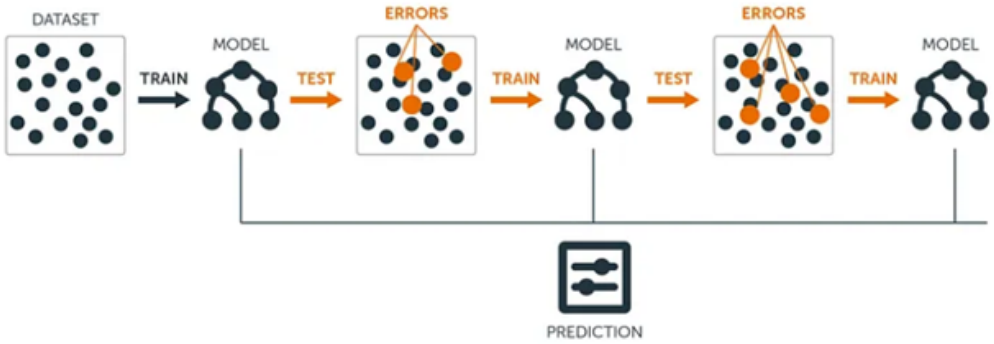
SVM algorithm is discussed in more detail by Cervantes et al. [4]. Python's sklearn library has been used for the SVM implementation. The features derived from TF-IDF were used for the classification model. Hyperparameters were fine-tuned to improve the performance of an SVM model. Optimal hyperparameters were found using grid search and cross-validation methods [4, 14].

2.3.4 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an efficient and scalable machine learning method which is suitable for text classification task like websites classification as e-commerce and non-e-commerce. Developed by Chen and Guestrin [6], XGBoost is an advanced method of gradient boosting with the same general framework. It combines weak learner trees into strong learners by adding up residuals. It is known for its speed, efficiency and ability to scale well with large datasets.

As Figure 3 shows, XGBoost, like other boosting methodologies, starts with a weak learner to make predictions. The first decision tree in gradient boosting is called the base learner. Next, new trees are created additively based on the base learner's mistakes. The algorithm then calculates the residuals (errors in a Figure 3) of each tree's predictions to determine how far the model's predictions

were from reality. Residuals are the difference between the model's predicted and actual values. The residuals are then aggregated to score the model with a loss function [15].



3 figure. *Extreme gradient boosting model schema*

According to Chen and Guestrin [6], XGBoost operates by iteratively training decision trees, where each subsequent tree minimizes the residual errors of its previous tree. The optimization objective for XGBoost includes a regularized loss function as follows:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{21}$$

where:

- $l(y_i, \hat{y}_i)$ is the loss function;
- $\Omega(f_k)$ is the regularization term to penalize model complexity;
- $\hat{y}_i$ represents predictions;
- f_k denotes the k-th decision tree.

The regularization term is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \|w\|^2 \tag{22}$$

where γ is complexity term, T is the classification's tree number of leaves, λ is penalty parameter and $\|w\|^2$ is the output of each leaf node.

Hyperparameters tuning is the optimization process of machine learning algorithms. Grid search and cross-validation methods are used to find the best hyperparameters training XGBoost model. Here is a description of hyperparameters for XGBoost decision trees:

- Learning rate determines how the boosting algorithm learns from each iteration. A lower value of learning rate means slower learning, as it scales down the contribution of each tree in the ensemble, therefore it helps prevent overfitting. A higher value of learning rate speeds up learning, but it may lead to overfitting. A value of learning rate should be between 0 and 1 [15].

- The number of trees to be built in the ensemble: each boosting round adds a new tree to the ensemble and the model slowly learns to correct the errors made by the previous trees. It specifies the complexity of the model and influences both the training time and the model's ability to perform on unseen data. Increasing the value of the number of trees typically increases the complexity of the model, as it allows the model to capture more sophisticated patterns in the data. However, adding too many trees can lead to overfitting [15].

- Gamma (also known as Lagrange multiplier) controls the minimum amount of loss reduction required to make a further split on a leaf node of the tree. A lower value means XGBoost stops earlier but may not find the best solution, while a higher value means XGBoost continues training longer, potentially finding better solutions, but with a higher risk of overfitting. There is no upper limit for gamma, but the values of gamma over 10 are considered high [15].

- Maximum depth represents how deeply each tree in the boosting process can grow during training. A tree's depth refers to the number of levels or splits it has from the root node to the leaf nodes. Increasing this value will make the model more complex and more likely to overfit [15].

XGBoost model is discussed in more detail by Chen and Guestrin [6]. The sparse matrix generated using the TF-IDF preprocessing method was used in XGBoost model training and testing. Python's xgboost library has been used for the XGBoost implementation.

2.3.5 BERT model

Bidirectional Encoder Representations from Transformers (BERT), is a powerful natural language processing (NLP) model developed by Google that uses a deep neural network architecture [25]. The BERT model architecture is based on a deep neural network called a transformer, which is different from traditional NLP models that process one word at a time. Transformers can process the entire text input all at once, which helps them to capture the relationships between words and phrases more effectively [25]. BERT uses a multi-layer bidirectional transformer encoder to represent the input text in a high-dimensional space [25]. Differently from other machine learning models in this thesis, BERT also looks at the words that come before and after for each word and this is the essence of bidirectional training. This context-awareness is a key aspect of why BERT usually outperforms traditional machine learning models [12]. BERT is a pre-trained model, which means it can be trained on massive amounts of text data, such as books, articles, and websites. BERT model can develop a deep understanding of the underlying structure and meaning of language. A pre-trained BERT can be fine-tuned for specific tasks, which allows it to adapt to the specific nuances of the task and improve its accuracy [25].

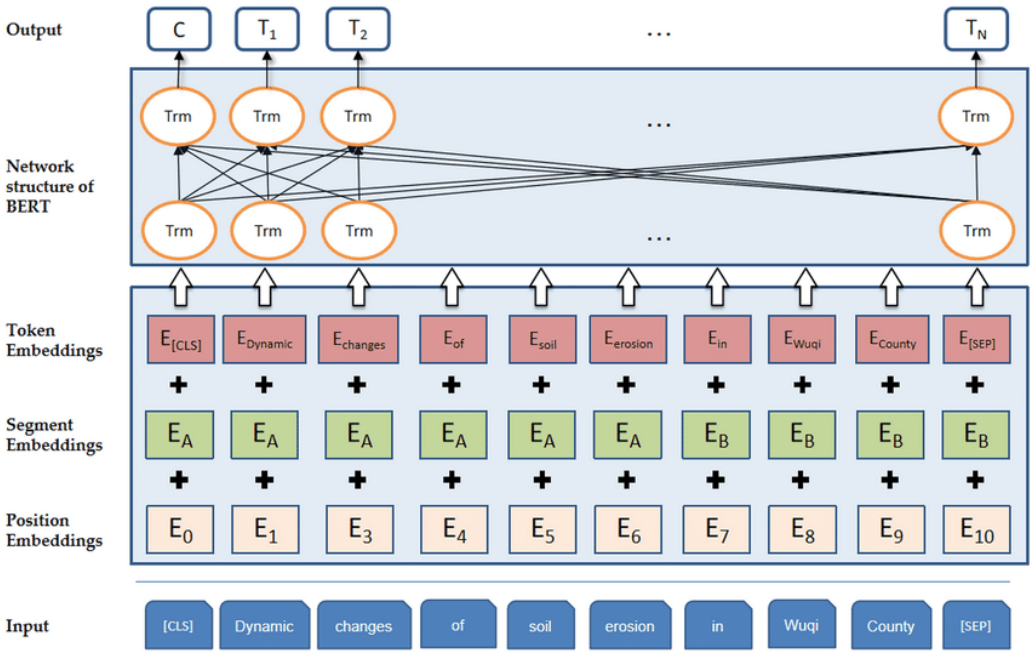
The overall structure of the BERT model is shown in Figure 4. According to the methodology of Pham [25], firstly, the text is being preprocessed and tokenized. The sparse matrix generated using TF-IDF is not used for BERT model, opposite as for other machine learnings models used in this thesis. Despite that, the text was normalized, stop words, numbers and punctuation marks were removed

from the text. Text should be tokenized specifically for BERT model. BERT requires specific formatting inputs, including special tokens CLS, which is added at the beginning of the text, and SEP, which is added at the end of the text for classification tasks [32]. After tokenization, features are extracted, BERT transforms tokens into embeddings. This process involves converting each token into a vector that represents both the token itself and its context within the sentence. These embeddings are dynamically refined through the layers of the BERT model [18]. When the text is converted into embeddings by the BERT encoder, these embeddings are passed to a classification layer. This layer typically consists of a few additional neural network layers. For text classification tasks, BERT model has the final hidden layer h [32]. A softmax classifier is added to the top of BERT to predict the probability of class c, as follows:

$$P(c|h) = \text{softmax}(Wh) \tag{23}$$

where:

- P is the probability of the class c given the hidden state h;
- c is the class;
- h is the final hidden layer output of the BERT model;
- W is the task-specific parameter matrix;
- softmax function transforms a previous layer's output into a probability.



4 figure. *The structure of BERT model [33]*

A pre-trained Bert model has been fine-tuned to classify websites as e-commerce or non-e-commerce companies. The base Bert model was used which contains an encoder with 12 transformer

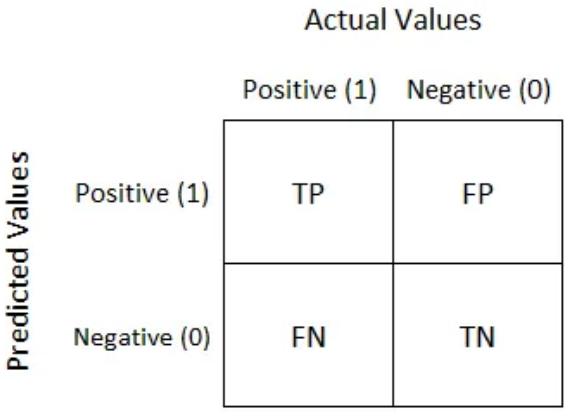
blocks, 12 self-attention heads, and the hidden size of 768. The model was fine-tuned using the text scraped from the companies' websites. Fine-tuning allowed the model to learn the unique language patterns and features of these e-commerce and non-e-commerce categories, increasing classification accuracy. To find the best hyperparameters for Bert model, grid search, and cross-validation methods are used. Here is a description of hyperparameters for Bert model:

- Learning rate: a lower value of learning rate means slower learning, therefore it helps prevent overfitting. A higher value of learning rate speeds up learning, but it may lead to overfitting [8].
- Epochs–the number of times the model has seen the entire dataset. If the number of epochs is too small, the model may not learn the underlying patterns in the data, resulting in underfitting. If the number of epochs is too large, the model may overfit the training data, leading to poor generalization performance on unseen data [8].
- Batch size–the number of training examples utilized in one iteration of model training. The batch size is a trade-off between accuracy and speed. Large batch size can lead to faster training times but may result in lower accuracy and overfitting, while smaller batch sizes can provide better accuracy, but can be computationally expensive and time-consuming [8].
- Warmup steps–the learning rate gradually increases to the initial specified learning rate over a number of warmup steps. This approach can help stabilize the model's training [35].
- Weight decay adds a regularization term to the loss function to prevent the weights from growing too large, which helps in controlling overfitting [35].

The methodology of the BERT model is presented by Pham [25] and Devlin [8]. Python's transformers library and PyTorch library were used for the BERT model implementation. Google Colab's GPU resources were utilized for Bert model training.

2.4 Performance evaluation metrics

This section describes metrics commonly used to evaluate the performance of text classification models. Performance measures generally evaluate specific aspects of classification task performance. Therefore, understanding what exactly each of these metrics represents and what kind of information they are trying to convey is crucial for comparability [19]. The performance of models will be evaluated and compared using the most common performance metrics in the analyzed related works, including accuracy, precision, recall, and F1 score. Confusion matrix, shown in Figure 5, is a performance measurement for machine learning classification. Let TP, FP, TN, FN denote true positive, false positive, true negative, and false negative, respectively.



5 figure. *Confusion matrix*

Accuracy is the most used performance metric for evaluating a binary classification model. It measures the proportion of correct predictions made by the model out of all the predictions, as denoted in equation (24). A high accuracy score indicates that the model is making a large proportion of correct predictions, while a low accuracy score indicates that the model is making too many incorrect predictions [19].

Precision is a metric that measures the proportion of true positive (TP) instances among the instances that are predicted as positive by the model, as denoted in equation (25). In other words, precision measures the accuracy of the positive predictions made by the model. In this case, a high precision score indicates that the model accurately identifies e-commerce enterprises. In contrast, a low precision score indicates that the model is making too many false positive (FP) predictions [19].

Recall, also known as sensitivity, is a performance metric that measures the proportion of positive instances that a model correctly identifies out of all the actual positive instances, as denoted in equation (26). Recall measures the proportion of true positive (TP) instances among all the actual positive instances. In this case, recall measures the model's ability to correctly identify e-commerce enterprises among all the actual e-commerce enterprises. A high recall score indicates that the model accurately identifies a large proportion of e-commerce enterprises [19].

F1 score is a performance metric that combines precision and recall, providing a comprehensive evaluation of a binary classification model's performance. It measures the harmonic mean of precision and recall, giving equal importance to both metrics, as denoted in equation (27). This metric balances the importance of precision and recall, and is preferable to accuracy for class-imbalanced datasets. The dataset of websites is unbalanced, because only 15% of enterprises are identified as e-commerce. Therefore, the F1 score is especially useful for model evaluation in this master's thesis [19].

$$accuracy = \frac{TP + TN}{N} \tag{24}$$

$$precision = \frac{TP}{TP + FP} \tag{25}$$

$$recall = \frac{TP}{TP + FN} \tag{26}$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{27}$$

where N is the total number of cases.

Performance evaluation metrics are calculated on testing set, when machine learning models are already trained. These metrics are crucial for evaluating the performance of classification models. Performance measures help to understand their strengths and weaknesses in classifying websites into e-commerce and non-e-commerce enterprises.

2.5 The Inverse Probability Weighting Estimator

Machine learning models have been trained to classify enterprises as e-commerce and non-e-commerce based on the text from their websites. However, the proportion of e-commerce enterprises in Lithuania cannot be calculated just by applying the trained models and classifying all enterprises in a dataset because not all enterprises' websites were scraped. We do not have scraped data on the whole population of enterprises in Lithuania. That is, there is only a non-probability sample, which has been used to estimate the proportion of e-commerce enterprises in Lithuania. It is a non-probability sample because the mechanism of how enterprises are included in a sample is unknown. According to Burakauskaitė and Čiginas [3], the inverse probability weighting (IPW) estimator has been applied to estimate the population proportion of e-commerce enterprises in Lithuania.

Let s_A be the non-probability sample and $R_k = I(k \in s_A)$ be the indicator variable for a unit $k \in \mathcal{U}$ selected to the sample s_A. The probabilities in equation (28) are called the propensity scores, where the subscript q refers to the propensity score model.

$$\pi_k^A = E_q(R_k \mid x_k) = P_q(R_k = 1 \mid x_k), \quad k \in \mathcal{U}, \tag{28}$$

where x_k represents the set of independent variables used in the logistic regression model to estimate propensity scores.

According to [3, 5], the non-probability sample itself does not represent the target population, and naive estimators based on it are typically biased. The main problem is the unknown selection mechanism for a unit to be included in the sample. Probabilities (28) are analogous to the inclusion into the sample probabilities π_k (for probability samples) since they describe the inclusion into the sample s_A. The propensity scores π_k^A, $k \in s_A$, need to be estimated before using them to weigh the units of the non-probability sample. In this case, the propensity score is the probability for a certain enterprise to be included in the sample. This approach allows to make more accurate inferences about the population based on the non-probability sample. IPW estimator can correct the sample selection bias efficiently if the propensity score model is well-specified.

We model the propensity scores $\pi_k^A = P_q(R_k = 1 \mid x_k)$ parametrically using the inverse logit function

$$\pi_k^A = \pi(x_k, \theta) = \frac{\exp(x_k'\theta)}{1 + \exp(x_k'\theta)}, \tag{29}$$

where θ is the model parameter with the unknown true value θ_0. The propensity scores $\hat{\pi}_k^A$ are modeled by the logistic regression model and obtained using the maximum likelihood method, where $\hat{\theta}$ maximizes the log-likelihood function

$$l(\theta) = \sum_{k \in s_A} \log\left(\frac{\pi(x_k, \theta)}{1 - \pi(x_k, \theta)}\right) + \sum_{k \in \mathcal{U}} \log(1 - \pi(x_k, \theta)) = \sum_{k \in s_A} x_k'\theta - \sum_{k \in \mathcal{U}} \log(1 + \exp(x_k'\theta)).$$

where s_A represents a non-probability sample from the population U. The maximum likelihood estimator $\hat{\theta}$ is found by solving the score equations as follows

$$U(\theta) = \frac{\partial}{\partial \theta} l(\theta) = \sum_{k \in \mathcal{U}} (R_k - \pi(x_k, \theta))\, x_k = 0.$$

The estimated propensity scores are given by

$$\hat{\pi}_k^A = \pi(x_k, \hat{\theta}), k \in S_A \tag{30}$$

The known population size is N. According to [3], the population parameter is computed using IPW estimator of the population proportion μ, which is given by

$$\hat{\mu}^{IPW} = \frac{1}{\hat{N}^A} \sum_{k \in s_A} \frac{y_k}{\hat{\pi}_k^A}, \quad \text{where} \quad \hat{N}^A = \sum_{k \in s_A} \frac{1}{\hat{\pi}_k^A}. \tag{31}$$

where y_k is the indicator of e-commerce, $\hat{N}_k^A$ is the estimated size of population. The variance of the estimator can be estimated by

$$\hat{v}^{IPW} = \frac{1}{(\hat{N}^A)^2} \sum_{k \in s_A} \left(1 - \hat{\pi}_k^A\right) \left(\frac{y_k - \hat{\mu}^{IPW}}{\hat{\pi}_k^A} - \hat{b}' x_k\right)^2, \tag{32}$$

where

$$\hat{b}' = \left\{\sum_{k \in s_A} \left(\frac{1}{\hat{\pi}_k^A} - 1\right)(y_k - \hat{\mu}^{IPW}) x_k'\right\} \left\{\sum_{k \in \mathcal{U}} \hat{\pi}_k^A (1 - \hat{\pi}_k^A) x_k x_k'\right\}^{-1},$$

given the non-probability sample s_A.

3 Results

In this section, the performance of classification models will be evaluated, including a discussion of the optimal hyperparameters for each model. Secondly, the application of the logistic regression model for propensity scores will be reviewed. Finally, the estimated proportion of e-commerce enterprises in Lithuania will be presented.

3.1 Hyperparameters

Machine learning models were trained to classify companies into e-commerce and non-e-commerce categories based on text data from their websites. Classification was coded in Python 3 programming language using Jupyter Notebook. Codes were run on Google Colab environment. Cross-validation and Grid search methods were used for hyperparameters optimization. The optimal hyperparameters were found for SVM, XGBoost and BERT models. Optimal hyperparameters are presented in Table 1.

SVM was fine-tuned using The Radial basis kernel function. According to the results, the most optimal parameters of SVM are gamma set to 0.05 and cost set to 10. These hyperparameters of SVM control the trade-off between the accuracy on the training data and the risk of overfitting. The best performance of XGBoost model was achieved when learning rate was 0.1, the number of trees was 200, maximum depth was set to 5 and gamma was at 0.1. Adam optimization algorithm was selected for BERT hyperparameters fine-tuning. The best performance of BERT model was achieved when the learning rate was set to 0.00002, the number of epochs was set to 5 with batches of 32. During the best performance of Bert model, warmup steps was 500 and weight decay was set to 0.05.

1 table. *Optimal hyperparameters*

Model	Hyperparameters	Optimal parameter
SVM	Gamma	0.05
	Cost	10
XGBoost	Learning rate	0.1
	Number of trees	200
	Maximum depth	5
	Gamma	0.1
BERT	Learning rate	0.00002
	Epochs	5
	Batch size	32
	Warmup steps	500
	Weight decay	0.05

3.2 Performance of classification models

Data was split into training and testing sets: 80% of enterprises were split into training set and 20% of them into testing set. Logistic regression, Naïve bayes, Support Vector Machines, Extreme Gradient Boosting and BERT models were trained on testing set and applied for a binary text classification task to categorize companies into e-commerce and non e-commerce companies. In this section, the performance of machine learning models is compared and the main findings are made. Table 2 provides a summary of each model's performance evaluation by essential metrics such as accuracy, precision, recall, and F1 score.

According to the results in Table 2, all models demonstrated high accuracy, but notable differences are in precision, recall and F1 score metrics. These differences are significant because the dataset is unbalanced. Only 15% of enterprises are e-commerce in a sample. This imbalance influences the models' performance, especially regarding recall and precision. High accuracy across all models might be misleading.

According to the results, BERT model showed the best performance among all applied models, achieving the highest recall (0.89), F1 score (0.89), accuracy (0.97) and the second best precision (0.90) after XGBoost. BERT shows exceptional balance across all metrics, making it highly reliable for both identifying e-commerce sites correctly and minimizing false positives. Its high recall and accuracy suggest that it effectively captures contextual nuances in text data. XGBoost model showed the second best performance, achieving the highest precision (0.93) among all trained models, which indicates its strength in correctly labeling true e-commerce sites. XGBoost showed the second best recall (0.81), F1 score (0.87) after BERT model and the same accuracy as Bert model (0.97). Naïve Bayes model showed better performance than SVM and logistic regression. According to the results, Naïve Bayes model achieved the precision of 0.89, recall of 0.78, F1 score of 0.83 and accuracy of 0.96. SVM showed the same accuracy (0.95) and precision (0.86) as logistic regression, but better recall (0.71) and F1 score (0.78) than logistic regression. Logistic regression showed the worst performance among all the models, struggling with recall (0.66) and F1 score (0.75).

2 table. *Performance of classification models*

Model	Precision	Recall	F1 score	Accuracy
BERT	0.90	0.89	0.89	0.97
XGBoost	0.93	0.81	0.87	0.97
Naïve Bayes	0.89	0.78	0.83	0.96
SVM	0.86	0.71	0.78	0.95
Logistic regression	0.86	0.66	0.75	0.95

According to the results, models perform better in classifying non-e-commerce companies than e-commerce companies, as seen by lower precision, recall, and F1 scores compared to accuracy across the models. BERT model is the most effective model for identifying enterprises as e-commerce and non-e-commerce companies.

3.3 Propensity scores

The logistic regression for propensity scores models the probability that an enterprise is included in a sample based on the enterprise's characteristics. The explanatory variables in the logistic regression are:

- Big city: enterprises located in Vilnius, Kaunas and Klaipėda are defined as enterprises from a big city.
- Income: enterprises are categorized by income into five equal size groups, where fifth group identifies the highest income level and first group identifies the lowest income level.
- Economic activity: enterprises are categorized into three different economic activities: IT, retail and other activity.
- Employees: enterprises are categorized by the number of employees into five equal size groups.

Two different logistic regressions were estimated for propensity scores model. These logistic regression models are given in Table 3. According to the results of the second model, if we make an assumption that the significance level is 0.05, enterprises located in not big cities are statistically significantly more likely to be scraped and included in the sample than companies located in big cities. Income level, from the second to the fourth group, excluding the highest income level, has a positive statistically significant impact on an enterprise to be scraped and included in a sample. Retail enterprises and enterprises from other economic activity are statistically significantly more likely to be included in a sample. The results showed that the number of employees does not significantly affect the likelihood of an enterprise being included in a sample. Therefore, this categorical explanatory variable was not included in the second model.

3 table. *Logistic regression models for propensity scores model*

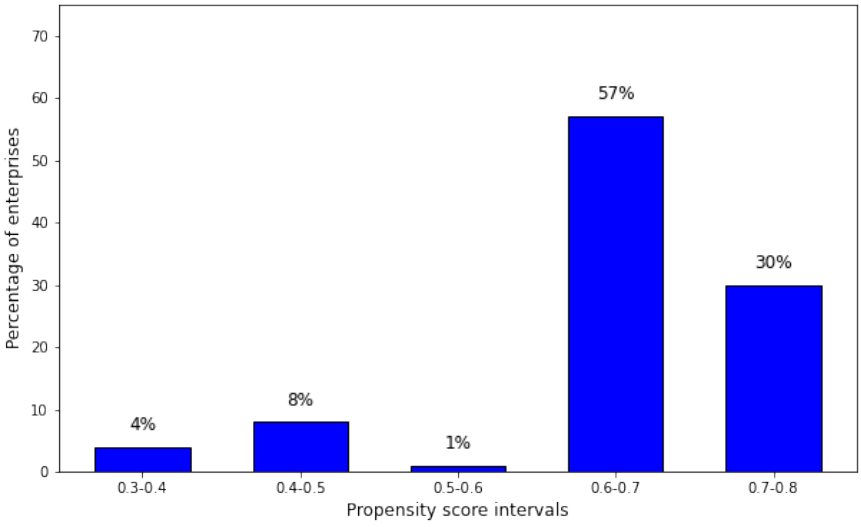
Variable	Model 1		Model 2	
	Coefficient	P-value	Coefficient	P-value
Intercept	-0.317	0.015*	-0.285	0.025*
Big city: No	0.0834	0.059	0.084	0.048*
Income: group 2	0.1664	0.021*	0.185	0.008*
Income: group 3	0.1524	0.043*	0.185	0.007*
Income: group 4	0.2391	0.003*	0.277	0.000*
Income: group 5	0.0859	0.338	0.114	0.089
Economic activity: Retail	1.099	0.000*	1.098	0.000*
Economic activity: Other	0.8433	0.000*	0.846	0.000*
Employees: 10-19	0.0861	0.225		
Employees: 20-49	0.0391	0.592		
Employees: 50-249	0.1079	0.167		
Employees: >250	0.0536	0.542		

According to the performance results in Table 4, the second model showed significantly better performance than the first model, achieving higher precision (0.77), recall (0.87), F1 score (0.82) and accuracy (0.85), classifying which companies are included into a sample and which are not. Performance results in Table 4 were used only to evaluate the performance of logistic regressions and decide which of them to apply in further work on the propensity scores model. The performance of logistic regressions for the propensity scores model in Table 4 is not related to enterprise classification models, whose performance results are presented in Table 2.

4 table. *Performance of logistic regression models for propensity scores model*

Model	Precision	Recall	F1 score	Accuracy
Model 1	0.64	0.69	0.66	0.69
Model 2	0.77	0.87	0.82	0.87

The propensity scores are calculated by equation (30) applying the second logistic regression model. The second logistic regression model was selected because it showed the better performance than the first model. Figure 6 shows the distribution of propensity scores. According to the histogram, 87% of companies have a propensity score between 0.6 and 0.8. The remaining part of enterprises has a propensity score between 0.3 and 0.6. As a conclusion, most enterprises tend to be scraped and included in the sample.



6 figure. *Propensity scores*

3.4 The proportion of e-commerce enterprises

According to the methodology, when enterprises are classified as e-commerce and non-e-commerce and propensity scores are calculated, the proportion of e-commerce enterprises in Lithuania can be estimated. IPW estimator has been applied to estimate the proportion of e-commerce enterprises in Lithuania. The estimated population proportion of e-commerce enterprises in Lithuania

is 0.25, and the estimate of standard deviation is 0.16. This estimated proportion is higher than the naive estimate of 15% calculated from only scraped enterprises.

In contrast, the State Data Agency [30] reports that 39% of Lithuanian enterprises were engaged in e-commerce in 2023. This higher result comes from a survey asking businesses about their online sales. According to the survey of usage of information technologies in enterprises, an e-commerce company has sales via the company's website, online stores, the company's internal website or mobile applications. My estimate that 25% of enterprises in Lithuania are e-commerce businesses is significantly lower than the 39% reported by Lithuania's official statistics. The main reason for the different statistics is the different definitions of e-commerce enterprises. In this master's thesis, an enterprise is considered e-commerce only if it has an e-shop on its main website. The definition of e-commerce company by the survey methodology is broader, including sales of any form of online sales, not only via the website but also online stores, the company's internal website or mobile applications. This difference in definitions explains why study's findings show a lower proportion of e-commerce enterprises compared to the official statistics.

4 Conclusions and recommendations

The main goal of this master's thesis to estimate the proportion of e-commerce enterprises in Lithuania was achieved. In the process, all companies were manually classified as e-commerce and non-e-commerce based on their websites to have actual values of companies' e-commerce status for models training and performance testing. Companies' websites were scraped to collect the text from them. Logistic regression, Naïve Bayes, Support Vector Machines, Extreme Gradient Boosting, and BERT models were trained on training set to classify enterprises as e-commerce and non-e-commerce based on the extracted text from their websites. The inverse probability weighting estimator was applied to adjust for non-random sampling of the websites and estimate the proportion of e-commerce enterprises in Lithuania. The following conclusions were made:

- The estimated proportion of e-commerce enterprises in Lithuania is 25%, which is higher than the naive estimate of 15% calculated from only scraped enterprises. The estimated proportion of e-commerce enterprises significantly differs from 39% reported by Lithuania's official statistics. The main reason for that discrepancy is the different definitions of e-commerce companies. In this study, an e-commerce company was defined as a company that has an e-shop on its main website, while the methodology of the survey also includes companies which sell via online stores, the company's internal website or mobile applications. The application of the inverse probability weighting estimator helped to solve the problems of non-random sampling and estimate the population proportion of e-commerce enterprises in Lithuania.
- The BERT model showed the best performance classifying enterprises as e-commerce and non-e-commerce enterprises, achieving the highest recall, F1 score and accuracy among all trained models. The Extreme Gradient Boosting model showed the second-best performance after the BERT model. Machine learning algorithms combined with NLP methods are a powerful and scalable tool for companies classification as e-commerce and non-e-commerce companies.

The recommendations for future research are:

- Train models and classify enterprises using data not only from enterprises' websites but also from online stores and mobile applications.
- Apply additional statistical methods alongside the inverse probability weighting estimator. Comparing various estimators can provide a deeper understanding of how different approaches impact the outcomes. It can help identify the most accurate and reliable method for estimating the proportion of e-commerce enterprises.

This thesis contributes to the academic literature on applying machine learning to real-world classification problems, specifically in the context of e-commerce companies identification. The developed methodology, which integrates web scraping, machine learning models, and the adjustment of the non-probability sample, could serve as a framework for other countries or regions, estimating the proportion of e-commerce enterprises.

References and sources

[1] G. Barcaroli, G. Bianchi, R. Bruni, A. Nurra, S. Salamone, M. Scarnò. "Machine learning and statistical inference: the case of Istat survey on ICT." In: *Proceeding of 48th scientific meeting of the Italian Statistical Society SIS*. Salerno, Italy. 2016.

[2] G. Barcaroli, A. Nurra, S. Salamone, M. Scannapieco, M. Scarnò, D. Summa. "Internet as data source in the istat survey on ICT in enterprises." In: *Austrian Journal of Statistics* 44.2 (2015), pages 31–43.

[3] I. Burakauskaitė, A. Čiginas. "An approach to integrating a non-probability sample in the population census." In: *Mathematics* 11.8 (2023), pages 1782–1795.

[4] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez. "A comprehensive survey on support vector machine classification: Applications, challenges and trends." In: *Neurocomputing* 408 (2020), pages 189–215.

[5] Y. Chen, P. Li, C. Wu. "Doubly robust inference with nonprobability survey samples." In: *Journal of the American Statistical Association* 115.532 (2020), pages 2011–2021.

[6] T. Chen, C. Guestrin. "Xgboost: A scalable tree boosting system." In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pages 785–794.

[7] F. De Fausti, F. Pugliese, D. Zardetto. "Towards automated website classification by deep learning." In: *arXiv preprint arXiv:1910.09991* (2019).

[8] J. Devlin. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[9] Eurostat. *E-commerce statistics*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=E-commerce_statistics (viewed 2024-11-27).

[10] Eurostat. *Glossary: E-commerce*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:E-commerce&oldid=49208 (viewed 2024-11-27).

[11] Eurostat. *ICT in Households and by Individuals*. URL: https://ec.europa.eu/eurostat/cache/infographs/ict_2017/bloc-2b.html (viewed 2024-11-27).

[12] S. González-Carvajal, E. C. Garrido-Merchán. "Comparing BERT against traditional machine learning text classification." In: *arXiv preprint arXiv:2005.13012* (2020).

[13] A. Gupta, R. Bhatia. "Ensemble approach for web page classification." In: *Multimedia Tools and Applications* 80.16 (2021), pages 25219–25240.

[14] IBM. *Support Vector Machine*. URL: https://www.ibm.com/topics/support-vector-machine (viewed 2024-11-26).

[15] IBM. *XGBoost: Gradient Boosting for Machine Learning*. URL: https://www.ibm.com/topics/xgboost (viewed 2024-11-26).

[16] G. James, D. Witten, T. Hastie, R. Tibshirani, et al. *An introduction to statistical learning*. Volume 112. Springer, 2013.

[17] D. Jurafsky. *Speech and language processing (3rd ed.)* 2019.

[18] J. D. M.-W. C. Kenton, L. K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *Proceedings of naacL-HLT*. Volume 1. Minneapolis, Minnesota. 2019, page 2.

[19] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown. "Text classification algorithms: A survey." In: *Information* 10.4 (2019), page 150.

[20] C. D. Manning. *Introduction to information retrieval*. 2008.

[21] S. Markkandeyan, M. Indra Devi. "Efficient machine learning technique for Web page classification." In: *Arabian Journal for Science and Engineering* 40 (2015), pages 3555–3566.

[22] G. Matošević, J. Dobša, D. Mladenić. "Using machine learning for web page classification in search engine optimization." In: *Future Internet* 13.1 (2021), page 9.

[23] M. M. Mirończuk, J. Protasiewicz. "A recent overview of the state-of-the-art elements of text classification." In: *Expert Systems with Applications* 106 (2018), pages 36–54.

[24] Mordor Intelligence. *Lithuania E-commerce Market - Growth, Trends, and Forecasts (2024 - 2029)*. URL: https://www.mordorintelligence.com/industry-reports/lithuania-ecommerce-market (viewed 2024-11-27).

[25] K. Pham. *Text Classification with BERT*. URL: https://medium.com/@khang.pham.exxact/text-classification-with-bert-7afaacc5e49b (viewed 2024-11-27).

[26] A. Rajaraman. *Mining of massive datasets*. 2011.

[27] K. I. Roumeliotis, N. D. Tselikas, D. K. Nasiopoulos. "Unmasking Misinformation: Leveraging CNN, BERT, and GPT Models for Robust Fake News Classification." In: (2024).

[28] S. S. Shaffi, I. Muthulakshmi. "Search engine optimization by using machine learning for Web page classification." In: *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. IEEE. 2022, pages 342–349.

[29] M. Somesha, A. R. Pais, R. S. Rao, V. S. Rathour. "Efficient deep learning techniques for the detection of phishing websites." In: *Sādhanā* 45 (2020), pages 1–18.

[30] Statistics Lithuania. *Analysis of Statistical Indicators*. 2023. URL: https://osp.stat.gov.lt/statistiniu-rodikliu-analize?hash=3323ad50-fee5-4d46-b02b-4db05fb42104#/ (viewed 2024-10-05).

[31] Statistics Lithuania. *Skaitmeninė ekonomika ir visuomenė Lietuvoje 2022*. URL: https://osp.stat.gov.lt/skaitmenine-ekonomika-ir-visuomene-lietuvoje-2022/e-komercija/e-prekyba-imonese (viewed 2024-11-27).

[32] C. Sun, X. Qiu, Y. Xu, X. Huang. "How to fine-tune bert for text classification?" In: *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*. Springer. 2019, pages 194–206.

[33] J. Sun, Y. Liu, J. Cui, H. He. “Deep learning-based methods for natural hazard named entity recognition.” In: *Scientific reports* 12.1 (2022), page 4598.

[34] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[35] A. Vaswani. “Attention is all you need.” In: *Advances in Neural Information Processing Systems* (2017).

Appendix 1. Python code

```
# packages
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

#import data
df = pd.read_csv("df_websites.csv")

### Data preprocessing
# Cleaning function
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuations and numbers
    text = ''.join(e for e in text if e.isalnum() or e.isspace())
    return text

# Clean the text column
df['text'] = df['text'].apply(clean_text)

# Splitting data into training and testing sets
X = df['text']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

### Features extraction using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english',
max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

### Models training
## Logistic Regression
clf = LogisticRegression()
clf.fit(X_train_vec, y_train)
y_pred = clf.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n",
```

```
classification_report(y_test, y_pred))

## Naive Bayes
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.naive_bayes import MultinomialNB

# K-Fold cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Model Training & Hyperparameter Tuning
param_grid = {
    'alpha': [0.01, 0.1, 1.0, 10.0]  # Smoothness parameter
}
nb = MultinomialNB()
grid_search = GridSearchCV(estimator=nb, param_grid=param_grid,
     cv=cv, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search.fit(X_train_vec, y_train)

# Best parameters
best_params = grid_search.best_params_
print(f"Best Parameters: {best_params}")

# Evaluation
best_nb = grid_search.best_estimator_
y_pred = best_nb.predict(X_test_vec)
print("Classification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")

## Support vector machine
from sklearn.svm import SVC

# Training and Hyperparameter Tuning
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'degree': [1, 2, 3, 4],
    'gamma': ['scale', 'auto']
}

svc = SVC(random_state=42)
grid_search = GridSearchCV(estimator=svc, param_grid=param_grid,
cv=3, n_jobs=-1, verbose=2, scoring='accuracy')
```

```
grid_search.fit(X_train_vec, y_train)

# Best parameters
best_params = grid_search.best_params_
print(f"Best Parameters: {best_params}")

# Evaluation
best_svc = grid_search.best_estimator_
y_pred = best_svc.predict(X_test_vec)
print("Classification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")

## XGBoost
import xgboost as xgb

# Training and Hyperparameter Tuning
param_grid = {
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5, 7],
    'n_estimators': [100, 200],
    'subsample': [0.7, 0.9],
    'colsample_bytree': [0.6, 0.8]
}

xgb_model = xgb.XGBClassifier(use_label_encoder=False,
eval_metric='logloss', random_state=42)
grid_search = GridSearchCV(estimator=xgb_model,
param_grid=param_grid, cv=3, n_jobs=-1,
verbose=2, scoring='accuracy')
grid_search.fit(X_train_vec, y_train)

# Best parameters
best_params = grid_search.best_params_
print(f"Best Parameters: {best_params}")

# Evaluation
best_xgb = grid_search.best_estimator_
y_pred = best_xgb.predict(X_test_vec)
print("Classification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
## BERT model
from transformers import BertTokenizer,
from transformers import BertForSequenceClassification
from transformers import Trainer, TrainingArguments
import torch
from torch.utils.data import Dataset
import numpy as np
from transformers import EarlyStoppingCallback

# Split data into training and validation sets
train_df, val_df = train_test_split(df, test_size=0.2, random_state=42)

# Initialize the tokenizer
tokenizer = BertTokenizer.from_pretrained(
'bert-base-multilingual-cased')

# Tokenization function
#def tokenize_function(examples):
 #   return tokenizer(examples['text'], padding="max_length",
     truncation=True,
     return_tensors="pt")

# Tokenization function that receives a series of text
def tokenize_function(text_series):
    return tokenizer(text_series.tolist(),
    padding="max_length", truncation=True,
    return_tensors="pt")

# Tokenize the text column of the DataFrame
train_encodings = tokenize_function(train_df['text'])
val_encodings = tokenize_function(val_df['text'])

# Convert labels to tensors
train_labels = torch.tensor(train_df['label'].values)
val_labels = torch.tensor(val_df['label'].values)

# Dataset class
class LithuanianTextDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
```

```
        item = {key: val[idx] for key, val
        in self.encodings.items()}
        item['labels'] = self.labels[idx]
        return item

    def __len__(self):
        return len(self.labels)

# Create datasets
train_dataset = LithuanianTextDataset(train_encodings,
train_labels)
val_dataset = LithuanianTextDataset(val_encodings, val_labels)

# Load a pre-trained BERT model and run it on GPU
model = BertForSequenceClassification.from_pretrained(
'bert-base-multilingual-cased')
model.to(torch.device('cuda' if torch.cuda.is_available()
else 'cpu'))

# Training arguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    evaluation_strategy="steps",
    eval_steps=500,
    save_strategy="steps",
    save_steps=500,
    load_best_model_at_end=True,
    metric_for_best_model='loss',
    greater_is_better=False,
)

# Initialize Trainer with EarlyStoppingCallback
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
```

```
    eval_dataset=val_dataset,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)]
    # stop if no improvement after 3 evals
)

# Train the model
trainer.train()

# Save the model
model.save_pretrained('bert-finetuned-lithuanian')

# Make predictions on the validation set
predictions = trainer.predict(val_dataset)
pred_labels = np.argmax(predictions.predictions, axis=1)

# Generate a classification report
report = classification_report(val_df['label'],
pred_labels, target_names=['Not Eshop', 'Eshop'])
print(report)

### IPV estimator
## Logistic regression for propensity scores
import statsmodels.api as sm
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Prepare the feature matrix and target vector
X = df[['city_region','income', 'sector']]
y = df['scraped']

# Create dummy variables
column_transformer = ColumnTransformer(
    [('cat', OneHotEncoder(drop='first'), ['city_region','income', 'sector'])],
    remainder='passthrough'
)

# Apply the transformer to the feature matrix
X_transformed = column_transformer.fit_transform(X)

# Convert the transformed features into a dataframe
if hasattr(X_transformed, 'toarray'):
    X_transformed_df = pd.DataFrame(X_transformed.toarray(),
    columns=column_transformer.get_feature_names_out())
```

```
else:
    X_transformed_df = pd.DataFrame(X_transformed,
    columns=column_transformer.get_feature_names_out())

X_transformed_df = sm.add_constant(X_transformed_df)

# Logistic regression
model = sm.Logit(y, X_transformed_df)
result = model.fit()
print(result.summary())

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
   random_state=42)

# Apply the transformer to the training feature matrix
X_train_transformed = column_transformer.fit_transform(X_train)
X_test_transformed = column_transformer.transform(X_test)


# Convert the transformed features into a dataframe
if hasattr(X_train_transformed, 'toarray'):
    X_train_transformed_df = pd.DataFrame(X_train_transformed.toarray(),
    columns=column_transformer.get_feature_names_out())
else:
    X_train_transformed_df = pd.DataFrame(X_train_transformed, columns=
       column_transformer.get_feature_names_out())


# Reset indices for both X and y to ensure alignment
X_train_transformed_df = X_train_transformed_df.reset_index(drop=True)
y_train = y_train.reset_index(drop=True)
# Add a constant to the model
X_train_transformed_df = sm.add_constant(X_train_transformed_df)

# Initialize and fit the Logistic Regression model using statsmodels
model = sm.Logit(y_train, X_train_transformed_df)
result = model.fit()

# Prepare the test data (adding a constant column)
\begin{lstlisting}
if hasattr(X_test_transformed, 'toarray'):
    X_test_transformed_df = pd.DataFrame(X_test_transformed.toarray(),
       columns=column_transformer.get_feature_names_out())
```

```
else:
    X_test_transformed_df = pd.DataFrame(X_test_transformed, columns=
       column_transformer.get_feature_names_out())
X_test_transformed_df = sm.add_constant(X_test_transformed_df)
X_test_transformed_df = X_test_transformed_df.reset_index(drop=True)

# Predicting on testing data
y_pred = result.predict(X_test_transformed_df)
y_pred_class = (y_pred > 0.5).astype(int)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred_class)
conf_matrix = confusion_matrix(y_test, y_pred_class)
report = classification_report(y_test, y_pred_class)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", report)

## Propoensity scores
results = model.fit()

# Model parameters
params = results.params
intercept = params[0]
coefficients = params[1:]

logit_scores = np.dot(X_dense, coefficients) + intercept
propensity_scores = np.exp(logit_scores) / (1 + np.exp(logit_scores))
df['propensity_scores'] = propensity_scores

## Population parameter
# Calculate the weights for each observation
weights = 1 / df['propensity_scores']

# The weighted sum of y_k
weighted_sum = np.sum(df['ecommerce'] / df['propensity_scores'])

# The total weight for normalization
total_weight = np.sum(weights)

# Calculate the IPW estimator
mu_ipw = weighted_sum / total_weight
```

```
print("Estimated population mean (mu^IPW):", mu_ipw)

## Variance of estimator
# Sample data extraction
S = df[df['scraped'] == 1]
S = S[S['ecommerce','city_region','income', 'sector','propensity_scores
   ']]
S_encoded = pd.get_dummies(S, columns=['city_region','income', 'sector'])
x_columns = [col for col in S_encoded.columns if col not in ['ecommerce
   ','city_region','income', 'sector','propensity_scores']]
x = S_encoded[x_columns].values
x = np.array(x, dtype=float)

weights_diff = np.array(1 / S['propensity_scores'] - 1)
ecommerce_diff = np.array(S['ecommerce'] - mu_ipw)

# Reshape data
weights_diff = weights_diff[:, np.newaxis]
ecommerce_diff = ecommerce_diff[:, np.newaxis]

# Calculating weighted residuals
weighted_residuals = weights_diff * ecommerce_diff * x
# Sum of weighted residuals
sum_weighted_residuals = np.sum(weighted_residuals, axis=0)
# Calculating the sum of covariance
propensity_scores = np.array(S['propensity_scores'])
sum_covariance = x.T @ (propensity_scores[:, np.newaxis]
* (1 - propensity_scores[:, np.newaxis]) * x)  # Matrix multiplication
# Invert the sum covariance matrix and multiply by the sum of weighted
   residuals
b_prime = np.linalg.inv(sum_covariance) @ sum_weighted_residuals

# Calculate residuals and variance
residuals = (ecommerce_diff.flatten() / propensity_scores) - x @ b_prime
V_ipw = np.sum((1 - propensity_scores) * residuals**2)
/ (np.sum(1 / propensity_scores)**2)
print(V_ipw)
```

Appendix 2. Usage of AI

Grammarly program was used to check spelling, punctuation and increase readability.