VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistro baigiamasis darbas

Kintamumo prognozavimas panaudojant rekurentines diagramas ir giliojo mokymosi metodus

Volatility forecasting using recurrence plots and deep neural networks

Atliko:
Vaidas Karvelis parašas

Vadovas:
prof. dr. Tadas Meškauskas

Vilnius
2024

Turinys

Sutartinis terminų žodynas **4**

Santrauka **5**

Summary **6**

Įvadas **7**

1. Literatūros analizė **9**

2. Naudojamų algoritmų teorija **10**
2.1. Kintamumas . . . 10
2.2. Rekurentinės diagramos . . . 10
2.2.1. Signalo būsenos . . . 10
2.2.2. Rekurentinių diagramų apibrėžimas . . . 10
2.2.3. Slenkstinio parametro pasirinkimas . . . 11
2.2.4. Normos pasirinkimas . . . 11
2.2.5. Struktūrų interpretavimas . . . 11
2.2.6. Neapibrėžtosios rekurentinės diagramos . . . 12
2.3. RQA . . . 12
2.3.1. Slenkančio lango principas . . . 14
2.4. Prižiūrimas mokymasis . . . 15
2.4.1. Duomenų padalinimas . . . 15
2.4.2. Regresija . . . 17
2.4.3. Klasifikavimas . . . 17
2.5. Giliojo mokymosi metodai . . . 19
2.5.1. Gilieji neuroniniai tinklai . . . 19
2.5.2. Aktyvacijos funkcijos . . . 19
2.5.3. Tikslo funkcijos . . . 20
2.5.4. Konvoliuciniai neuroniniai tinklai . . . 20
2.5.5. Liekanų neuroninis tinklas . . . 22
2.5.6. Žinių perdavimas . . . 23
2.5.7. Rekurentiniai neuroniniai tinklai . . . 24
2.5.8. Ilgos trumpalaikės atminties modelis . . . 24
2.6. Garch modelis . . . 26

3. Programinės realizacijos detalės **27**

4. Duomenys ir jų paruošimas **29**
4.1. Trūkstamų reikšmių paieška . . . 29
4.2. Kintamumo apskaičiavimas . . . 29
4.3. Nuokrypiai . . . 30
4.4. Kategorijos . . . 30
4.5. Rekurentinės diagramos . . . 32

5. Modelių apmokymo aspektai **33**

5.1. Konvoliuciniai neuroniniai tinklai 33
5.1.1. Įvestis ir žymėjimas 33
5.1.2. Įvesties pasirinkimas 34
5.1.3. Duomenų padalinimo pasirinkimas 38
5.1.4. Duomenų augmentacija 39
5.1.5. Apmokymas 39
5.2. Rekurentiniai neuroniniai tinklai 41
5.3. Garch modeliai 41

6. Rezultatai **42**
6.1. Modelių įvertinimas 42
6.2. Modelių įvertinimas su paklaida 45

Išvados ir rekomendacijos **47**

Ateities tyrimų gairės **48**

Literatūros šaltiniai **49**

Priedai **52**

A. Kintamumo prognozavimo su leidžiama paklaida vertinimo metrikos **52**

Sutartinis terminų žodynas

1. Kintamumas (angl. *volatility*) – statistinis dydis žymintis signalo polinkį greitai ir nenuspėjamai keistis.
2. Rekurentinės diagramos (angl. *recurrence plots*) – signalų analizės algoritmas, skirtas lengviau suprasti signalus.
3. RQA (angl. *recurrence quantification analysis*) – rinkinys metrikų skirtų išgauti rekurentinių diagramų informaciją.
4. MSE (angl. *mean squared error*) – vidutinė kvadratinė paklaida.
5. ACC (angl. *accuracy*) – tikslumas.
6. PREC (angl. *precision*) – precizija.
7. REC (angl. *recall*) – atkuriamumas.
8. F1 (angl. *F1 score*) – F1 rezultatas.
9. Aktyvacijos funkciją (angl. *activation function*) – dirbtinių neuroninių tinklų parametras.
10. ReLU (angl. *Rectified Linear Unit*) – viena iš populiariausių pastarojo dešimtmečio aktyvacijos funkcijų.
11. RNN (angl. *Recurrent neural networks*) – rekurentiniai neuroniniai tinklai.
12. CNN (angl. *Convolutional neural networks*) – konvoliuciniai neuroniniai tinklai.
13. GARCH (angl. *Generalized AutoRegressive Conditional Heteroskedasticity*) – statistinis modelis, kuris yra plačiai naudojamas modeliuoti kintamumą laiko eilutėse.
14. ResNet (angl. *ressidual neural network arba ResNet*) – Liekanų neuroninis tinklas.
15. Persimokymas (angl. Overfitting) – modelio prisitaikymas prognozuoti tik matytus duomenis, taip nesugebant prognozuoti nematytų duomenų.

Santrauka

Šiame darbe buvo tiriamas SP500 akcijų kainų kintamumo prognozavimas, panaudojant rekurentines diagramas ir giliojo mokymosi metodus. Darbo metu buvo sudarytos spalvinės ir klasikinės rekurentinės diagramos, apskaičiuotos RQA metrikos. Tada pasinaudojant rekurentinėmis diagramomis, RQA ir akcijų kainų kintamumo laiko eilutėmis buvo apmokyti GARCH, RNN, ResNet ir LSTM tipo modeliai. Apmokyti modeliai prognozavo intervalą kuriame bus kintamumas po 1, 2, 5, 10 ir 21 dienos. Gauti rezultatai parodė, kad artimą kintamumą prognozuoja tiksliausiai RNN tipo modeliai, o kintamumą po 21 dienos tiksliausiai prognozuoja GARCH ir ResNet tipo modeliais.

Summary

Volatility forecasting using recurrence plots and deep neural networks

Market volatility forecasting has been an important research topic in recent decades. This paper aims to forecast realized market volatility for SP500 stocks by using recurrence plots, RQA and deep neural networks.

The experiments done in this paper were based on analysis done for daily stock market price changes. Logarithmic returns were calculated from daily closing stock prices which then were used to generate recurrence plots and RQA metrics. The resulting data was used to train GARCH, RNN, LSTM and ResNet models. These models were trained to forecast intervals in which stock market volatility should fall after 1, 2, 5, 10 and 21 days. Obtained results showed that RNN models give the best results when forecasting volatility after 1, 2, 5 and 10 days, while ResNet and GARCH models give better results for predictions after 21 days.

Įvadas

Visais laikais žmonės siekė nuspėti ateitį ir tai atlikdavo įvairiais būdais. Pavyzdžiui, senovės Kinijos civilizacijos siekdamos nuspėti mūšių baigtis, derliaus dydį ar lietaus tikimybę degindavo kaulus [15]. Ūkininkai planuodavo veiklą stebint gyvūnų elgseną, atmosferos reiškinius ar augalų pokyčius. Šiais laikais žmonės irgi bando prognozuoti ateitį, tačiau tai daro naudojant įvairius matematinius modelius, paremtus informacija sukaupta praeityje.

Viena iš šių laikų sričių, kurioje yra reikalingas prognozavimas, yra investavimo rizikos valdymas. Žmonės investuoja pinigus į akcijas, tikėdamiesi ateityje atgauti didesnį kiekį pinigų. Tačiau, ne visos investicijos yra saugios, todėl yra būtina, gebėti tinkamai pasirinkti investicijas. Siekdami tinkamai atsirinkti akcijas, į kurias verta investuoti, investuotojai bandydavo prognozuoti akcijų kainų dydžius ateityje. Tačiau, pastovių sėkmės istorijų šia tema literatūroje nėra daug. Dėl šios priežasties žmonės ėmė prognozuoti kitą dydį – kintamumą. Šis dydis yra svarbus investuotojams atsirenkant saugias investicijas. Yra laikoma, kad akcijos turinčios didesnį kintamumą yra rizikingesnės už akcijas su mažesniu kintamumu. Be to, šio dydžio prognozavimas yra paprastesnis, nes kintamumas nusako tik kainos svyravimo intensyvumą, o ne konkrečia reikšmę tam tikru metu.

Bet kokiam prognozavimui, naudojant matematinius modelius, reikia duomenų. Tokių duomenų pavyzdys galėtų būti akcijų kainų pokyčiai surinkti tam tikru metu. Akcijų kainų pokyčiai sudaro laiko eilutes (angl. *time series*). Pačios laiko eilutės ne visada yra informatyvios, todėl siekiant ištraukti prasmingą informaciją yra naudojami signalų analizės ir apdorojimo algoritmai. Vienas iš tokių algoritmų yra rekurentinės diagramos (angl. *recurrence plots*). Šis algoritmas padeda suprasti sistemų atsikartojimus ir būsenas laiko eilutėse, todėl gali būti naudingas prognozuojant kintamumą.

Kitas svarbus pasirinkimas kuriant modelius prognozuojančius kintamumą, yra matematinio modelio architektūros pasirinkimas. Literatūros šaltiniuose yra gausu informacijos apie giliųjų neuroninių tinklų (angl. *deep neural networks*) sėkmę atliekant įvairius prognozavimo uždavinius, kaip kad kintamumo prognozavimą.

Atlikus susijusių darbų analizę, nebuvo rasta darbų tiriančių kintamumo prognozavimą, naudojant rekurentines diagramas ir giliuosius neuroninius tinklus. Šio **darbo tikslas** yra ištirti rekurentinių diagramų panaudojimą, siekiant prognozuoti kintamumą, naudojant giliojo mokymosi metodus. Siekiant pagrindinio darbo tikslo buvo suformuluoti šie **uždaviniai**:

- paruošti modelių apmokymui ir kintamumo prognozavimui reikalingus duomenis (4 skyrius),
- apmokyti GARCH modelius (5.3 skyrelis),
- apmokyti konvoliuciniais neuroniniais tinklais paremtus modelius (5.1 skyrelis),
- apmokyti rekurentiniais neuroniniais tinklais paremtus modelius (5.2 skyrelis),
- įvertinti modelių prognozavimo kokybę (6 skyrius).

Pirmame skyriuje yra atliekama susijusių darbu apžvalga. Antrame skyriuje aprašoma teorija reikalinga kintamumo skaičiavimo modelių kūrimui: kintamumo skaičiavimo apibrėžimas, rekurentinės diagramos ir jų savybės, atsikartojimų kiekybinė analizė, slenkančio lango principas, prižiūrimojo mokymosi (angl. *supervised learning*) uždaviniai, jų vertinimo metrikos, giliojo mokymosi metodai ir GARCH modelis. Trečiame skyriuje yra detaliai aprašoma programos realizacija ir naudotos bibliotekos. Ketvirtas skyrius yra skirtas aprašyti darbe naudotus duomenis ir jų

paruošimui atliktus veiksmus. Penktame skyriuje modelių apmokymo detalės. Šeštame skyriuje aprašyta kaip buvo vertinama modelių kokybė. Darbo pabaigoje pateikiamos išvados ir rekomendacijos bei ateities tyrimų gairės.

Šis darbas yra testinis ir dalis informacijos buvo paimta iš praėjusiame semestre buvusio „Mokslo tiriamojo darbo projekto“. Paimta informacija:

- Dalis įvado.
- Dalis literatūros analizės (1 skyrius).
- 2.1. „Kintamumas“.
- 2.2. „Rekurentinės diagramos“.
- 2.3. „RQA“.
- 2.4. „Prižiūrimas mokymasis“ (išskyrus 2.4.1 „Duomenų padalinimas“).
- 2.5.1. „Gilieji neuroniniai tinklai“.
- 2.5.3. „Tikslo funkcijos“.

Tiksliam giliojo mokymosi metodų sąvokų vertimui buvo naudojamas dirbtinio intelekto sąvokų žodynėlis [30].

1. Literatūros analizė

Kintamumo prognozavimas yra svarbi tema aktuali įvairiose srityse, kaip kad medicinoje ar finansuose. Ši tema yra aktyviai tiriama, todėl yra gausu literatūros šaltinių aprašančių kintamumo prognozavimo ir jų tikslumo vertinimo būdus.

Christian L. Dunis ir Xuehuan Huang [16] savo darbe rašė, kaip galima prognozuoti valiutų kursų kintamumą. Kintamumui prognozuoti autoriai naudojo dviejų tipų neuroninius tinklus: NNR (angl. *Neural Network Regression*) ir RNN (angl. *recurrent Neural Network*). Modeliams apmokyti ir prognozėms atlikti autoriai pasirinko naudoti prieš tai buvusių dienų kintamumus, prieš tai buvusias valiutų kursų pokyčių logaritmines absoliučias grąžas ir aukso arba naftos kainų pokyčių logaritmines grąžas. Modeliams įvertinti autoriai naudojo šaknies iš vidutinių kvadratinių paklaidų (angl. root mean squared error, RMSE) vertinimo metriką. Gauti rezultatai parodė, kad tiksliausiai veikia RNN modeliai. Savo tikslumu abu modeliai lenkia ir plačiau naudojamą GARCH modelį.

Chun-Xia Zhang, Jun Li, Xing-Fang Huang, Jiang-She Zhang, Hua-Chuan Huang [40] tyrė Amerikos technologijų kompanijos Apple akcijų kainų kitimą. Savo darbe autoriai siekė sukurti modelius gebančius prognozuoti kintamumą, kuris bus po 22 darbo dienų, tiksliau nei GARCH modeliai. Šiam darbui atlikti buvo pasirinktos LSTM (angl. *long short term memory*) ir TCN (angl. *temporal convolutional networks*) architektūros, o joms apmokyti mokslininkai naudojo daugiau nei 2500 akcijų kainų, kurios buvo renkamos dienos pabaigoje. Taip pat šių kainų pokyčiams buvo paskaičiuojamos logaritminės grąžos. Rezultatai parodė, kad tiksliausiai kintamumą prognozuoja TCM tipo modeliai ir savo tikslumu lenkia GARCH modelius.

Xixi Li, Yanfei Kang ir Feng Li [24] tyrė būdus kaip galima automatiškai ištraukti esmines laiko eilučių savybes. Vienas iš pasiūlymų, minimų jų darbe, yra iš laiko eilučių sugeneruoti rekurentines diagramas, tada adaptuoti (angl. fine-tune) didelį konvoliucinį neuroninį tinklą užduočiai spręsti tinkamais duomenimis. Taip adaptuoto modelio prieš paskutinis sluoksnis grąžina esmines laiko eilučių savybes.

Efthymios Tzinis, Georgios Paraskevopoulos, Christos Baziotis, Alexandros Potamianos [35] savo darbe tyrė kalbos emocijų klasifikavimą. Mokslininkai pastebėjo, kad iš rekurentinių diagramų gaunamos RQA metrikos, puikiai tinka apmokyti ilgos trumpalaikės atminties modelius (angl. LSTM), kurie geba klasifikuoti žmogaus emocijas girdimas balse.

Marc Wenninger, Sebastian P. Bayerl, Jochen Schmidt1, Korbinian Riedhammer [37] ir Nima Hatami, Yann Gavet, Johan Debayle [21] savo darbuose tyrinėjo garsų klasfikavimą. Abiejų darbų autoriai garsų klasifikavimui pasirinko naudojo rekurentines diagramas ir liekanų konvoliucinį neuroninį tinklą (angl, ResNet).

Otávio Penatti ir Milton Santos [29] ieškojo būdų kaip klasifikuoti žmonių judesius. Mokslininkai aptiko, kad iš giroskopo surinktų duomenų nubraižius rekurentines diagramas, galima klasifikuoti žmonių judesius panaudojant SVM (angl. support vector machine) mašininio mokymo algoritmą.

Apžvelgus literatūros šaltinius, kintamumo prognozavimo ir rekurentinių diagramų temomis, buvo padarytos išvados:

- kintamumo prognozavimui yra plačiai taikomi dirbtinių neuroninių tinklų modeliai,
- modelių rezultatai yra lyginami su slenkančio vidurkio pagrindu veikiančiu GARCH modelio rezultatais,
- rekurentinės diagramos ir konvoliuciniai neuroniniai tinklai yra naudojami klasifikuoti laiko eilutes.

2. Naudojamų algoritmų teorija

2.1. Kintamumas

Literatūroje minimas ne vienas būdas, kaip galima skaičiuoti kintamumą [34]. Šiame darbe nuspręsta apsiriboti būdu, kuris skaičiuoja kintamumą pagal logaritmines grąžas:

$$r_t = ln(\frac{P_t}{P_{t-1}}) \tag{2.1}$$

kur P_t ir P_{t-1} žymi akcijų kainą dienose t ir $t-1$.

Pats kintamumas yra skaičiuojamas iš logaritminių grąžų pagal standartinio nuokrypio formulę:

$$\sigma = \sqrt{\frac{\sum_{t=1}^{n}(r_t - \bar{r})^2}{n-1}} \tag{2.2}$$

σ žymi kintamumą, n – duomenų kiekis, r_t yra logaritminė grąža apskaičiuota dienai t, $\bar{r}$ žymi visų logaritminių grąžų vidurkį.

2.2. Rekurentinės diagramos

Kintamumo prognozavimui pasirinkta naudoti istorinius kainų pokyčius. Šie pokyčiai sudaro laiko eilutes. Laiko eilutėmis vadinsime skaičių sekas sudarytas iš $N+1$ reikšmių $f_0, f_1, ..., f_N$.

Vienas iš būdų, kaip galima tirti laiko eilutes, jų reguliarius ir nereguliarius pasikartojimus, yra pasitelkiant rekurentines diagramas [25, 36].

2.2.1. Signalo būsenos

Prieš pradedant gilintis į rekurentines diagramas yra svarbu suprasti kas yra signalo būsena. Signalo būsena vadinsime signalo reikšmių vektorius aprašytus formule 2.3 [11]:

$$y_i = (f_i, f_{i+d}, f_{i+2d}, ..., f_{i+(D-1)d}), \quad i = 0, 1, ..., M, \quad M = N - (D-1)d, \tag{2.3}$$

čia D žymi signalo dimensijų skaičių, o d – signalo vėlinimą.

Rekurentinės diagramos dažniausiai skaičiuojamos su $D = 2$ ir $d = 1$ parametrais [11], todėl šiame darbe rekurentinės diagramos bus skaičiuojamos būtent su šiomis reikšmėmis. Norint paeksperimentuoti ir pabandyti kitokias šių parametrų reikšmes, D parametrą galima parinkti artimą signalo koreliacinei dimensijai, o d rinktis reikšmę su kuria signalo auto koreliacijos funkcija pirmą karta tampa artima nuliui [11].

2.2.2. Rekurentinių diagramų apibrėžimas

Rekurentinės diagramos padeda suprasti daugiadimensinių signalų atsikartojimus, juos atvaizduojant dviejų dimensijų grafike. Rekurentinių diagramų būsenos laikomos kvadratinėse matricose *R*, kuriose abi ašys žymi laiką.

$$\Theta(x) := \begin{cases} 1, & x > 0 \\ 0, & x \leqslant 0 \end{cases} \tag{2.4}$$

$$R^r_{i,j} = \Theta(r - ||y_i - y_j||), \qquad i, j = 0...N \tag{2.5}$$

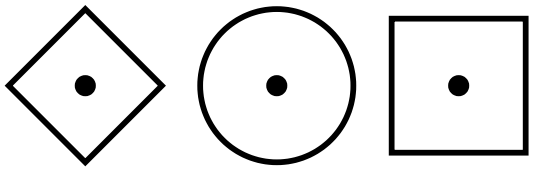
2.5 formulėje N yra būsenų skaičius, y_i žymi signalo būsenų vektorių (2.3 formulė), laiko metu i, r yra didžiausias atstumas tarp būsenų, laikomas atsikartojimu, $||\cdot||$ žymi normą, Θ – Hevisaido funkcija (2.4 formulė).

2.2.3. Slenkstinio parametro pasirinkimas

Vienas svarbiausių parametrų, kuriuos reikia pasirinkti, prieš sudarant rekurentinę diagramą yra slenkstinio parametro r pasirinkimas. Slenkstinis parametras r žymi didžiausią toleruojamą atstumą tarp dviejų signalo būsenų, kad jos būtų laikomos atsikartojančiomis. Šis parametras yra reikalingas, nes praktikoje idealiai atsikartojančių būsenų nebūna.

2.2.4. Normos pasirinkimas

Kitas svarbus pasirinkimas kuriant rekurentines diagramas yra normos pasirinkimas. Normos naudojamos atstumų tarp signalo būsenų skaičiavime (2.5 formulė). Populiariausios naudojamos normos yra L_2 (Euklidinė), L_1 (Manheteno) ir L_∞ (maksimumo) [11, 25]. Jų įtaka atstumų skaičiavimui pavaizduota 1 pav. Derėtų paminėti, kad L_1 norma reikalauja mažiausių skaičiavimo resursų, todėl ją naudojantys algoritmai veikia greičiausiai.

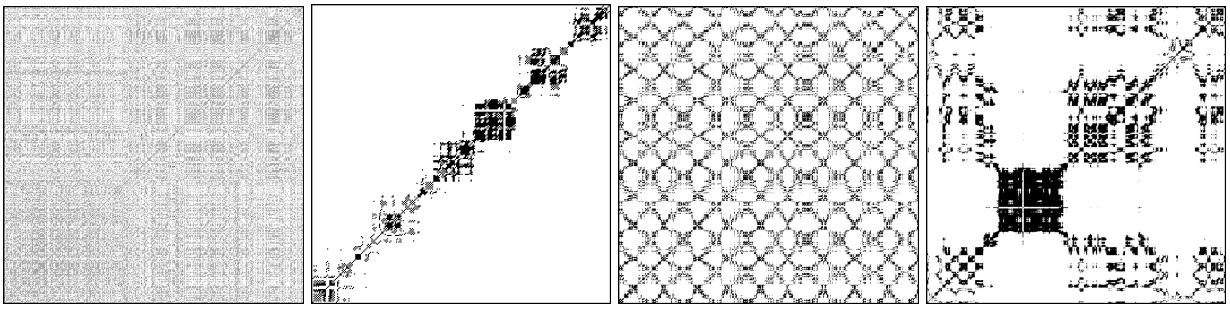


1 pav. Normų pasirinkimo įtaka atstumų skaičiavimui aplink tašką. L_1 norma (iliustracija kairėje), L_2 norma (iliustracija per vidurį), L_∞ norma (iliustracija dešinėje).

2.2.5. Struktūrų interpretavimas

Rekurentinių diagramų paskirtis yra išryškinti signalų savybes ir taip suteikti papildomos informacijos. Pagrindinės rekurentinių diagramų struktūros yra šios [11]:

- **Periodinės ir kvazi-periodinės struktūros** – diagramoje matomos įstrižainės linijos arba šachmatų lentą primenančios struktūros. Šios struktūros parodo, kad signale yra periodų ir signalų būsenos kartojasi (2c pav.).
- **Homogeniškos struktūros** – diagramoje matomi tolygiai išsidėstę juodi taškai. Jie parodo, kad signalas yra stacionarus (signalo statistinės savybės, kaip kad vidurkis, nekinta) (2a ir 2c pav.).
- **Homogeniškos, be įžvelgiamos tvarkos struktūros** – signalas yra atsitiktinis triukšmas (2a pav.).
- **Išbalę kampai** – išbalęs viršutinis kairysis ir apatinis dešinysis kampai parodo, kad signale vyrauja tendencija. Tendencijos pavyzdys gali būti signalo reikšmių augimas (2b pav.).



(a) (b) (c) (d)

2 pav. Rekurentinių diagramų struktūrų pavyzdžiai. a – atsitiktinis triukšmas, b – tendencija, c – periodinis signalas, d – sudėtingos kilmės signalas, turintis ir periodų ir lėtai kintančių būsenų ir tendencijų.

- **Horizontalios ir vertikalios juodų taškų linijos** – signalo būsenos tam tikrais laiko momentais nekinta arba beveik nekinta (2d pav.).

2.2.6. Neapibrėžtosios rekurentinės diagramos

Neapibrėžtosios rekurentinės diagramos (angl. Unthresholded Recurrence Plots) atvaizduoja atstumus tarp signalo būsenų. Jos sudaromas naudojantis formule:

$$R_{i,j} = ||y_i - y_j||, \qquad i,j = 0...N \tag{2.6}$$

2.6 formulėje N yra būsenų skaičius, y_i žymi signalo būsenų vektorių (2.3 formulė), laiko metu i, laikomas atsikartojimu, $||\cdot||$ žymi normą.

2.3. RQA

Rekurentinės diagramos suteikia naudingos informacijos apie signalo atsikartojimus, tačiau kadangi tai yra vaizdinis įrankis, jo interpretavimas priklauso nuo interpretuojančio žmogaus. Be to, tokią analizę sunku automatizuoti. Dėl šios priežasties yra naudojama RQA [36].

RQA tai rinkinys metrikų skirtų matuoti rekurentinių diagramų savybes. Toliau bus aprašomos šiame darbe naudojamos RQA metrikos [36, 31].

Kai kurioms metrikoms apskaičiuoti yra reikalinga žinoti pikselių histogramas. 2.7 formulėje yra parašyta, kaip reikia apskaičiuoti horizontalių juodų įstrižainių histogramas, 2.8 formulėje parodyta, kaip reikia apskaičiuoti vertikalių juodų linijų histogramas, o 2.9 formulėje parodyta, kaip reikia apskaičiuoti vertikalių baltų linijų histogramas. Visose formulėse $R_{i,j}$ žymi rekurentinės diagramos pikselio reikšmę esančią koordinatėse i, j. Ši reikšmė yra lygi 1, kai pikselis yra juodos spalvos ir 0 kai pikselis yra baltos spalvos. N yra signalo imčių skaičius, o l yra ieškomos linijos vertė.

$$H_D(l) = \sum_{i,j=1}^{N} (1 - R_{i-1,j-1})(1 - R_{i+l,j+l}) \prod_{k=0}^{l-1} R_{i+k,j+k} \tag{2.7}$$

$$H_V(l) = \sum_{i,j=1}^{N} (1 - R_{i,j-1})(1 - R_{i,j+l}) \prod_{k=0}^{l-1} R_{i,j+k} \tag{2.8}$$

$$HW_V(l) = \sum_{i,j=1}^{N} R_{i,j-1} R_{i,j+l} \prod_{k=0}^{l-1} (1 - R_{i,j+k}) \tag{2.9}$$

Pasikartojimų dažnis (angl. *recurrence rate*, 2.10 formulė) skaičiuoja juodų pikselių kiekį rekurentinėje diagramoje, atmetant LOI (angl. *Line of identity* – pagrindinės įstrižainės) taškus. r – slenkstinis parametras, reikalingas sudaryti rekurentinę diagramą.

$$RR(r, N) = \frac{1}{N^2 - N} \sum_{i \neq j=1}^{N} R_{i,j}^{m,r} \tag{2.10}$$

Determinizmas (angl. *determinism*, 2.11 formulė) sistemos periodinio elgesio nuspėjamumas. d_{min} parametras nustato žemiausios įstrižainės linijos ilgį, naudingas studijuojant sistemos elgseną.

$$DET = \frac{\sum_{l=d_{min}}^{N} l H_D(l)}{\sum_{i,j=1}^{N} R_{i,j}} \tag{2.11}$$

Vidutinis įstrižainės linijos ilgis (2.12 formulė). Ši metrika rodo vidutinį atsikartojimų laiką.

$$L = \frac{\sum_{l=d_{min}}^{N} l H_D(l)}{\sum_{l=d_{min}}^{N} H_D(l)} \tag{2.12}$$

Didžiausias įstrižainės linijos ilgis (2.13 formulė). Ši metrika rodo didžiausią atsikartojimų laiką.

$$L_{max} = \arg\max_{l} H_D(l) \tag{2.13}$$

Divergencija (angl. *Divergence*, 2.14 formulė).

$$DIV = \frac{1}{L_{max}} \tag{2.14}$$

Įstrižainių linijų entropija (2.15 formulė) parodo kokia sudėtinga yra sistema.

$$L_{entr} = -\sum_{l=d_{min}}^{N} p(l) \ln p(l) \quad kai \quad p(l) = \frac{H_D(l)}{\sum_{l=d_{min}}^{N} H_D(l)} \tag{2.15}$$

Laminarumas (angl. *laminarity*, 2.16 formulė) parodo sistemoje esančių pastovių būsenų dažnį.

$$LAM = \frac{\sum_{l=\nu_{min}}^{N} l H_V(l)}{\sum_{i,j=1}^{N} R_{i,j}} \tag{2.16}$$

Gaudymo laikas (angl. *trapping time*, 2.17 formulė) žymi vidutinį vertikalių linijų ilgį ir parodo kaip ilgai sistemos būsena nekinta. Šios metrikos skaičiavimui naudojamas ν_{min} – vartotojo pasirinktas mažiausias horizontalios linijos ilgis.

$$TT = \frac{\sum_{l=\nu_{min}}^{N} \nu H_V(l)}{\sum_{l=\nu_{min}}^{N} H_V^r(l)} \tag{2.17}$$

Didžiausios vertikalios linijos ilgis (2.18) siejamas su būsenomis, kai sistema nustoja kitusi.

$$V_{max} = \arg\max_{l} H_V(l) \tag{2.18}$$

Vertikalių linijų entropija (2.19 formulė) yra dar vienas dydis, kuris parodo, kokia sudėtinga yra sistema.

$$V_{entr} = -\sum_{l=v_{min}}^{N} p(l)\ln p(l) \quad kai \quad p(l) = \frac{H_V(l)}{\sum_{l=v_{min}}^{N} H_V(l)} \tag{2.19}$$

Vidutinis baltos vertikalios linijos ilgis (2.20).

$$W = \frac{\sum_{l=v_{min}}^{N} lHW_V(l)}{\sum_{l=v_{min}}^{N} HW_V(l)} \tag{2.20}$$

Didžiausias baltos vertikalios linijos ilgis (2.21).

$$W_{max} = \arg\max_{l} HW_V(l) \tag{2.21}$$

Baltų linijų divergencija (2.22).

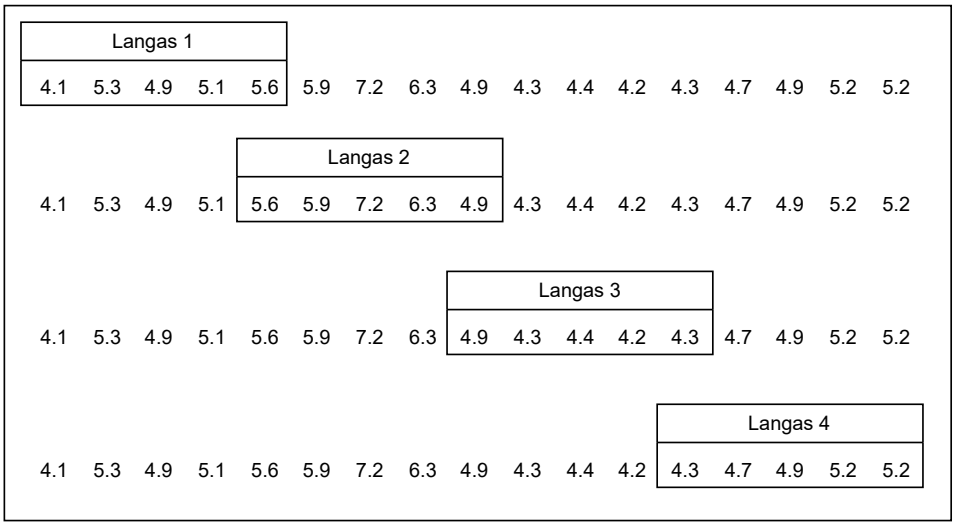
$$W_{div} = \frac{1}{W_{max}} \tag{2.22}$$

Baltų linijų entropija (2.22).

$$W_{entr} = -\sum_{l=v_{min}}^{N} p(l)\ln p(l) \quad kai \quad p(l) = \frac{HW_V(l)}{\sum_{l=v_{min}}^{N} HW_V(l)} \tag{2.23}$$

2.3.1. Slenkančio lango principas

Literatūros šaltiniuose teigiama, kad RQA padeda suprasti, kaip maži pokyčiai paveikia visą sistemą [36]. Kintamumo prognozavimo atveju tai reikštų, kaip maži pokyčiai RQA rezultatuose, gali pakeisti kintamumą. Kadangi šio darbo tikslas yra prognozuoti kintamumą, tai yra labai svarbu. Autoriai [36], tokiems pokyčiams aptikti, siūlo naudoti slenkančio lango principą (3 pav.). Šiuo būdu yra sudaromi nauji duomenų rinkiniai, kuriems yra atskirai braižomos rekurentinės diagramos ir vėliau joms atliekama RQA.

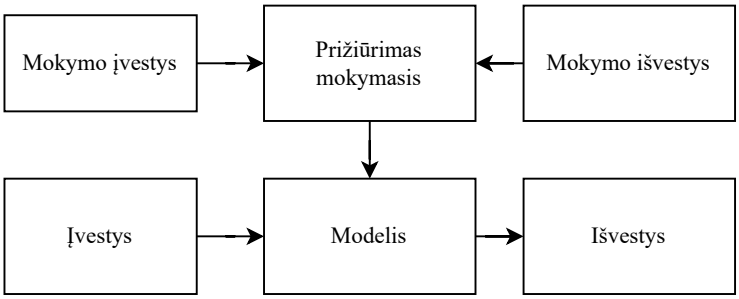


3 pav. Slenkančio lango veikimo principas: pasirenkamas lango dydis (N=5) ir žingsnis (s=4). Pagal pasirinktą lango dydį N, slenkant per žingsnį s, yra sudaromi nauji duomenų rinkiniai.

2.4. Prižiūrimas mokymasis

Prižiūrimas mokymasis (angl. Supervised Learning) tai viena iš mašininio mokymosi (angl. Machine Learning) sprendžiamų užduočių rūšių [39]. Ši rūšis problemų apibūdina užduotis, kai bandoma sukurti modelius, kurie iš tam tikrų požymių (įvesčių) nuspėtų kitus požymius (išvestis). Pavyzdžiui, sukurti modelį, kuris iš kompanijos akcijų kitimo informacijos, bandytų nuspėti kito mėnesio akcijų kainą.

Prižiūrimas mokymasis vadinamas prižiūrimuoju, nes modelio mokymo stadijoje modelio įvestys ir išvestys yra lyginamos su turimais duomenimis. Taip modeliai išmoksta prognozuoti įvestis, kurių nėra turimame duomenų šaltinyje (4 pav.).



4 pav. Prižiūrimas mokymasis. Viršutinėje eilutėje pavaizduota mokymosi fazė, o apačioje pavaizduotas apmokyto modelio naudojimas.

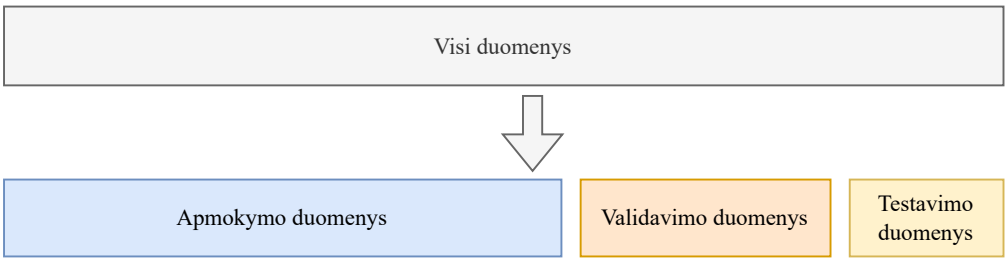
2.4.1. Duomenų padalinimas

Prižiūrimajam mokymuisi reikia mažiausiai dviejų tipų duomenų: skirtų modeliams apmokyti ir skirtų įvertinti jų tikslumui. Duomenų šaltinis dažniausiai būna vienas, todėl yra svarbu mokėti

padalinti duomenis į apmokymui ir vertinimui skirtas dalis. Šiame darbe yra naudojami dviejų tipų duomenų padalinimo būdai: išlaikymo validavimas (angl. hold-out validation) ir k dalių kryžminė validacija (agl. k-fold cross validation).

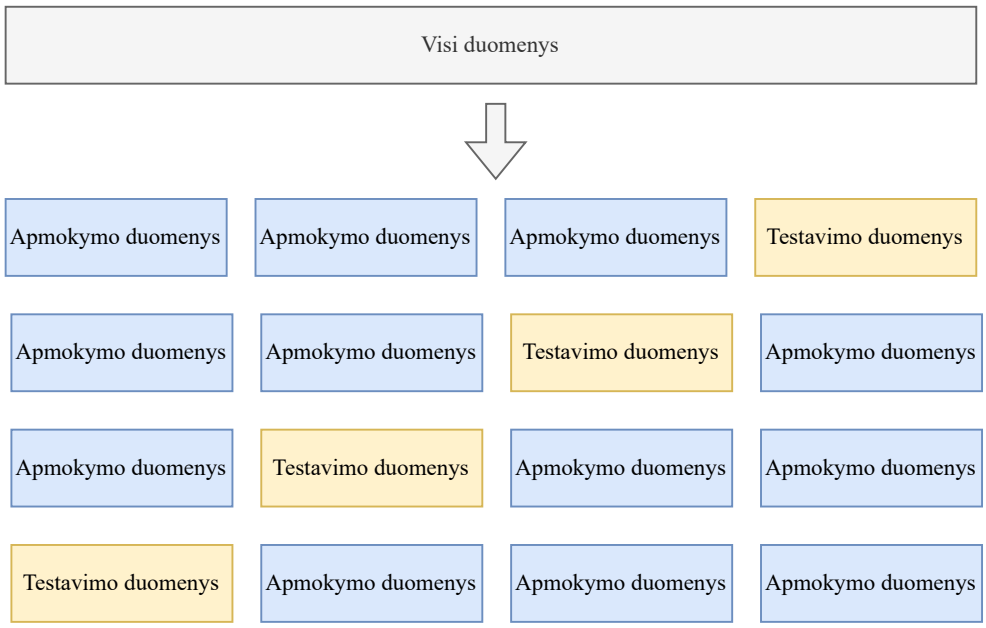
Išlaikymo validavimo būdu duomenys yra padalinami į tris dalis: apmokymo, validavimo ir testavimo duomenis, pasirenkant skirtingas proporcijas (5 pav.).

- Apmokymo duomenys – skirti apmokyti neuroninį tinklą.
- Validavimo duomenys – skirti įvertinti modelio tikslumą apmokymo metu.
- Testavimo duomenys – skirti įvertinti modelio tikslumą apmokius modelį.



5 pav. Duomenų padalinimas išlaikymo validavimo būdu. Kai apmokymo duomenų aibės dydis yra 70%, validavimo aibės dydis yra 20% ir testavimo aibės dydis yra 10%.

Kryžminės validacijos būdu duomenys yra padalinami į k dalių. Iš tų dalių vieną yra naudojama kaip testavimo aibė, o likusios naudojamos apmokyti neuroninį tinklą. Apmokius modelį toks procesas yra kartojamas pasirinkus kitą dalį testavimui iki kol visos dalys yra panaudojamos testavimui (6 pav.).



6 pav. Duomenų padalinimas k dalių kryžminės validacijos būdu, kai k=4. Šio proceso metu yra apmokomi 4 modeliai, kiekvienam apmokyti panaudojant skirtingus apmokymo ir testavimo duomenis.

Viename literatūros šaltinyje [38] buvo tiriama išlaikymo validavimo ir k dalių kryžminės validacijos duomenų padalinimo įtaką didelių duomenų klasifikavimo kokybei. Jame atlikti eksperimentai su dideliais duomenų kiekiais parodė, kad k dalių kryžminė validacijos duomenų padalinimas yra pranašesnis už išlaikymo validavimo padalinimą, nes suteikia 0.1-0.3% tikslesnius rezultatus. Taip pat šaltinyje paminėta, kad reikiamas laiko kiekis modelių apmokymui, gal būti daug didesnės naudojant kryžminės validacijos padalinimą.

2.4.2. Regresija

Regresija tai viena iš prižiūrimo mokymosi užduočių. Užduotis vadinama regresija, kai siekiama apmokyti modelį, kuris prognozuoja skaitinę reikšmę. Regresijos pavyzdys galėtų būti filmo populiarumo prognozavimas, pagal jame vaidinančius aktorius.

Norint įvertinti modelio veikimą yra svarbu atskirti kaip smarkiai modelio prognozės skiriasi nuo tikrų reikšmių. Šio darbo metu regresijos uždavinių tikslumui matuoti bus naudojama MSE:

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(\hat{y_i} - y_i)^2 \tag{2.24}$$

N – duomenų imčių skaičius, $\hat{y_i}$ – prognozuojama reikšmė, y_i – tikra reikšmė.

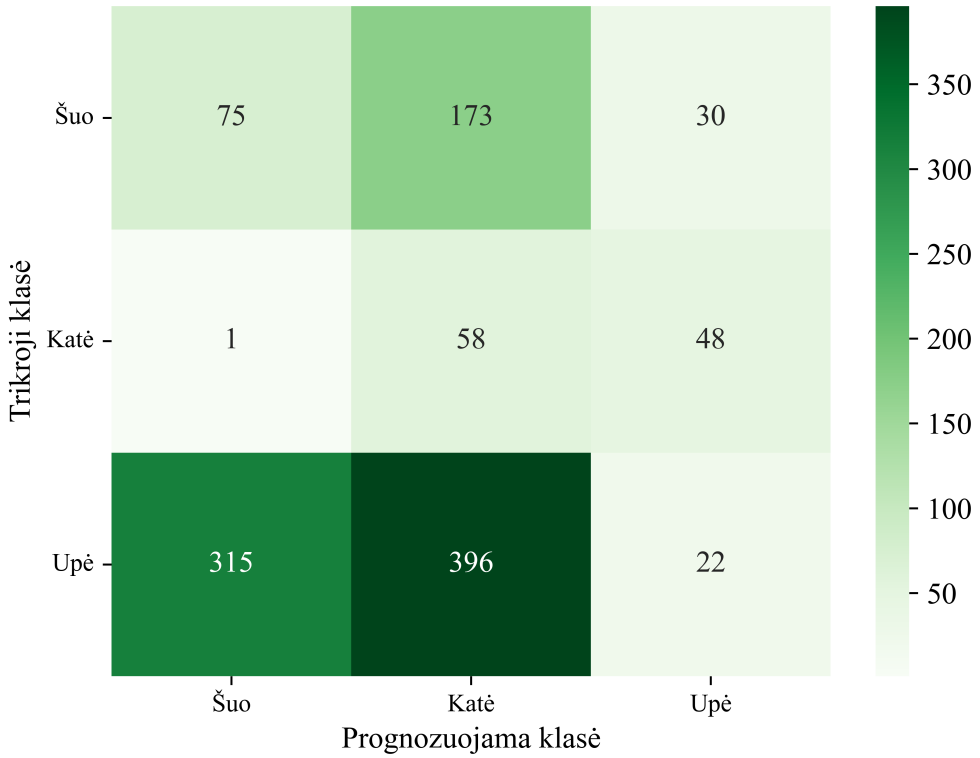
2.4.3. Klasifikavimas

Užduotis, kai siekiama prognozuoti ne skaitinę reikšmę, bet kategoriją, vadinama klasifikavimo užduotimi. Klasifikavimo užduotis gali būti skirstoma į vienos klasės atpažinimo užduotį arba kelių. Kadangi šiame darbe atliekamas kelių klasių atpažinimo uždavinys, todėl bus aprašomas tik tokio tipo uždavinys. Tokio klasifikavimo uždavinio atpažinimo pavyzdys galėtų būti skaičiaus nuotraukoje atpažinimas.

Norint įvertinti modelio atliekančio klasifikavimo užduotį tikslumą yra naudojamos painiavos matricos (angl. Confusion matrix, 7 pav.) ir metrikos gaunamos iš jų [19]. Tam, kad būtų lengva suprasti painiavos matricas, reikia gebėti įvertinti modelio rezultatą. Dažniausiai yra norima suprasti modelio gebėjimą atpažinti kurią nors vieną klasę, ją vadinsime lyginamąja klase. Kai modelio rezultatas sutampa su lyginamąja klase, jis laikomas teigiamu, kai nesutampa neigiamu. Tai žinant tampa lengviau suprasti painiavos matricos reikšmes, kurių yra 4:

- Teisingai klasifikuotos teigiamos reikšmės (angl. true positives, TP) – kiekis atvejų kai modelis prognozuoja teigiamą klasę, kuri yra teigiama.
- Teisingai klasifikuotos neigiamos reikšmės (angl. true negatives, TN) – kiekis atvejų kai modelis prognozuoja neigiamą klasę, kuri yra neigiama.
- Klaidingai klasifikuotos teigiamos reikšmės (angl. false positives, FP) – kiekis atvejų kai modelis prognozuoja teigiamą klasę, kuri yra neigiama.
- Klaidingai klasifikuotos neigiamos reikšmės (angl. false negatives, FN) – kiekis atvejų kai modelis prognozuoja neigiamą klasę, kuri yra teigiama.

Turint painiavos matricą, galima išgauti metrikas reikalingas įvertinti modelio tikslumui. Šio darbo metu yra naudojamos keturios metrikos, padedančio įvertinti, klasifikuojančio modelio tikslumą: ACC, PREC, REC ir F1. Paskutinės trys metrikos, dar gali būti skirstomos į mažumos (angl.



7 pav. Painiavos matricos pavyzdys.

micro) arba daugumos (angl. *macro*) rūšis, kai atliekamas kelių klasių klasifikavimas. Apžvelgus literatūros šaltinius nepavyko rasti informacijos apie vienos arba kitos metrikų rūšių pranašumą, todėl nuspręsta apsiriboti tik daugumos metrikomis.

ACC yra viena iš populiariausių metrikų skirtų įvertinti modelio tikslumą. Ji nusako kokia tikimybė, kad modelis teisingai nuspės prognozuojamą klasę.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.25}$$

Deja, tikslumo metrika turi trūkumų. Kai duomenyse klasių skaičius yra nevienodas, ši metrika gali klaidinti. Pavyzdžiui, jei klasės A yra 990 elementų, o klasės B yra tik 10 elementų, tada modelis pranašaudamas visada klasę A, pasieks 99% tikslumą. Dėl šios priežasties naudojamos kitos metrikos.

PREC nusako santykį, kiek modelis atspėja klasių teisingai iš visų teisingų klasių. Kitaip tariant, kiek galima pasitikėti modeliu.

$$PREC_{macro} = \frac{\sum_{k=1}^{K} PREC_k}{K}, \quad kur \quad PREC_k = \frac{TP_k}{TP_k + FP_k} \tag{2.26}$$

REC vertina modelio gebėjimą atpažinti visas teigiamas klases.

$$REC_{macro} = \frac{\sum_{k=1}^{K} REC_k}{K}, \quad kur \quad REC_k = \frac{TP_k}{TP_k + FN_k} \tag{2.27}$$

Dar viena iš metrikų tai F1. Ši metrika apjungia PREC ir REC metrikas:

$$F1_{macro} = 2 * \frac{PREC_{macro} * REC_{macro}}{PREC_{macro}^{-1} + REC_{macro}^{-1}} \tag{2.28}$$

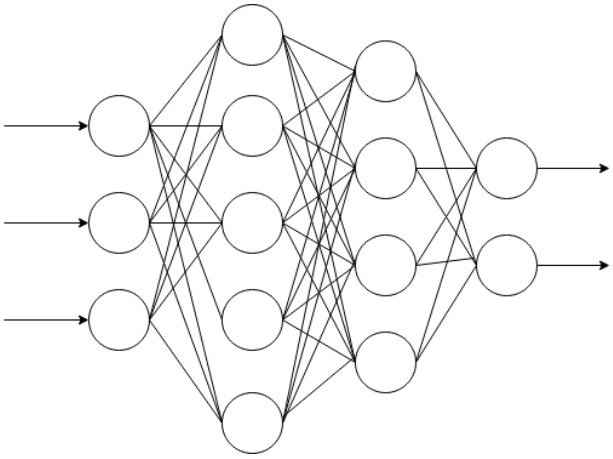
2.5. Giliojo mokymosi metodai

Gilusis mokymasis (angl. Deep learning) tai mašininio mokymosi dalis, kurios modeliai yra sudaryti iš daug sluoksnių turinčių neuroninių tinklų [39]. Pastebėta, kad šie modeliai smarkiai pranoksta tradicinius modelius sprendžiant regresijos ir klasifikavimo uždavinius. Dėl šios priežasties, šiame darbe nuspręsta naudoti būtent giliojo mokymosi metodus.

2.5.1. Gilieji neuroniniai tinklai

Gilieji neuroniniai tinklai, dar vadinami gileisiais tiesioginio sklidimo neuroniniais tinklais (angl. deep feedforward networks), yra esminiai ir plačiausiai žinomi neuroniniai tinklai [18].

Literatūros šaltiniuose [17] minima, kad šio tipo neuroniniai tinklai (Fig. 8) yra efektyvi alternatyva tradiciniams statistiniams skaičiavimams. Didžiausias šių modelių pranašumas yra tai, kad tokie neuroniniai tinklai gali būti apmokyti atitikti bet kokią funkciją.



8 pav. Gilusis neuroninis tinklas su dviem paslėptais sluoksniais, trimis įvesties vektoriais ir dvejais išeities vektoriais.

Gilieji tiesioginio sklidimo neuroniniai tinklai susideda iš sluoksnių. Pirmas sluoksnis vadinamas įvesties sluoksniu, paskutinis sluoksnis – išvesties, o sluoksniai tarp jų vadinami bendru pavadinimu – paslėptais sluoksniai. Sluoksniai šiuose tinkluose gali būti suprantami kaip skaičių vektoriai (svoriai). Norint apskaičiuoti sluoksnių svorius, kitaip tariant, apmokyti neuroninį tinklą, rekomenduojama pasirinkti aktyvacijos funkciją ir tikslo funkciją (angl. cost function). Tuomet taikant atgalinio sklidimo algoritmą (angl. back-propagation), yra atnaujinami sluoksnių svoriai [18].

2.5.2. Aktyvacijos funkcijos

Aktyvacijos funkcijos yra skirtos sukurti neuroninius tinklus, kurie galėtų atitikti netiesines funkcijas. Šios funkcijos naudojamos atgalinio sklidimo algoritmo metu, atnaujinant sluoksnių svorius.

Literatūros šaltiniuose minima, kad pati populiariausia ir rekomenduojama aktyvacijos funkcija yra ReLU (9 pav.) [39]:

$$ReLU(x) = max(0, x) \tag{2.29}$$

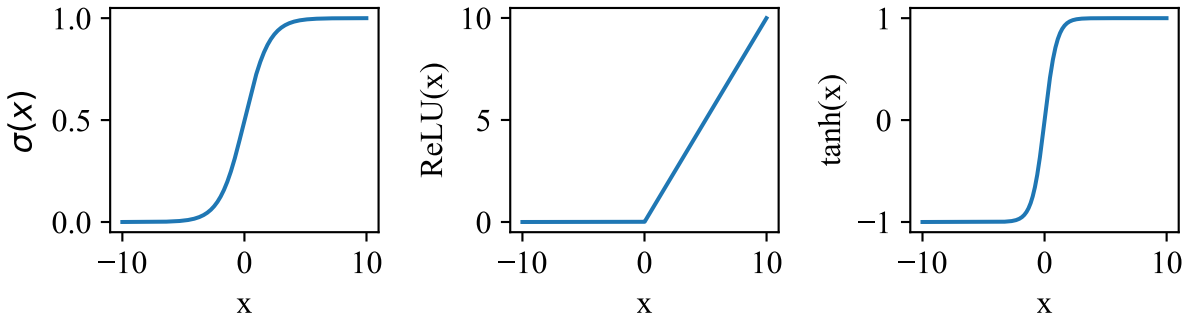
Ši funkcija yra artima tiesiniai dėl to išlieka, daug tiesinių savybių svarbių modelio optimizavimui atliekant atgalinio sklidimo algoritmą.

Kitos dvi dažnai naudojamos aktyvacijos funkcijos yra sigmoido (angl. sigmoid, 2.30 formulė) ir tanh (2.31 formulė).

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.30}$$

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + exp(-2x)} \tag{2.31}$$

Šiame darbe šios aktyvacijos funkcijos yra naudojamos su ilgos trumpalaikės atminties neuroniniais tinklais.



9 pav. Aktyvacijos funkcijų grafikai: sigmoido grafikas (iliustracija kairėje), relu grafikas (iliustracija viduryje), tanh grafikas (iliustracija dešinėje).

2.5.3. Tikslo funkcijos

Siekiant apmokyti neuroninį tinklą yra svarbu tinkamai pasirinkti tikslo funkciją. Taip yra dėl to, kad atgalinio sklidimo algoritmas, atnaujina sluoksnių svorius, taip kad tikslo funkcijos reikšmė būtų mažiausia (siekia ją minimizuoti) [18].

Yra įvairių tikslo funkcijų ir jų pasirinkimas priklauso nuo užduoties. **Regresijos** uždaviniams spręsti dažniausiai MSE arba vidutinės absoliučios paklaidos (angl. mean absolute error, MAE) funkcijos. **Klasifikacijos** uždaviniams spręsti dažniausiai naudojamos dvejetainės kryžminės entropijos (angl. *binary cross entropy*) arba kryžminės entropijos (angl. *cross entropy*) tikslo funkcijos. Dvejetainės kryžminės entropijos tikslo funkcija naudojama kai modelis bando atsakyti į klausimą, kuris susideda iš dviejų reikšmių: taip arba ne. Pavyzdžiui: ar paveikslėlyje matomas šuo, ar yra matoma katė. Kryžminė entropija naudojama, kai turi būti pasirenkama vieną iš kelių klasių ir kelios klasės negali būti prognozuojamos. Pavyzdžiui, koks skaitmuo, nuo 0 iki 9, yra matomas nuotraukoje.

2.5.4. Konvoliuciniai neuroniniai tinklai

Konvoliuciniai neuroniniai tinklai yra neuroninių tinklų rūšis, dažniausiai naudojama atpažinti paveikslėliuose esančią informaciją.

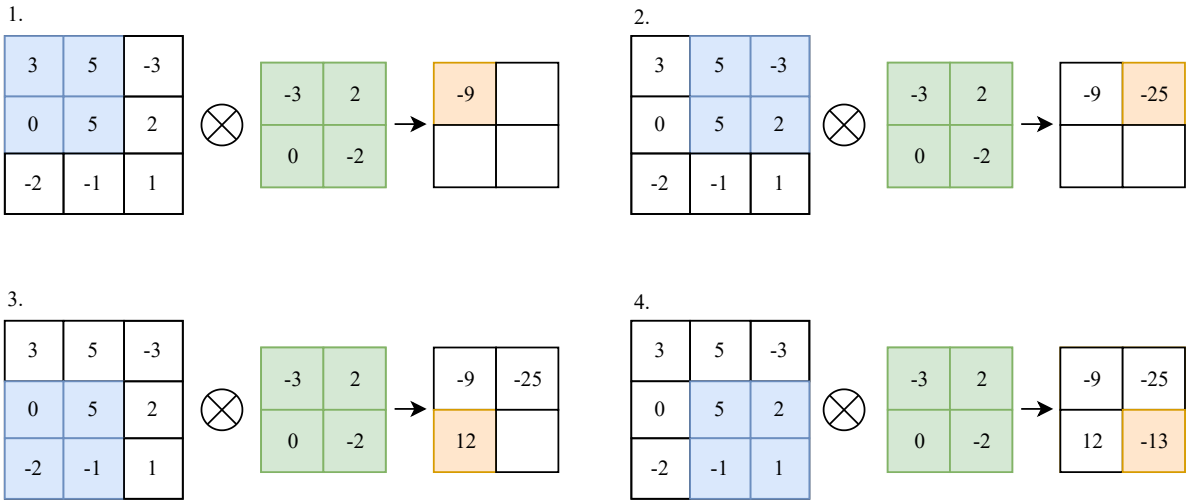
Literatūros šaltinyje [12] išskiriami trys pagrindiniai konvoliucinių neuroninių tinklų pranašumai juos lyginant su tradiciniais neuroniniais tinklais:

- Mažesnis apmokamų parametrų skaičius – padeda išvengti persimokymo (angl. overfitting) ir sustiprina modelio generalizaciją (angl. generalization).

- Požymių išgavimo (angl. featue extraction) ir klasifikatimo sluoksniai yra apmokami kartu.
- Dideli neuroniai tinklai yra lengviau apmokami naudojant konvoliucinius neuroninius tinklus nei kito tipo neuroninius tinklus.

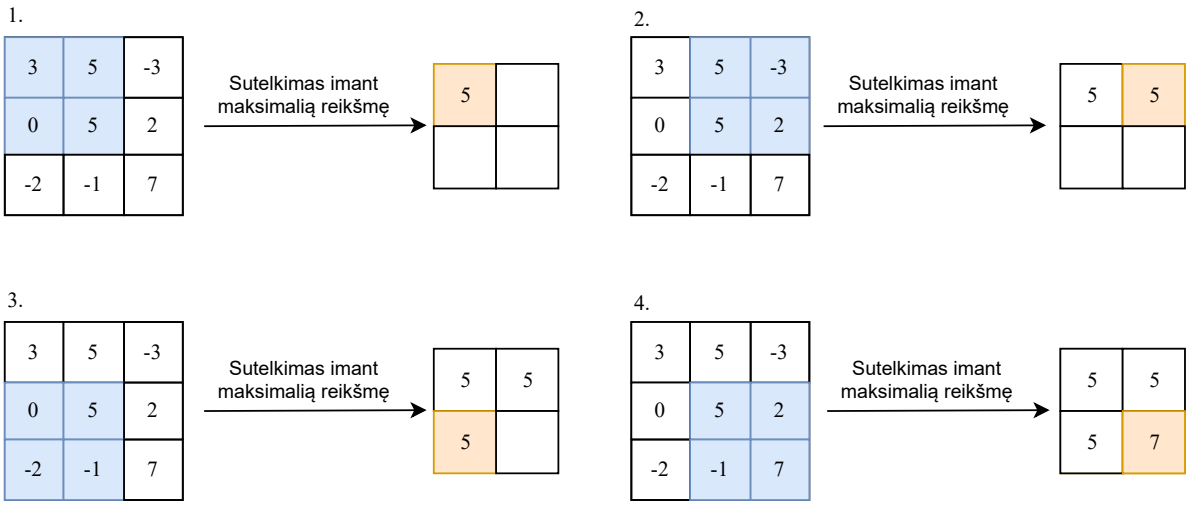
Konvoliuciniai neuroniniai tinklai taip pat kaip ir dirbtiniai neuroniai tinklai yra sudaryti iš sluoksnių. Yra išskiriami 5 tipų konvoliucinių neuroninių tinklų sluoksniai [12]:

1. Konvoliuciniai sluoksniai. Sudaryti iš konvoliucinių filtrų vadinamų kerneliais.
    - Kernelis – diskrečių numerių matrica. Matricos reikšmės yra konvoliucinio sluoksnio svoriai, kurie yra koreguojami apmokant neuroninį tinklą. Šitaip konvoliuciniai sluoksniai išmoksta atpažinti svarbias savybes.
    - Konvoliucijos – operacijos atliekamos tarp sluoksnio įvesties matricos ir kernelio.



10 pav. Konvoliucijos operacijos pavyzdys. Kernelio filtras (pažymėtas žalia spalva) yra pritaikomas įvesties matricos dalims (pažymėtomis mėlyna spalva) ir taip gaunamas rezultatas (pažymėtas oranžine spalva).

2. Sutelkimo sluoksniai. Kaip ir konvoliuciniai sluoksniai sumažina įvesties požymių kiekį, tačiau skirtingai nei konvoliucijos kerneliai, šie sluoksniai neturi svorių, kurie yra koreguojami apmokant neuroninį tinklą. Dažniausiai naudojami sutelkimo sluoksniai yra: sutelkimo imant maksimalią reikšmę (angl. max pooling), sutelkimo imant mažiausią reikšmę (angl. min pooling) ir sutelkimo vidurkinant (angl. average pooling) sluoksniai.



11 pav. Sutelkimo imant maksimalią reikšmę operacijos pavyzdys. Įvesties matricos dalims (pažymėtomis mėlyna spalva) yra pritaikoma sutelkimo imant maksimalią reikšmę operacija ir taip gaunamas rezultatas (pažymėtas oranžine spalva).

3. Aktyvacijos funkcijos. Padeda apibrėžti netiesinį sąryšį tarp sluoksnio įvesties ir išvesties. Aktyvacijos funkcijų naudojimas padeda modeliams išmokti sudėtingus sąryšius.

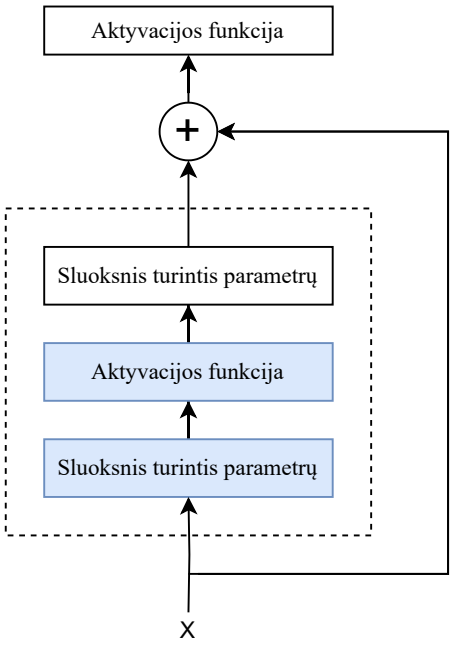
4. Pilnai sujungtų sąryšių sluoksnis. Dažniausiai naudojamas kaip paskutinis, klasifikavimui skirtas sluoksnis. Šis sluoksnis yra identiškas dirbtinių neuroninių tinklų sluoksniui.

5. Tikslo funkcijos.

2.5.5. Liekanų neuroninis tinklas

Liekanų neuroninis tinklas (angl. ressidual neural network arba ResNet) tai konvoliucinis neuroninis tinklas, kurio pagrindu 2015 metais buvo laimėtos ImageNet aptikimo, ImageNet lokalizacijos, COCO aptikimo ir COCO segmentacijos varžybos [22].

Šio tipo modeliai, skirtingai nei prieš tai buvę, išsprendė nykstančių gradientų problemą (angl. vanishing gradient problem) pridedant liekanų blokus – jungtis tarp sluoksnių įvesčių ir išvesčių (12 pav).



12 pav. Diagramoje pavaizduotas liekanų blokas [39]. X – įvestis, kuri yra pridedama prie paskutinio sluoksnio išvesties. Šitos įvesties pridėjimas išsprendžia nykstančių gradientų problemą: jei paskutinio sluoksnio turinčio parametrų (pažymėto baltai) koeficientai mokymo metu nustotų keistis, į kitus sluoksnius vis tiek nueitų įvestis ir kiti sluoksniai galėtų toliau mokytis.

2.5.6. Žinių perdavimas

Žinių perdavimas (angl. transfer learning) tai būdas perduoti apmokyto modelio žinias naujam modeliui. Vienas iš būdų kaip galima atlikti žinių perdavimą yra naudoti modelių adaptavimą (angl. fine-tuning) [33].

Adaptavimas pradedamas nukopijuojant visus, išskyrus paskutinio sluoksnio, apmokyto modelio sluoksnių koeficientus į norimą apmokyti modelį. Paskutinio sluoksnio koeficientai nėra kopijuojami, nes jie yra pritaikyti atlikti apmokyto modelio užduotį ir netinka naujam modeliui. Dėl to paskutinio sluoksnio koeficientai norimam apmokyti modeliui turėtų būti inicijuoti atsitiktinai. Tai atlikus, visi naujojo modelio sluoksniai išskyrus paskutinį turėtu būti užšaldyti (jų koeficientai negali būti keičiami apmokant modelį) ir modelis turi būti apmokomas. Apmokius modelį jo paskutinis sluoksnis turės tinkamus koeficientus reikiamai užduočiai atlikti. Yra būtina užšaldyti koeficientus nes atsitiktinai inicijuotas sluoksnis gali iškraipyti perduodamus koeficientus ir taip žinių perdavimą padaryti neefektyvų. Kai modelio klasifikavimo (paskutinis) sluoksnis nustoja mokytis, koeficientai turėtu būti atšaldomi ir sumažinus mokymosi žingsnį modelis dar kartą turėtu būti apmokomas. Šitaip modelio tikslumas turėtų dar pagerėti.

Vienas iš klausimų kuris gali kilti tai, ar galima panaudoti žinių perdavimą iš skirtingų sričių? Pavyzdžiui, ar galima panaudoti modelio, apmokyto atskirti gyvūnus, žinias, atlikti finansinės kilmės užduotis. Literatūros šaltinyje [33] yra keliamas toks pat klausimas tik su medicininės kilmės duomenims. Šaltinyje įrodoma, kad žinių perdavimas ir modelių adaptavimas yra tinkamas būdas norint adaptuoti modelius nesusijusios kilmės duomenimis.

2.5.7. Rekurentiniai neuroniniai tinklai

Rekurentiniai neuroniniai tinklai (toliau RNN), yra rūšis neuroninių tinklų, kuri geba priimti skirtingo ilgio eilučių sekas ir atlikti prognozes. Esminis skirtumas nuo paprastų dirbtinių neuroninių tinklų yra tai, kad RNN kartu su įvestimi priima paslėptą būseną ir kartu su išvestimi atiduoda, pakitusią paslėptą būseną. Šitaip modeliai gali įsiminti informaciją gautą iš prieš tai buvusių elementų [39].

Paslėptos būsenos, laiko momentu t, apskaičiavimo formulė:

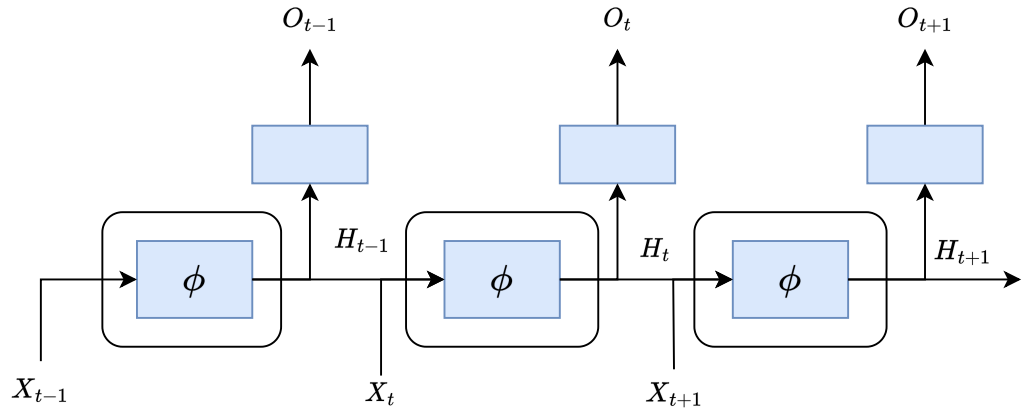
$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{2.32}$$

šioje formulėje X_t žymi įvesties reikšmę laiko metu t, W_{xh} yra svoriai skirti apmokyti modelį priimti įvestį, H_{t-1} yra prieš tai buvusios paslėptos būsenos reikšmė, W_{hh} yra svoriai skirti apmokyti modelį priimti paslėptą būseną, b_h – laisvasis narys (angl bias), ϕ – aktyvacijos funkcija, H_t – gaunama paslėpta būsena.

Išvesties sluoksnio rezultato skaičiavimo formulė, laiko momentu t:

$$O_t = H_t W_{hq} + b_q \tag{2.33}$$

čia W_{hq} yra išvesties sluoksnio svoriai, b_q – laisvasis narys ir O_t – modelio rezultatas laiko momentu t.



13 pav. RNN modelio skaičiavimo grafikas, parodantis kaip modelio būsenos ir rezultatai yra skaičiuojami laiko momentais $t-1$, t ir $t+1$.

2.5.8. Ilgos trumpalaikės atminties modelis

Ilgos trumpalaikės atminties (angl. Long-Short term memory) modeliai, tai rekurentinių neuroninių tinklų rūšis. Paprasti rekurentiniai neuroniniai tinklai turi nykstančių ir sprogstančių gradientų problemą. LSTM modeliai šių bėdų neturi nes pakeičia rekurentinių tinklų mazgus atminties ląstelėmis [39].

Ilgos trumpalaikės atminties modelių atminties ląstelės susideda iš įvesties, užmiršimo ir išvesties sklendžių, įvesties mazgo, vidinės būsenos ir paslėptos būsenos.

Užmiršimo sklendės nusprendžia ar reikia laikyti dabartinę įsimintą reikšmę, esančią vidinėje atmintyje, ar verčiau ją išvalyti. Užmiršimo sklendės atliekamos operacijos formulė:

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{2.34}$$

Šioje formulėje σ yra sigmoido aktyvacijos funkcija, X_t yra įvestis laiko momentu t, W_{xf} yra svoriai skirti apmokyti modelį priimti įvestį, H_{t-1} tai paslėpta būsena, prieš tai buvusiu laiko mementu, W_{hf} – svoriai skirti išmokti priimti prieš tai buvusią būseną, b_f – laisvasis narys.

Atidarymo sklendė sprendžia kiek įvesties mazgo informacijos turi būti pridėtą į atminties ląstelės vidinę būseną. Atidarymo sklendės atliekamos operacijos formulė:

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{2.35}$$

Šios formulės nariai yra analogiški užmiršimo sklendės nariams, tik pritaikyti atidarymo sklendei.

Išvesties sklendė sprendžia ar atminties ląstelė turi daryti įtaką modelio išvesčiai. Išvesties sklendės atliekamos operacijos formulė:

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{2.36}$$

Šios formulės nariai yra analogiški užmiršimo sklendės nariams, tik pritaikyti išvesties sklendei.

Įvesties mazgas apskaičiuoja reikšmę, kuri vėliau gali būti pridėta prie atminties ląstelės vidinės atminties. Įvesties mazgo atliekamos operacijos formulė:

$$\widetilde{C}_t = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{2.37}$$

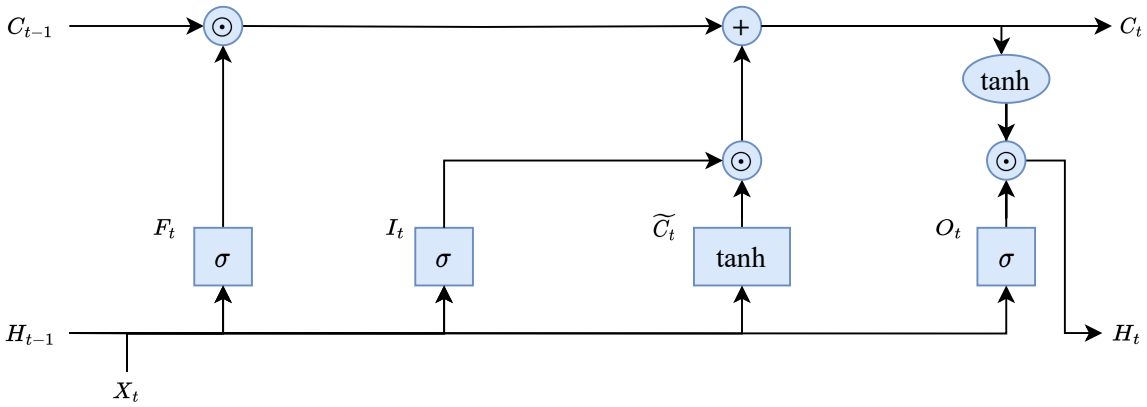
Šioje formulėje $tanh$ yra tanh aktyvacijos formulė. Likę formulės nariai yra analogiški užmiršimo sklendės nariams, tik pritaikyti įvesties mazgui.

Vidinės būsenos atnaujinimo formulė:

$$C_t = F_t \odot C_{t-1} + I_t \odot \widetilde{C}_t \tag{2.38}$$

Paslėptos būsenos atnaujinimo formulė:

$$H_t = O_t \odot tanh(C_t) \tag{2.39}$$



14 pav. LSTM modelio paslėptos būsenos ir atminties ląstelės vidinės būsenos atnaujinimo grafikas.

2.6. Garch modelis

Garch yra statistinis modelis pristatytas T. Bollerslev 1986 metais. Šis modelis yra plačiai naudojamas modeliuoti kintamumą laiko eilutėse [10, 14].

GARCH(p, q) modelio formulė:

$$\sigma^2 = \omega + \alpha_1\epsilon_{t-1}^2 + ... + \alpha_q\epsilon_{t-q}^2 + \beta_1\sigma_{t-1}^2 + ... + \beta_p\sigma_{t-p}^2 = \omega + \sum_{i=1}^{q}\alpha_i\epsilon_{t-i}^2 + \sum_{i=1}^{p}\beta_i\sigma_{t-i}^2 \quad (2.40)$$

[14] šaltinyje buvo tiriamas įvairių GARCH šeimos modelių tikslumas, bandant prognozuoti akcijų rinkos indeksų grąžas. Taip pat šiame, šaltinyje buvo tiriami tik GARCH(1, 1) šeimos modeliai, teigiant, kad maži p ir q parametrai yra pakankami modeliuoti ilgai trunkančius dispersijos kitimo procesus. Dėl šios priežasties tolimesniame darbe nuspręsta lyginti rezultatus tik su GARCH(1, 1) apmokytų modelių rezultatais.

3. Programinės realizacijos detalės

Visi eksperimentai buvo rašomi su Python [7] programavimo kalba, tačiau kai kuriais atvejais, kai reikdavo valdyti nuotolinius serverius buvo naudojama ir bash [2] programavimo kalba.

Visų pirma, S&P500 [8] duomenys buvo paruošiami apmokymams. Tai yra, ieškoma trūkstamų reikšmių (akcijų uždarymo kainų), tada trūkstamos reikšmės buvo interpoliuojamos ir iš akcijų uždarymo kainų skaičiuojamos logaritminės grąžos. Kadangi duomenys buvo įrašyti CSV formatu, jų nuskaitymui buvo naudojama Pandas [26] biblioteka. Su ta pačia biblioteka buvo interpoliuojamos trūkstamos reikšmės. Trūkstamų reikšmių interpoliavimas atliktas tiesiniu būdu. Užpildžius visas trūkstamas reikšmes buvo skaičiuojamas kintamumas. Kintamumo skaičiavimui panaudota ta pati Pandas [26] biblioteka. Tai atlikus, buvo ieškomi nuokrypiai. Nuokrypių paieškai buvo panaudota Z rezultatų taisyklė, kuri buvo aprašyta panaudojant Numpy [20] biblioteką. Nuokrypių reikšmės buvo pakeičiamos didžiausiomis ne nuokrypiu laikomomis reikšmėmis. Tai atlikta naudojant Pandas [26] biblioteką. Galiausiai kintamumai buvo sugrupuojami į rėžius panaudojant Pandas [26] ir Numpy [20] bibliotekas.

Paruošus duomenis, buvo sudaromos rekurentinės diagramos. Tam, kad būtų sudarytos šios diagramos, akcijų kintamumo laiko eilutėms reikėjo pritaikyti slenkančio lango principą. Slenkančio lango principas buvo pritaikytas panaudojant Pandas [26] biblioteką. Tada buvo sugeneruotos rekurentinės diagramos. Diagramų sudarymui buvo panaudota PyRQA [31] biblioteka. Su ta pačia biblioteka buvo ir apskaičiuojamos RQA metrikos. Šių metrikų išsaugojimui į CSV failus buvo panaudota Pandas [26] biblioteka.

Turint rekurentines diagramas, toliau duomenys buvo padalinami į mokymo, testavimo ir validavimo aibes. Dalinant duomenis išlaikymo (angl. hold-out) būdu, kiekvienos akcijos kintamumų laiko eilutės buvo išrikiuojamos pagal datą. Tada testavimui atidedama 10% naujausių duomenų, validavimui 20% likusių naujausių duomenų, o likę duomenys buvo atidedami apmokymui. Visos rekurentinės diagramos buvo laikomos vienoje failų direktorijoje, o nuorodos į rekurentines diagramas, reikalingas RQA metrikas ir kintamumo istorijas buvo išsaugomi testavimui, validavimui ir apmokymui skirtuose CSV failuose. Dalinant duomenis 5 dalių kryžminės validacijos būdu, 20% naujausių kiekvienos akcijos kintamumo duomenų buvo atidedama testavimui. Tada likusios akcijos buvo padalintos į 5 dalis, kurių viena buvo naudojama validavimui, o likusios 4 testavimui. Toks padalinimas buvo atliktas panaudojant scikit-learn [28] biblioteką. Kiekvienos dalies padalinimai buvo saugomi į atskirus testavimui, validavimui ir apmokymui reikalingus CSV failus (analogiškai išlaikymo validacijai reikalingu padalinimo būdu).

Padalinus duomenis, kintamumo istorijoms ir RQA metrikoms, reikalingoms apmokyti neuroninius tinklus, buvo pakeičiamas mastelis (angl. scale), pritaikant mažiausią, didžiausią mastelio keitimo taisyklę (angl. min-max scaling). Ši taisyklė buvo pritaikoma panaudojant scikit-learn [28] biblioteką, kuri sukurdavo objektą, galinti keisti mastelį programos vykdymo metu. Šio objekto išsaugojimui buvo panaudota joblib [5] biblioteka, o sukūrimui panaudoti tik mokymui reikalingi duomenys.

Po visų šių darbų, buvo pasiruošta apmokyti modelius. Pirma pradėta apmokyti ResNet50 konvoliucinius modelius. Šių modelių apmokymui buvo naudojama PyTorch [27] bilioteka. Šios bibliotekos pagalba buvo atsisiųsta ResNet50 modelio architektūra ir įkeliami ImageNet [4] duomenimis apmokyto modelio parametrai. Paskutinis modelio sluoksnis buvo pakeičiamas kitu, atitinkančiu užduoties klasifikavimą ir visi išskyrus paskutinį sluoksnį buvo užšaldomi. Tada modeliai buvo mokomi 200 epochų. Kiekvieno modelio būsenos buvo saugomos epochai pasibaigus, šitaip buvo galima nutraukti ir pratęsti mokymą. Po 200 epochų modeliai buvo tikrinima dėl persi-

mokymo. Tam buvo braižomi nuostolių ir tikslumo grafikai naudojant Matplotlib [23] biblioteka. Pastebėjus kiekvieno iš modelių apsimokymą, modelio sluoksniai būdavo atšaldomi ir modelis buvo mokomas dar 50 epochų sumažinus mokymosi žingsnį 10 kartų. Pirmasis mokymosi žingsnio dydis buvo atrenkamas eksperimentuojant. Be to, buvo naudojamas SGD optimizatorius (iš PyTorch [27] bibliotekos), su 0.0005 dydžio reguliarizacijos parametru.

Apmokius konvoliucinius neuroninius tinklus, buvo mokomi rekurentiniai neuroniniai tinklai. Skirtingai nei mokant konvoliucinius neuroninius tinklus, šiems modeliams buvo naudojamas Adam optimizatorius ir OneCycleLR planuotojas (abu paimti iš PyTorch [27] bibliotekos). Šie modeliai buvo mokomi 100 epochų. Po 100 epochų kiekvienas iš modelių buvo įvertinamas ir sprendžiama ar jam dar reikia mokytis.

Apmokius RNN modelius buvo mokomi GARCH(1, 1) modeliai. Šių modelių apmokymui buvo panaudota arch [1] biblioteka.

Visų modelių rezultatų įvertinimui buvo skaičiuojamos F1, REC, ACC, PREC metrikos. Tam atlikti buvo naudojama scikit-learn [28] biblioteka. Taip pat braižomi grafikai su Matplotlib [23] biblioteka.

Visi veiksmai išskyrus konvoliucinių neuroninių tinklų apmokymus buvo atliekami su kompiuteriu turinčiu debian [3] operacinę sistemą, ryzen 5 3600 CPU ir GTX 1660S vaizdo plokštę. ResNet50 modelio apmokymas atliekant kryžminę validaciją buvo atliktas naudojant MIF VU Paskirstytų skaičiavimų tinklą (PST) [6]. ResNet50 modeliai apmokyti su išlaikymo validacijai reikalingu duomenų padalinimo būdu, buvo apmokyti nuomojantis skaičiavimo resursus iš vast.ai [9]. Visi išsinuomoti skaičiavimo resursai turėjo RTX 3060 vaizdo plokštę.

Šio darbo metu papildomai buvo sugeneruoti dirbtiniai duomenys. Atsitiktinių signalų sudarymui buvo panaudotos Numpy [20] ir Matplotlib [23] bibliotekos. Duomenų paruošimas ir modelių apmokymas buvo analogiškas finansinių duomenų paruošimui.

4. Duomenys ir jų paruošimas

Šiam darbui atlikti buvo paimti duomenys iš *kaggle.com* [8]. Duomenų šaltinis sudarytas iš S&P 500 akcijų indekso kainų kitimų kiekvieną dieną. Pagrindinė informacija, kuri buvo panaudota iš šio duomenų šaltinio yra: diena ir akcijos kaina, biržai užsidarius (angl. *closing price*). Duomenys buvo paimti už 5 metus (nuo 2013-02-08 iki 2018-02-07).

4.1. Trūkstamų reikšmių paieška

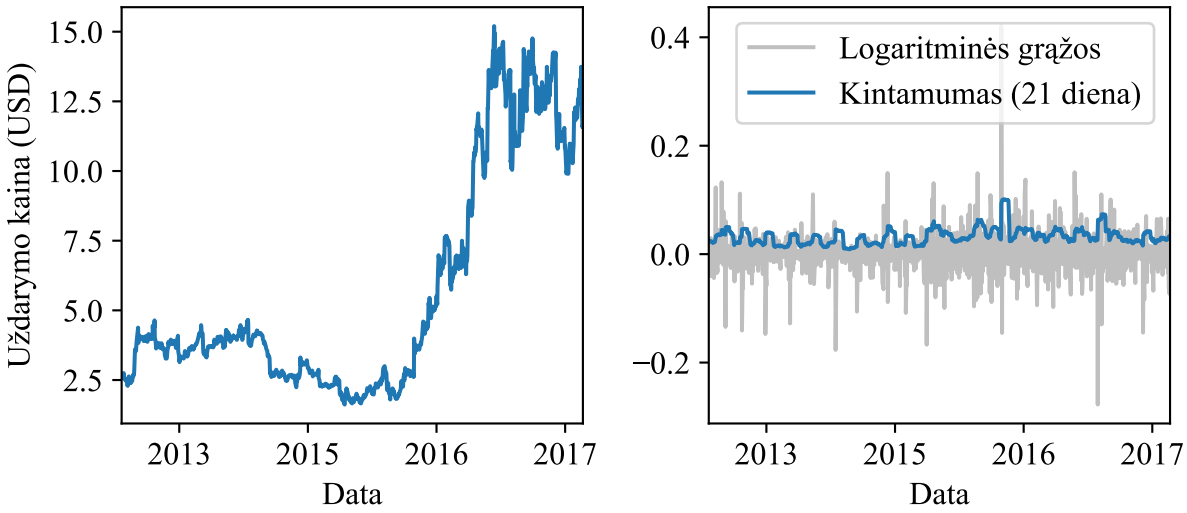
Prieš pradedant skaičiuoti akcijų kainų kintamumus, visų pirma, yra svarbu įsitikinti, kad nėra trūkstamų duomenų. Kadangi dirbama su laiko eilutėmis, tokių duomenų trūkumas, nesukels pastebimos programinės klaidos, tačiau gali paveikti rezultatus.

Siekiant surasti visas trūkstamas reikšmes, buvo sudarytas dienų sąrašas, kai buvo parduodama bent viena akcija. Tuomet, kiekvienos akcijos kainų kitimų istorijoje, buvo ieškoma tokių dienų, kai nebuvo duomenų apie tikrinamos akcijos pardavimus, nors buvo duomenų apie kitų akcijų pardavimus (dienų paieška buvo daroma tarp tikrinamos akcijos pirmos ir paskutinės turimos dienos).

Iš viso buvo rasta, kad trūksta informacijos 51 kartą. Tai sudaro 0.008% visų duomenų. Trūkstamiems duomenims užpildyti buvo panaudota tiesinė interpoliacija. Buvo bandyta pritaikyti ir kubinę interpoliaciją, tačiau naudojant pastarąją buvo gaunamos neigiamos reikšmės. Neigiamų uždarymo kainų negali būti, nes akcijų uždarymo kaina visada yra teigiama.

4.2. Kintamumo apskaičiavimas

Duomenų šaltinyje pateiktos kiekvienos akcijos uždarymo kainos kiekvieną dieną. Iš šių kainų laiko eilučių, kiekvienai akcijai buvo apskaičiuotos logaritminės grąžos. Vėliau iš logaritminių grąžų laiko eilučių, kiekvienai akcijai buvo apskaičiuotas 21 dienos kintamumas. Iš viso buvo gauta 608486 įrašų su logaritminėmis grąžomis.



15 pav. AMD akcijų uždarymo kainų istorija (iliustracija kairėje) ir AMD akcijų uždarymo kainų logaritminės grąžos su 21 dienos kintamumu (iliustracija dešinėje).

4.3. Nuokrypiai

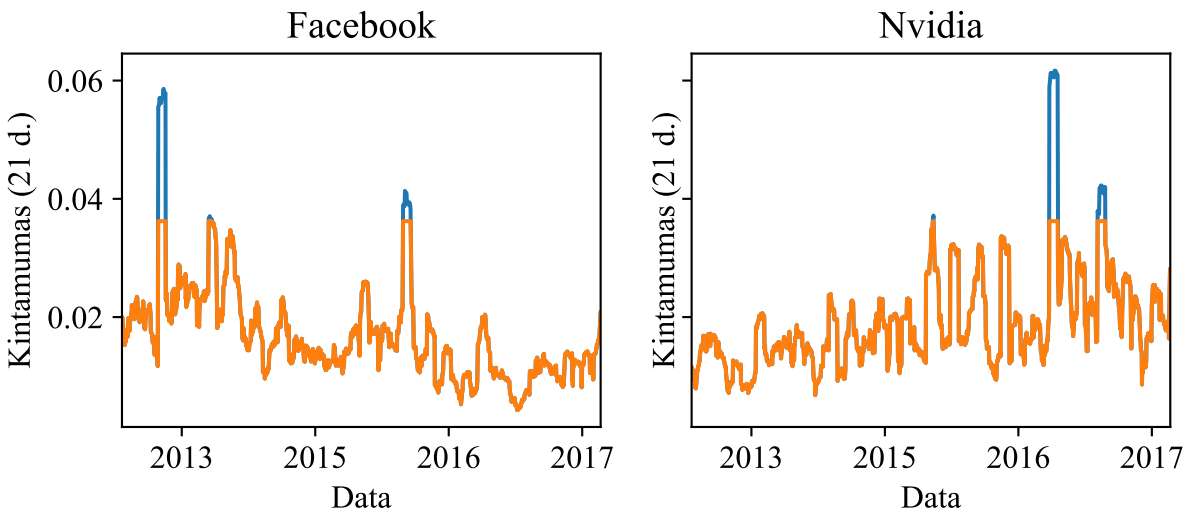
Nuokrypiai (angl. *outliers*) tai įrašai duomenyse, kurie smarkiai skiriasi nuo kitų įrašų. Tokie įrašai gali būti klaidos arba gali būti registruoti, vykstant išskirtinėms sąlygoms. Modeliai sunkiai išmoksta atpažinti tokius įrašus, todėl svarbu atpažinti nuokrypius [32].

Vienas iš būdų atpažinti nuokrypius yra naudotis Z rezultatų (angl. z-scores) atpažinimo taisyklę:

$$z_i = \frac{x_i - \overline{x}}{s} \tag{4.1}$$

s – standartinis nuokrypis, $\overline{x}$ – vidurkis, x_i – įrašo vertė. Įrašas pripažįstamas nuokrypiu, jei $|z_i| > 3$.

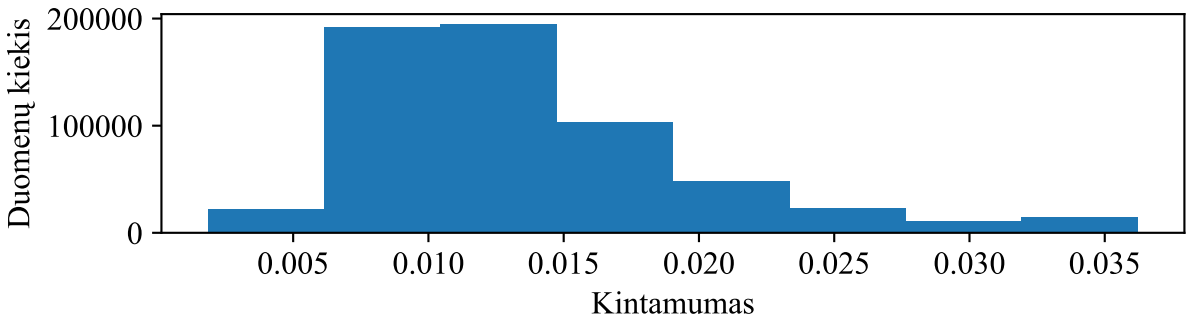
Šio darbo metu pritaikius Z rezultatų atpažinimo taisyklę buvo rasti nuokrypiai, kurie sudarė 8723 įrašus iš 608486. Tai yra 1.43% visų duomenų. Nuokrypiai buvo pakeisti didžiausia reikšme, kuri dar nėra priskiriama nuokrypiams – 0.036. Taip buvo pasirinkta, tam kad išlaikyti kintamumo svyravimus ir apriboti galimus kintamumo rėžius. Pakeitus nuokrypius galimo kintamumo rėžiai pasikeitė iš $[0.002; 0.213]$ į $[0.002; 0.036]$. Toks kintamumo svyravimo rėžio sumažinimas palengvins modelių apmokymą.



16 pav. Facebook ir Nvidia akcijų kintamumo ir kintamumo pašalinus nuokrypius grafikai. Mėlyna kreivė žymi kintamumą, o oranžinė kreivė žymi kintamumą pašalinus nuokrypius.

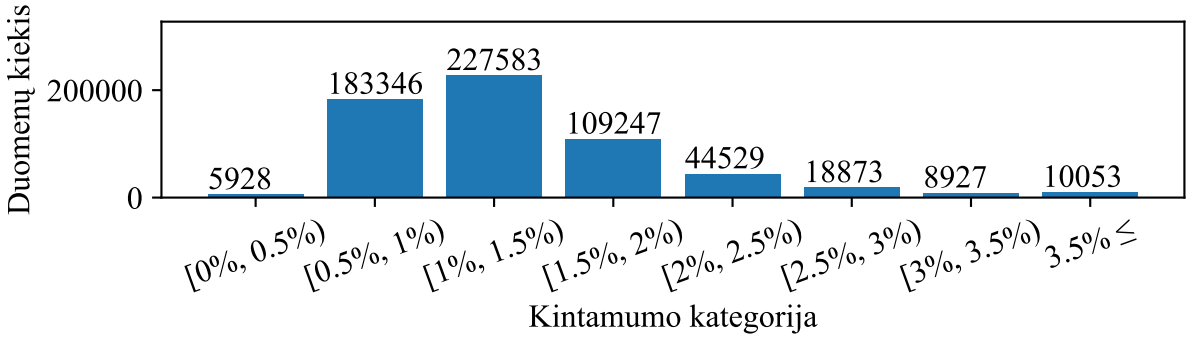
4.4. Kategorijos

Šiame darbe buvo prognozuojama ne kintamumo reikšmė, bet rėžis. Norint tinkamai pasirinkti kintamumo rėžius reikia gerai suprasti duomenis. Taip yra dėl to, kad blogai pasirinkus modelio spėjimo rėžius, gali būti tokių klasių, kurių duomenų tiesiog nebus. Dėl šios priežasties, buvo nubraižyta kintamumo pasiskirstymo histograma (17 pav.).



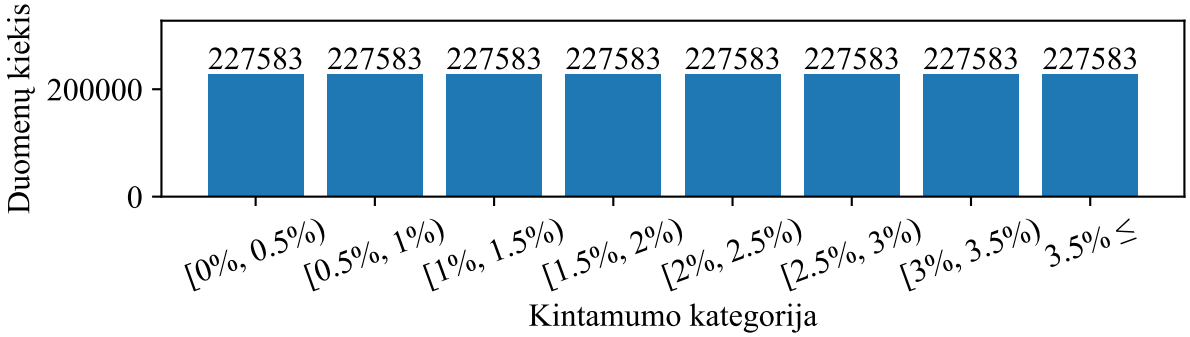
17 pav. Kintamumo pasiskirstymo histograma.

Atsižvelgiant į duomenų pasiskirstymą, matomą histogramoje, buvo nuspręsta sudaryti 8 kategorijas: $[0\%, 0.5\%)$, $[0.5\%, 1\%)$, $[1\%, 1.5\%)$, $[1.5\%, 2\%)$, $[2\%, 2.5\%)$, $[2.5\%, 3\%)$, $[3\%, 3.5\%)$ ir $3.5 \leqslant$.



18 pav. Kintamumo kategorijų duomenų kiekio diagrama.

Iš 18 paveikslėlio galima įžvelgti, kad duomenyse vyrauja kategorijų disbalansas. Kategorijų disbalansas yra blogas reiškinys, nes modeliai išmoksta atpažinti vienas kategorijas geriau nei kitas. Šio darbo metu, siekiant išspręsti disbalanso problemą buvo naudojama išplėtimo (angl. oversample) metodika. Ši metodika, reiškia, kad kategorijų, kurios turi mažesnį kiekį narių, nei daugiausiai narių turinti kategorija, nariai bus dubliuojami, kol visų kategorijų narių kiekis bus vienodas (19 pav.).

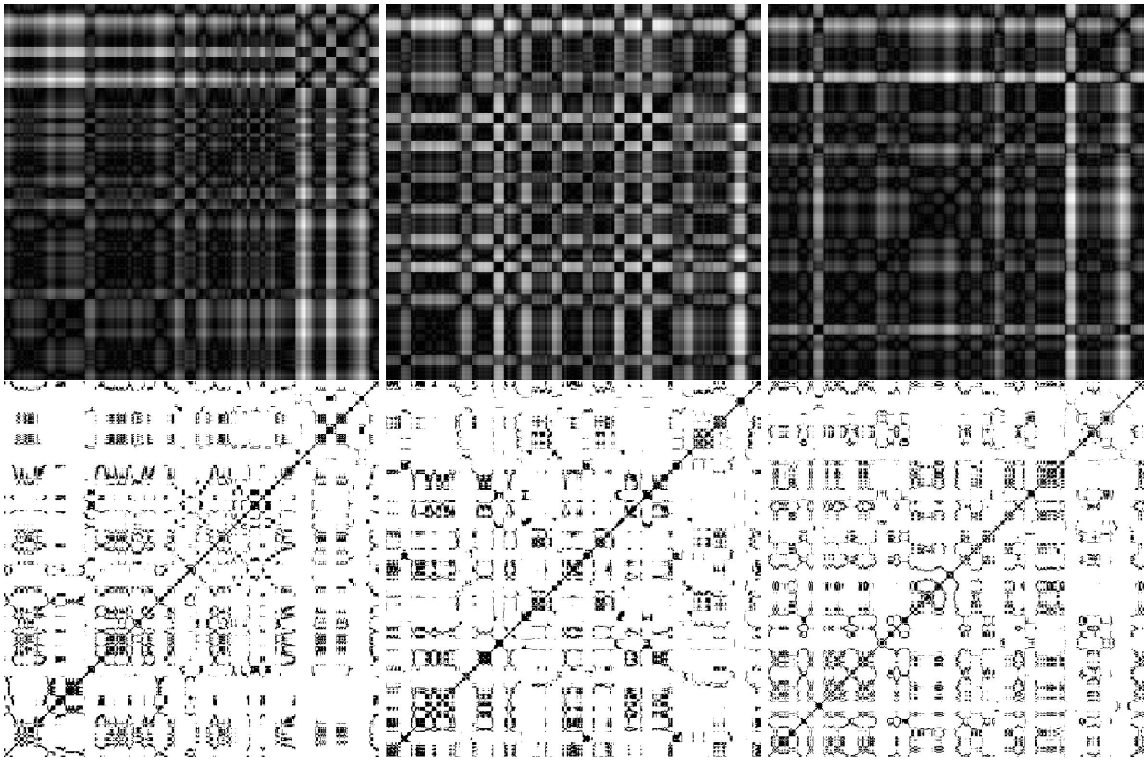


19 pav. Kintamumo kategorijų duomenų kiekio diagrama po išplėtimo.

4.5. Rekurentinės diagramos

Iš akcijų uždarymo kainų kintamumo, buvo sudarytos laiko eilutės. Iš laiko eilučių slenkančio lango būdu buvo sudaryti nauji duomenų rinkiniai. Lango dydis N buvo pasirinktas 780 darbo dienų. Toks pasirinkimas priimtas, dėl to, kad per metus yra vidutiniškai 260 darbo dienų, o akcijomis „NYSE“ prekiauja, tik darbo dienomis. Autoriaus nuomone, 3 metų dydžio slenkantis langas yra pakankamai didelis prognozuoti kintamumui. Lango žingsnis pasirinktas s=5. Iš kiekvieno gauto duomenų rinkinio buvo sudarytos dviejų tipų rekurentinės diagramos: spalvinės (Fig. 20) ir klasikinės išryškinant 10% juodų pikselių. Taip pat klasikinėms rekurentinėms diagramoms buvo paskaičiuotos RQA metrikos.

Iš viso buvo sugeneruota 44415 spalvinių ir 44415 klasikinių rekurentinių diagramų.



20 pav. Spalvinių ir klasikinių rekurentinių diagramų pavyzdžiai: Nvidia kompanijos akcijų rekurentinės diagramos (iliustracijos kairėje), Amazon kompanijos akcijų rekurentinės diagramos (iliustracijos per vidurį) ir Starbux kompanijos akcijų rekurentinės diagramos (iliustracijos dešinėje). Viršuje vaizduojamos spalvinės rekurentinės diagramos, o apačioje klasikinės išryškinant 10% visų juodų pikselių.

5. Modelių apmokymo aspektai

5.1. Konvoliuciniai neuroniniai tinklai

5.1.1. Įvestis ir žymėjimas

Konvoliucinių neuroninių tinklų modelių apmokymui buvo paruošti 4 tipų duomenys:

- 780 dienų (3 metų) akcijų uždarymo kainų kintamumų laiko eilutės.
- Spalvinės rekurentinės diagramos.
- Klasikinės rekurentinės diagramos išryškinančios 10% juodų pikselių.
- Iš klasikinių rekurentinių diagramų paskaičiuotos RQA metrikos.

Šiuos duomenis tarpusavyje galima kombinuoti ir juos panaudoti apmokant neuroninius tinklus. Šio darbo metu buvo tirtos šios duomenų įvestys:

1. Prognozuojantys kintamumą tik iš spalvinių rekurentinių diagramų (toliau žymimi $_{urp}$ apatiniu indeksu).
2. Prognozuojantys kintamumą iš spalvinių rekurentinių diagramų ir 780 dienų kintamumo laiko eilučių istorijos (toliau žymimi $_{urp+780d}$ apatiniu indeksu).
3. Prognozuojantys kintamumą iš spalvinių rekurentinių diagramų ir paskutinės dienos kintamumo (toliau žymimi $_{urp+1d}$ apatiniu indeksu).
4. Prognozuojantys kintamumą iš klasikinių rekurentinių diagramų išryškinant 10% juodų pikselių (toliau žymimi $_{rp10}$ apatiniu indeksu).
5. Prognozuojantys kintamumą iš klasikinių rekurentinių diagramų išryškinant 10% juodų pikselių ir RQA metrikų (toliau žymimi $_{rp10+rqa}$ apatiniu indeksu).
6. Prognozuojantys kintamumą iš klasikinių rekurentinių diagramų išryškinant 10% juodų pikselių ir 780 dienų kintamumo laiko eilučių istorijos (toliau žymimi $_{rp10+780d}$ apatiniu indeksu).
7. Prognozuojantys kintamumą iš klasikinių rekurentinių diagramų išryškinant 10% juodų pikselių, RQA metrikų ir 780 dienų kintamumo laiko eilučių istorijos (toliau žymimi $_{rp10+rqa+780d}$ apatiniu indeksu).
8. Prognozuojantys kintamumą iš klasikinių rekurentinių diagramų išryškinant 10% juodų pikselių, RQA metrikų ir paskutinės dienos kintamumo (toliau žymimi $_{rp10+rqa+1d}$ apatiniu indeksu).

5.1.2. Įvesties pasirinkimas

Modelio įvesties pasirinkimas yra svarbus uždavinys. Skirtingos įvestys turi skirtingą informaciją ir tai gali daryti įtaką modelio tikslumui. Dėl šios priežasties yra svarbu ištirti kokios įvestys padeda apmokyti tiksliausius modelius. Tam atlikti buvo ištirtos visos 5.1.1 skyrelyje aprašytos įvestys.

Vienas iš paprasčiausių būdų kaip galima būtų rasti, kurios įvestis leidžia apmokyti tiksliausius modelius, yra apmokyti skirtingus modelius ir įvertinti jų tikslumą. Nutarta adaptuoti ResNet50 [22] modelį. Visos įvestys pažymėtos 1 lentelėje.

1 lentelė. CNN modelių žymėjimo lentelė.

Žymėjimas	Pagrindinis modelis	Rekurentinės diagramos	RQA	780 d. kintamumo istorija	Paskutinės dienos kintamumas
$ResNet_{urp}$	ResNet50	Spalvinės			
$ResNet_{urp+780d}$	ResNet50	Spalvinės		✓	
$ResNet_{urp+1d}$	ResNet50	Spalvinės			✓
$ResNet_{rp10}$	ResNet50	Klasikinės			
$ResNet_{rp10+rqa}$	ResNet50	Klasikinės	✓		
$ResNet_{rp10+780d}$	ResNet50	Klasikinės		✓	
$ResNet_{rp10+rqa+780d}$	ResNet50	Klasikinės	✓	✓	
$ResNet_{rp10+rqa+1d}$	ResNet50	Klasikinės	✓		✓

Visų įvesčių tyrimas su finansiniais duomenimis užtruktų daug laiko (iš finansinių duomenų yra sudaryta 44 415 rekurentinių diagramų), dėl to nuspręsta sukurti atskirą, lengvai prognozuojamą dirbtinių duomenų šaltinį.

Duomenų sudarymas Dirbtiniams duomenims sudaryti buvo sugeneruoti 50 signalų, panaudojant formulę:

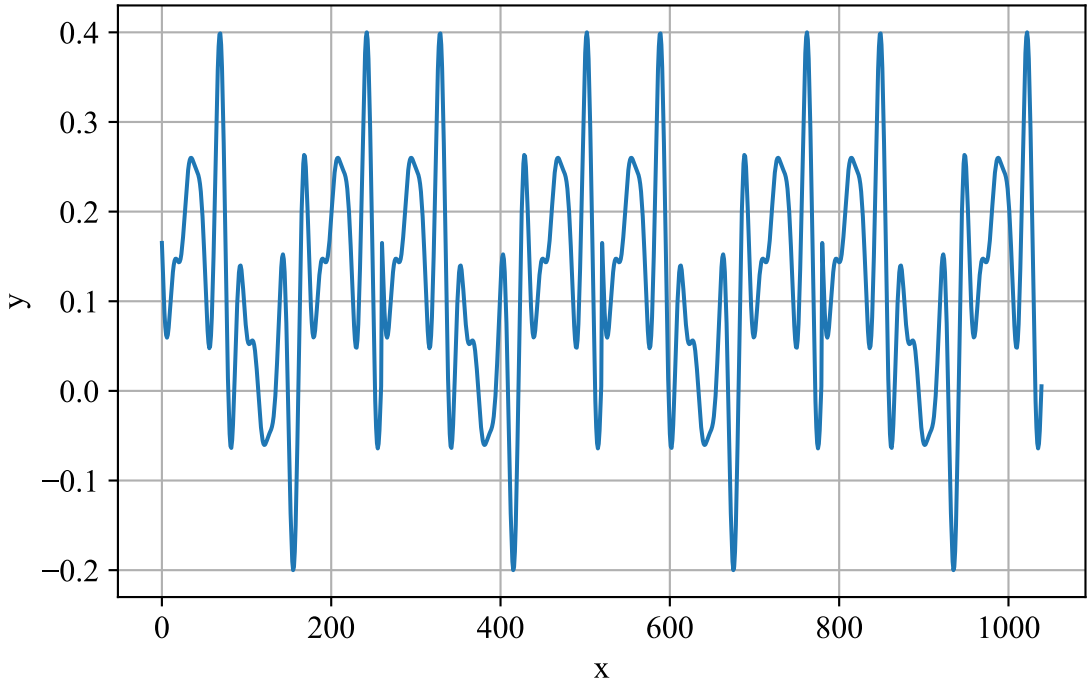
$$\begin{aligned} &f(x) = 3x\pi, && x \in [0;1) \\ &y = f(x) + c_0 f(3x + c_1) + c_2 f(5x + c_3) + c_4 f(7x + c_5) + c_6 f(9x + c_7), && c \in [-1;1) \end{aligned} \tag{5.1}$$

Signalus koeficientai c buvo atsitiktinai paimami iš intervalo $[-1; 1)$.

Iš sugeneruotų signalų buvo paimtos 260 reikšmių, nes vidutiniškai tiek dienų akcijų biržos dirba per metus. Po to, gautų y reikšmių skalė buvo paverčiama į intervalą $y \in [0; 0.5]$ panaudojant mažiausią-didžiausią normalizaciją:

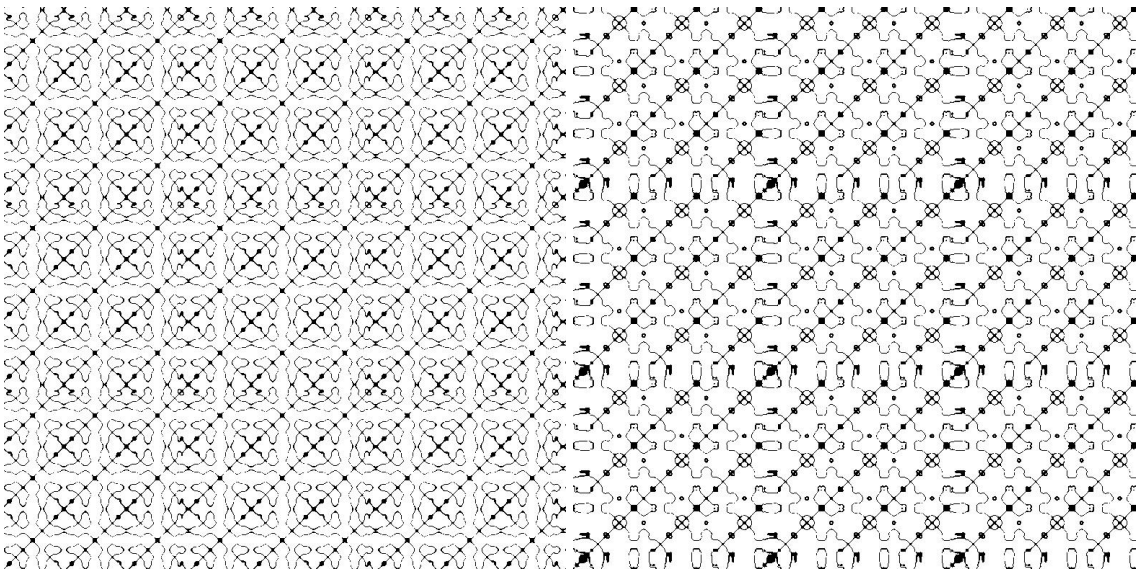
$$y' = \frac{y - min(y)}{max(y) - min(y)} \tag{5.2}$$

Gauti signalai buvo pakartojami dar 4 kartus, taip sudarantys atsikartojimus. Periodiniai atsikartojimai yra lengvai prognozuojami ir taip patys signalai tampa **lengvai prognozuojami**.



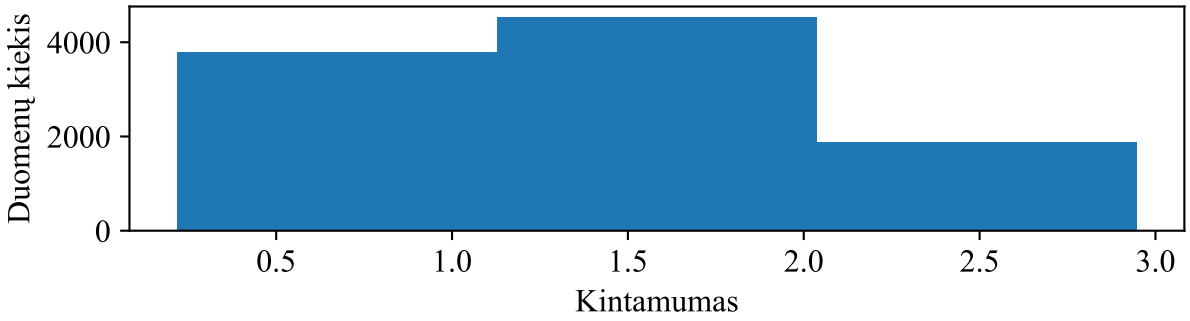
21 pav. Atsitiktinai sugeneruoto signalo pavyzdys, kur $c_0 = -0.07$, $c_1 = -0.97$, $c_2 = -0.84$, $c_3 = 0.52$, $c_4 = 0.74$, $c_5 = -0.34$, $c_6 = 0.42$, $c_7 = -0.91$.

Rekurentinės diagramos Taip pat kaip ir su finansiniais duomenimis, kiekvienam signalui buvo apskaičiuotas kintamumas, sudarytos spalvinės ir klasikinės, išryškinant 10% juodų pikselių, rekurentinės diagramos bei apskaičiuotos RQA metrikos. Skirtingai nei iš finansinių duomenų sudarytose rekurentinėse diagramose, iš dirbtinių duomenų sudarytose rekurentinėse diagramose išryškėja įstrižainės linijos identifikuojančios apie signale esančius atsikartojimus (22 pav.).



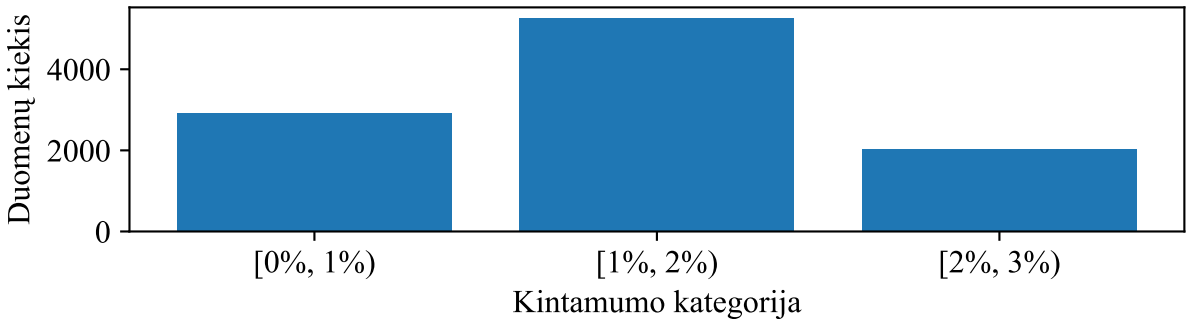
22 pav. Iš dirbtinai sugeneruotų signalų sudarytos rekurentinės diagramos. Įstrižainės linijos parodo, kad signale egzistuoja atsikartojimai.

Kategorijos Kaip ir finansiniuose duomenyse, dirbtiniai duomenys buvo paruošti prognozuoti kintamumo rėžius, todėl buvo nubraižoma kintamumo pasiskirstymo histograma (23 pav.).



23 pav. Kintamumo pasiskirstymo histograma.

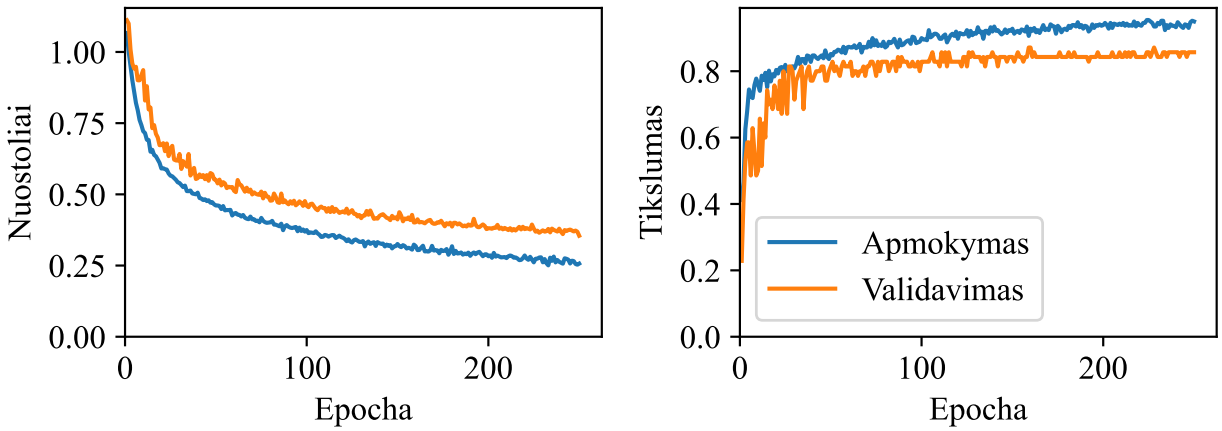
Atsižvelgiant į duomenų pasiskirstymą, matomą histogramoje, buvo nuspręsta sudaryti 3 kategorijas: $[0\%, 1\%)$, $[1\%, 2\%)$, $[2\%, 3\%)$.



24 pav. Dirbtinių duomenų kintamumo kategorijų duomenų kiekio diagrama. Iš skirtingo kiekio kategorijų galima įžvelgti nežymų disbalansą.

Modelių apmokymas Siekiant ištirti kaip modeliai geba prognozuoti tolimesnes reikšmes kiekvienas iš modelių aprašytų 1 lentelėje buvo apmokomas prognozuoti kitą laiko eilutės reikšmę ir 21-ąją laiko eilutės reikšmę. Laiko eilutės numerio prognozavimas žymimas viršutiniu indeksu. Pavyzdžiui, $ResNet^{1}_{urp}$ – žymės modelį prognozuojantį kitą laiko eilutės reikšmę, o $ResNet^{21}_{urp}$ – žymės modelį prognozuojantį 21-ąja laiko eilutės reikšmę.

Kiekvienas iš modelių buvo apmokomas jam priskirtais duomenų rinkiniais (lentelė 1), be to, duomenys buvo padalinti panaudojant k dalių kryžminę validaciją (plačiau 2.4.1 skyrelyje). Visų signalų 20% duomenų buvo atidėta kaip testavimui skirta aibė. Likę duomenys buvo padalinti į 5 dalis po 20%. Viena dalis buvo panaudota modelio validavimui, o likusios 4 jo apmokymui. Kiekvienas iš modelių buvo adaptuojamas, tai reiškia, kad visi sluoksniai išskyrus paskutinį buvo užšaldyti. Modeliai buvo apmokomi iki pastebimo apsimokymo. Modelių apsimokymo įvertinimui buvo braižomi nuostolių ir tikslumo grafikai (25 pav.).



25 pav. $ResNet^{1}_{urp+780d}$ modelio mokymo bei validavimo nuostolių grafikas (iliustracija kairėja) bei to paties modelio mokymo bei validavimo tikslumo grafikas (iliustracija dešinėje). Iš nusistovėjusio validavimo tikslumo galima įžvelgti, kad modelis apsimokė.

Rezultatai Kiekvienas iš modelių buvo įvertintas ACC, PREC, REC ir F1 metrikomis. Taip pat, kadangi buvo atliekama 5 dalių kryžminė validacija, buvo apskaičiuoti šių metrikų vidurkiai ir standartiniai nuokrypiai. Iš 2 lentelėje esančių rezultatų, F1 metrikų, galima padaryti išvadą, kad tiksliausi modeliai yra: $ResNet_{urp+780d}$, $ResNet_{rp10+780d}$ ir $ResNet_{rp10+rqa+780d}$. Kadangi jų rezultatai labai panašūs, nuspręsta paimti po vieną modelį naudojantį klasikines ir spalvines rekurentines diagramas, todėl finansiniais duomenimis buvo apmokomi tik $ResNet_{urp+780d}$ ir $ResNet_{rp10+rqa+780d}$ modeliai.

2 lentelė. CNN dirbtiniais duomenimis apmokytų modelių rezultatai.

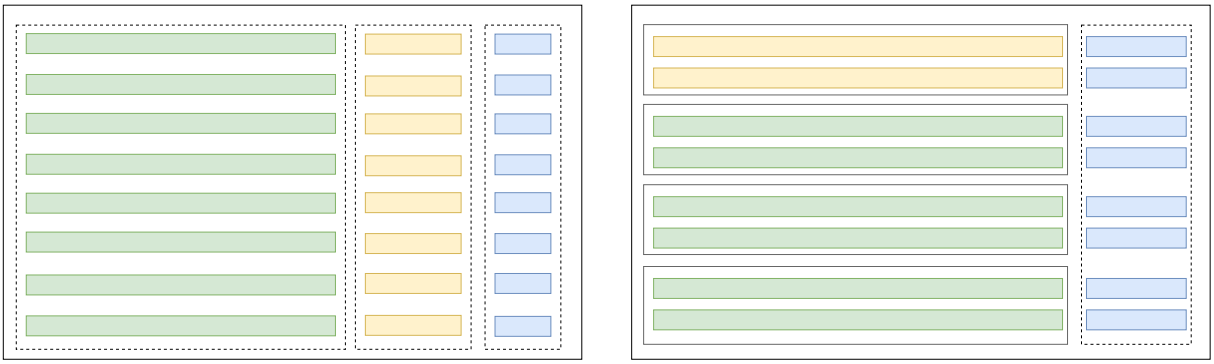
	Vidurkis				Standartinis nuokrypis			
Modelis	ACC	PREC	REC	F1	ACC	PREC	REC	F1
$ResNet^{1}_{urp}$	0.44	0.46	0.50	0.44	0.02	0.02	0.02	0.02
$ResNet^{1}_{urp+780d}$	0.84	0.82	0.88	**0.84**	0.03	0.03	0.03	0.03
$ResNet^{1}_{urp+1d}$	0.46	0.48	0.53	0.46	0.02	0.04	0.05	0.02
$ResNet^{1}_{rp10}$	0.44	0.49	0.52	0.44	0.04	0.04	0.04	0.05
$ResNet^{1}_{rp10+rqa}$	0.45	0.49	0.54	0.45	0.04	0.03	0.03	0.04
$ResNet^{1}_{rp10+780d}$	0.85	0.82	0.89	**0.84**	0.02	0.02	0.01	0.02
$ResNet^{1}_{rp10+rqa+780d}$	0.85	0.83	0.89	**0.85**	0.02	0.02	0.02	0.02
$ResNet^{1}_{rp10+rqa+1d}$	0.48	0.52	0.57	0.49	0.02	0.03	0.02	0.02
$ResNet^{21}_{urp}$	0.44	0.46	0.50	0.44	0.02	0.02	0.02	0.02
$ResNet^{21}_{urp+780d}$	0.84	0.82	0.88	**0.84**	0.02	0.02	0.03	0.02
$ResNet^{21}_{urp+1d}$	0.45	0.49	0.54	0.46	0.01	0.04	0.03	0.01
$ResNet^{21}_{rp10}$	0.44	0.49	0.54	0.45	0.04	0.06	0.04	0.04
$ResNet^{21}_{rp10+rqa}$	0.46	0.50	0.54	0.47	0.04	0.05	0.05	0.04
$ResNet^{21}_{rp10+780d}$	0.83	0.81	0.88	**0.83**	0.01	0.01	0.01	0.01
$ResNet^{21}_{rp10+rqa+780d}$	0.84	0.83	0.90	**0.85**	0.02	0.02	0.02	0.02
$ResNet^{21}_{rp10+rqa+1d}$	0.44	0.48	0.52	0.45	0.04	0.05	0.04	0.04

5.1.3. Duomenų padalinimo pasirinkimas

Vienas iš uždavinių prieš pradedant apmokyti modelius yra padalinti duomenis į testavimo, validavimo ir apmokymo aibes. Yra įprasta, modelius apmokant, naudoti k dalių kryžminė validaciją arba išlaikymo validavimą. [38] šaltinyje minima, kad modeliai apmokyti k dalių išlaikymo validacijos būdu pasiekia geresnius rezultatus, tačiau jų apmokymas trunka ilgiau. Yra verta ištirti šį teiginį su turimais duomenis, apmokant vieną modelį skirtingais padalinimo būdais. Pasirinkta apmokyti $ResNet^1_{urp+780d}$ modelį, nes su dirbtinais duomenimis šis modelis pasiekė vienus iš didžiausių tikslumų.

Modelio apmokymui, išlaikymo validvimo būdu, duomenys buvo padalinti į tris dalis: testavimui atidėta 10% visų akcijų naujausių duomenų, validavimui atidėta likę 20% naujausių duomenų ir likę 60% buvo panaudoti modelio apmokymui. Naujausių duomenų atidėjimas testavimui buvo atliktas siekiant įvertinti kaip gerai modelis prognozuoja naujausius, nematytus duomenis. Toks modelio padalinimas atvaizduotas 26 pav. (iliustracija kairėje).

Modelio apmokymui k dalių kryžminės validacijos būdu, buvo pasirinkta naudoti 5 dalis, nes tokia rekomendacija aprašyta [38], kai duomenų kiekis yra tarp 10000 ir 100000. Testavimui buvo atidėta 20% visų naujausių duomenų, nes norima įvertinti kaip modelis geba prognozuoti ateities duomenis. Modelio padalinimas atvaizduotas 26 pav. (iliustracija dešinėje).



26 pav. Išlaikymo padalinimas, padalinant duomenis 60/30/10 (iliustracija kairėje) ir 4 dalių kryžminė validacija (iliustracija dešinėje), kai 20% naujausių duomenų yra atidedama testavimui. Spalvoti kvadratėliai atvaizduoja akcijų kainų kintamumo laiko eilutes, horizontaliai išdėstyti kvadratėliai žymi vienos akcijos kintamumo istoriją. Žali kvadratėliai tai duomenys naudojami modelius apmokyti, geltoni modelių tikslumui validuoti, mėlyni – įvertinti galutiniam tikslumui (testavimui).

Apmokius $ResNet^1_{urp+780d}$ modelį abiem būdais (rezultatai 3 lentelėje) buvo pastebėta, kad modelis apmokytas kryžminės validacijos būdu yra 2% tikslesnis, nei išlaikymo būdu. Autoriaus nuomone toks rezultatas paaiškinamas tuo, kad kryžminės validacijos būdu apmokytas modelis turėjo 10% daugiau naujesnių duomenų nei išlaikymo būdu. Visgi, nors ir kryžminė validacija suteikia tikslesnius rezultatus, modelio apmokymas trunka 5 kartus ilgiau. Vienos dalies apmokymas truko beveik 14 valandų, o visų kryžminės validacijos dalių apmokymas užtruko beveik 70 valandų. Naudojant išlaikymo validacija, apmokyti modelį užtruko šiek tiek mažiau nei 15 valandų. Kadangi apmokyti modelius naudojant kryžminę validaciją trunka labai ilgai, dėl to nuspręsta šio darbo metu naudoti išlaikymo validaciją.

3 lentelė. $ResNet^{1}_{urp+780d}$ modelio apmokyto skirtingai padalinus duomenis rezultatai.

	ACC	PREC	REC	F1
Kryžminio padalinimo dalis 1	0.53	0.39	0.43	0.38
Kryžminio padalinimo dalis 2	0.54	0.39	0.44	0.39
Kryžminio padalinimo dalis 3	0.55	0.40	0.44	0.40
Kryžminio padalinimo dalis 4	0.52	0.39	0.45	0.39
Kryžminio padalinimo dalis 5	0.53	0.39	0.44	0.39
Kryžminio padalinimo vidurkis	0.54	0.39	0.44	0.39
Kryžminio padalinimo standartinis nuokrypis	0.01	0.00	0.01	0.00
Išlaikymo padalinimas	0.52	0.38	0.42	0.37

5.1.4. Duomenų augmentacija

Duomenų augmentacija tai būdas išvengti modelių persimokymo. Šio darbo metu, duomenų augmentacija buvo naudojama apmokant konvoliucinius neuroninius tinklus.

Apžvelgus literatūros šaltinius buvo rastas rekurentinių diagramų augmentavimo būdas [13]. Prieš modeliui priimant rekurentines diagramas, judant palei centrinę įstrižainę, atsitiktinai iškerpama 760x760 rekurentinės diagramos dalis (27 pav.).

Autoriaus nuomone toks iškirpimas nesugadina diagramos savybių ir padeda išvengti persimokymo.

27 pav. Rekurentinės diagramos augmentacijos pavyzdys. Atsitiktinai iškerpamas 760x760 pikselių dydžio paveikslėlis, kuris būtinai yra centruotas pagal pagrindinę rekurentinės diagramos įstrižainę.

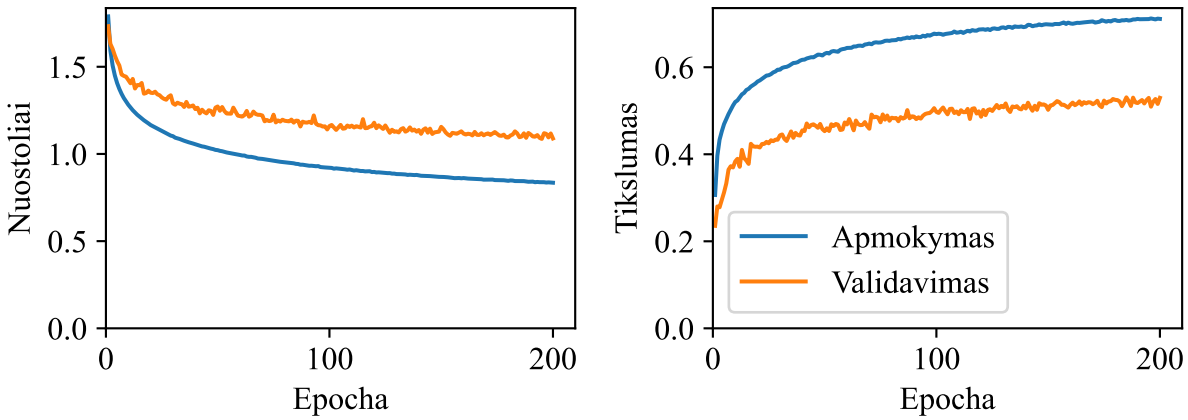
5.1.5. Apmokymas

Rementis 5.1.2 skirsnio rezultatais, finansiniais duomenimis buvo apmokyti tik $ResNet_{urp+780d}$ ir $ResNet_{rp10+rqa+780d}$ modeliai. Kiekvienas iš šių modelių buvo apmokytas prognozuoti kintamumo rėžį po 1, 2, 5, 10 ir 21 dienos. Be to, duomenys buvo padalinti į apmokymo, validavimo ir testavimo aibes, proporcijomis 70/20/10 (plačiau apie tai 5.1.3 skirsnyje).

ResNet50 paskutinis sluoksnis buvo pakeistas į pilnai sujungtų sąryšių sluoksnį, kuris gražina 8 reikšmes (nes tiek klasių prognozuojama). Šiam modeliui buvo perkelti apmokyto atpažinti

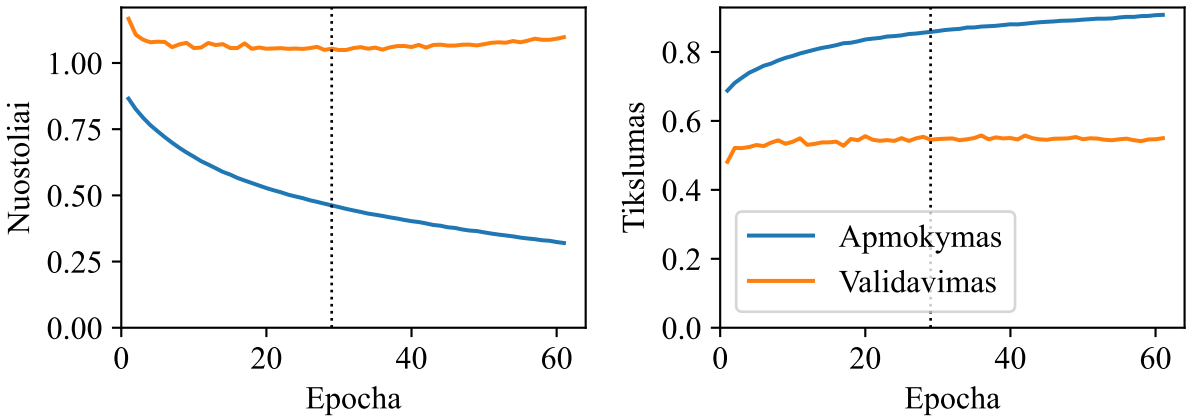
ImageNet paveiksliukų klasės svoriai. Tada visi apmokamo modelio svoriai buvo užšaldomi ir apmokomas tik paskutinis sluoksnis (plačiau apie adaptavimo procesą 2.5.6 skirsnyje).

Modeliai buvo apmokomi mažiausiai 200 epochų (28 pav.). Epocha tai ciklas, kurio metu modelis būna apmokomas visa mokymo aibe. Įvertinti modelių apsimokymui buvo braižomi nuostolių ir tikslumo grafikai. Pastebėjus nuostolių konvergavimą, modelis buvo laikomas apsimokiusiu ir jo mokymas nutrauktas.



28 pav. $ResNet^1_{urp+780d}$ apmokymo nuostolių ir tikslumo grafikai, kai visi išskyrus paskutinį sluoksnį buvo užšaldyti. Grafikuose matomas nusistovėjimas liudijantis apie modelio apsimokymą.

Apmokius paskutinį sluoksnį, modelio sluoksniai buvo atšaldomi ir jis dar kartą buvo apmokomas su 10 kartų mažesniu mokymosi žingsniu (29 pav.). Mokymas buvo daromas mažiausiai 50 kartų, tačiau jei būdavo pastebimas persimokymas (validavimo nuostolių grafikas imdavo didėti, kol apmokymo nuostolių grafikas mažėdavo) modelio apmokymas buvo stabdomas anksčiau laiko.



29 pav. $ResNet^1_{urp+780d}$ apmokymo nuostolių ir tikslumo grafikai, kai modelis buvo antrą kartą apmokomas atšaldžius visus sluoksnius ir sumažinus mokymosi žingsnį. Iš nuostolių grafiko galima įžvelgti nuostolių didėjimą, kuris liudija apie modelio persimokymą. Dėl to modelio mokymas buvo stabdomas ir naudojama 30-os epochos būsena.

Apmokymo metu, prieš modeliui priimant rekurentinių diagramų paveiksliukus, paveikslėlių dydis buvo pakeičiamas iš 780x780 į 224x224. Tokia transformacija yra reikalinga nes pirmasis

ResNet buvo apmokytas atpažinti tokio dydžio paveiksliukus ir be jos modelio žinių perdavimas bus neefektyvus.

5.2. Rekurentiniai neuroniniai tinklai

Darbo metu buvo apmokomi rekurentinių neuroninių tinklų šeimos modeliai: RNN ir LSTM. RNN modeliai žymimi $RNN^d_{m\times n}$, kur m – yra paslėptų sluoksnių skaičius, n – neuronų esančių paslėptame sluoksnyje skaičius ir d – žymi kurį kintamumą modelis yra apmokytas prognozuoti (busiantį po 1, 2, 5, 10 ir 21 dienos). LSTM modeliai žymimi analogiškai $LSTM^d_{m\times n}$. Taip pat, buvo bandoma ištirti RQA metrikų įtaką šių modelių tikslumui, todėl RNN ir LSTM modelių įvestims dar buvo pateikiamos RQA metrikos. Tokie modeliai bus žymimi $RNN^d_{m\times n+rqa}$ ir $LSTM^d_{m\times n+rqa}$.

Modelių apmokymui panaudotas toks pat duomenų padalinimas, kaip ir apmokant konvoliucinius neuroninius tinklus. Kiekvienas iš modelių buvo mokomas mažiausiai 100 epochų. Įžvelgus, kad modelis po 100 epochų dar nebuvo apsimokęs, jo mokymas buvo tęsiamas.

5.3. Garch modeliai

Siekiant palyginti CNN ir RNN modelius su egzistuojančiais matematiniais modeliais, buvo apmokytas ir GARCH(1,1) modelis. Šio modelio apmokymas yra kitoks nei RNN ir CNN modelių, todėl apmokant šį modelį nebus naudojamas toks pat duomenų padalinimas. GARCH modeliams apmokyti buvo perduoda kiekvienos akcijos visų logaritminių grąžų istorija atskirai. Apmokymui buvo skirta 90% visų duomenų, o testuojama buvo su 10% naujausių duomenų. Taigi, nors ir apmokymo procesas buvo kitoks nei CNN ir RNN šeimos modelių, tačiau modelių tikslumas buvo lyginamas su tais pačiais duomenimis.

Buvo apmokyti 5 modeliai, prognozuojantys kintamumą po 1, 2, 5, 10 ir 21 dienos. Šie modeliai rezultatuose yra žymimi $GARCH^d_{1\_1}$, kur d – žymi kurį kintamumą modelis yra apmokytas prognozuoti.

GARCH modeliai prognozuoja kintamumo reikšmę, todėl prognozuojamos reikšmės yra prilyginamos prognozuojamam kintamumo intervalui (prognozuojami intervalai aprašyti 4.4 skyrelyje).

6. Rezultatai

6.1. Modelių įvertinimas

5 „Modelių apmokymo aspektai“ skyriuje buvo aprašomi GARCH(1, 1), ResNet, RNN ir LSTM tipo modeliai, kurie buvo panaudoti prognozuojant S&P500 [8] uždarymo akcijų kainų kintamumą po 1, 2, 5, 10 ir 21 darbo dienos. Šiame skyriuje yra vertinamas aprašytų algoritmų efektyvumas. Modelių efektyvumui įvertinti buvo panaudotos ACC, PREC, REC ir F1 metrikos bei sudaromos painiavos matricos (žr. skyrelį 2.4.3).

Kintamumo prognozavimo po 1 darbo dienos rezultatai (lentelė nr. 4) parodė, kad geriausiai kitos dienos kintamumą prognozuoja RNN ir LSTM tipo šeimos modeliai. Šių modelių F1 metrikos yra 2 kartus aukštesnės nei ResNet ir GARCH tipo modelių. $RNN^1_{2\times10}$ ir $LSTM^1_{2\times10}$ modelių F1 metrikos siekė 80%, kol ResNet tipo modeliams pavyko pasiekti tik 37%. Visi RNN tipo modeliai nepriklausomai nuo vidinio sluoksnio neuronų skaičiaus ir naudojamų arba ne RQA metrikų pasiekė labai panašų rezultatą (apytiksliai 79%). Tai parodo, kad RNN paslėpto sluoksnio neuronų kiekio padidinimas nepadidins RNN modelio efektyvumo ir, kad RQA metrikos nedaro įtakos kintamumo po 1 darbo dienos prognozavimui šiais modeliais. Įvertinus LSTM šeimos modelių F1 rezultatus, galima pastebėti, kad modeliai turintys didesnį neuronų kiekį paslėptame sluoksnyje prognozuoja kintamumą prasčiau. Iš lentelė nr. 4 rezultatų galima padaryti, kad kitos dienos kintamumo prognozavimui verčiau rinktis RNN arba LSTM tipo modelius vietoj ResNet arba GARCH.

4 lentelė. Kintamumo prognozavimo po 1 dienos įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^1_{1\_1}$	0.41	0.43	0.27	0.26
$ResNet^1_{urp+780d}$	0.52	0.38	0.42	0.37
$ResNet^1_{rp10+rqa+780d}$	0.52	0.37	0.44	0.37
$RNN^1_{2\times10}$	0.87	0.78	0.86	**0.80**
$RNN^1_{2\times10+rqa}$	0.87	0.77	0.85	0.79
$RNN^1_{2\times50}$	0.86	0.77	0.86	0.79
$RNN^1_{2\times50+rqa}$	0.87	0.77	0.86	0.79
$LSTM^1_{2\times10}$	0.87	0.78	0.85	**0.80**
$LSTM^1_{2\times10+rqa}$	0.87	0.77	0.85	0.79
$LSTM^1_{2\times50}$	0.83	0.71	0.78	0.72
$LSTM^1_{2\times50+rqa}$	0.84	0.73	0.79	0.74

Kintamumo prognozavimo po 2 darbo dienų rezultatai (lentelė nr. 5) primena kintamumo prognozavimas po 1 darbo dienos rezultatus. Remiantis F1 metrika RNN ir LSTM šeimos modeliai prognozuoja kintamumą geriau nei ResNet ar GARCH(1, 1) modeliai. Vienintelis skirtumas yra tai, kad šių modelių prognozavimo kokybė yra prastesnė. $RNN_{2\times10}$ (geriausiai prognozuojančio kintamumą po 1 darbo dienos) modelio F1 metrikos reikšmė sumažėjo 5%, visų kitų tiriamų modelių F1 metrikų reikšmė taip pat sumažėjo. Toks F1 metrikos sumažėjimas parodo, kad kintamumo prognozavimas po 2 darbo dienų yra sudėtingesnis uždavinys nei po 1 darbo dienos.

5 lentelė. Kintamumo prognozavimo po 2 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{2}_{1\_1}$	0.42	0.43	0.27	0.25
$ResNet^{2}_{urp+780d}$	0.50	0.35	0.40	0.35
$ResNet^{2}_{rp10+rqa+780d}$	0.49	0.35	0.44	0.35
$RNN^{2}_{2\times10}$	0.82	0.73	0.81	**0.75**
$RNN^{2}_{2\times10+rqa}$	0.82	0.72	0.80	0.73
$RNN^{2}_{2\times50}$	0.81	0.71	0.79	0.72
$RNN^{2}_{2\times50+rqa}$	0.82	0.71	0.79	0.73
$LSTM^{2}_{2\times10}$	0.81	0.71	0.79	0.72
$LSTM^{2}_{2\times10+rqa}$	0.80	0.71	0.80	0.72
$LSTM^{2}_{2\times50}$	0.80	0.66	0.72	0.67
$LSTM^{2}_{2\times50+rqa}$	0.79	0.68	0.74	0.68

Kintamumo prognozavimo po 5 darbo dienų (lentelė nr. 6) ir 10 darbo dienų rezultatai (lentelė nr. 7) primena prieš tai aptartus rezultatus. Kintamumą vis dar prognozuoja tiksliausiai RNN ir LSTM tipo modeliai, taip pat, šių modelių kokybė prastėja prognozuojant kintamumą vėlesniu laiku. Įdomus pastebėjimas, kad GARCH(1, 1) tipo modelių F1 metrikos reikšmė neprastėja. Visgi, autoriaus nuomone, tai nėra reikšmingas atradimas, nes 25% F1 metrikos reikšmė yra pati mažiausia iš tiriamų modelių.

6 lentelė. Kintamumo prognozavimo po 5 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{5}_{1\_1}$	0.41	0.42	0.26	0.25
$ResNet^{5}_{urp+780d}$	0.44	0.30	0.36	0.30
$ResNet^{5}_{rp10+rqa+780d}$	0.42	0.29	0.36	0.28
$RNN^{5}_{2\times10}$	0.65	0.55	0.64	**0.57**
$RNN^{5}_{2\times10+rqa}$	0.66	0.57	0.65	**0.58**
$RNN^{5}_{2\times50}$	0.67	0.59	0.62	**0.57**
$RNN^{5}_{2\times50+rqa}$	0.66	0.57	0.64	0.56
$LSTM^{5}_{2\times10}$	0.64	0.56	0.63	0.55
$LSTM^{5}_{2\times10+rqa}$	0.63	0.58	0.62	0.56
$LSTM^{5}_{2\times50}$	0.65	0.52	0.60	0.52
$LSTM^{5}_{2\times50+rqa}$	0.64	0.49	0.56	0.49

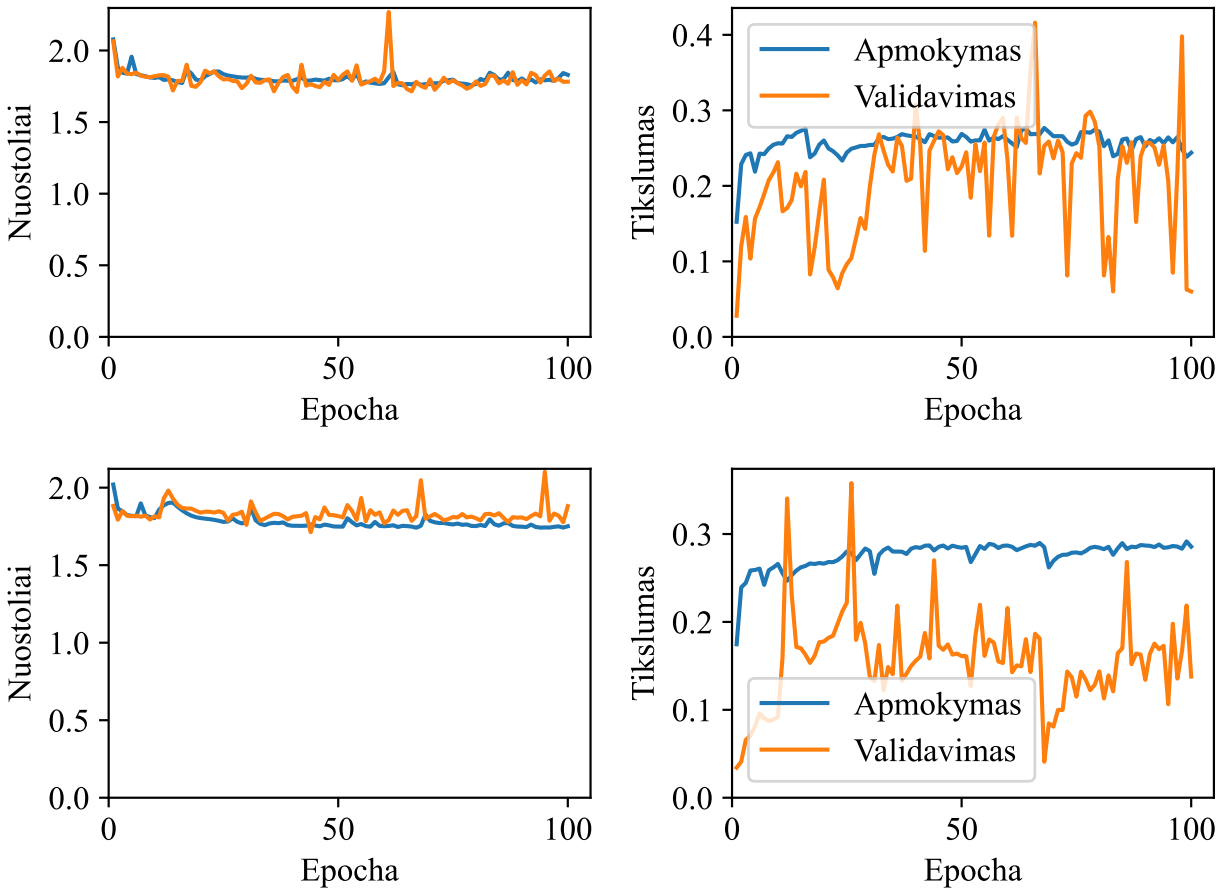
7 lentelė. Kintamumo prognozavimo po 10 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{10}_{1\_1}$	0.40	0.40	0.25	0.24
$ResNet^{10}_{urp+780d}$	0.40	0.26	0.31	0.25
$ResNet^{10}_{rp10+rqa+780d}$	0.37	0.25	0.28	0.23
$RNN^{10}_{2\times10}$	0.46	0.38	0.48	0.38
$RNN^{10}_{2\times10+rqa}$	0.51	0.44	0.50	**0.43**
$RNN^{10}_{2\times50}$	0.54	0.43	0.50	**0.43**
$RNN^{10}_{2\times50+rqa}$	0.53	0.38	0.49	0.38
$LSTM^{10}_{2\times10}$	0.46	0.44	0.47	0.40
$LSTM^{10}_{2\times10+rqa}$	0.49	0.49	0.48	**0.43**
$LSTM^{10}_{2\times50}$	0.51	0.34	0.43	0.34
$LSTM^{10}_{2\times50+rqa}$	0.52	0.35	0.44	0.35

Kintamumo prognozavimo po 21 darbo dienos (lentelė nr. 8) rezultatai parodė pačius prasčiausius rezultatus. Skirtingai nei praeituose rezultatuose, šį kartą geriausius rezultatus pasiekė ResNet ir GARCH(1, 1) tipo modeliai. Įdomu tai, kad lyginant $RNN^{21}_{2\times10}$ su $RNN^{21}_{2\times10+rqa}$ ir $RNN^{21}_{2\times50}$ su $RNN^{21}_{2\times50+rqa}$ modelius, galima pastebėti, kad RQA metrikos padeda pasiekti geresnius rezultatus, tačiau atkreipus dėmesį į modelių nuostolių ir tikslumo grafikus (30 pav) galima suprasti, kad modeliai neišmoko prognozuoti kintamumo ir todėl neverta kreipti dėmesio. Visų modelių, kurie neišmokusių prognozuoti kintamumo, F1 rezultatai pažymėti raudona spalva.

8 lentelė. Kintamumo prognozavimo po 21 dienos įvertinimo metrikos.

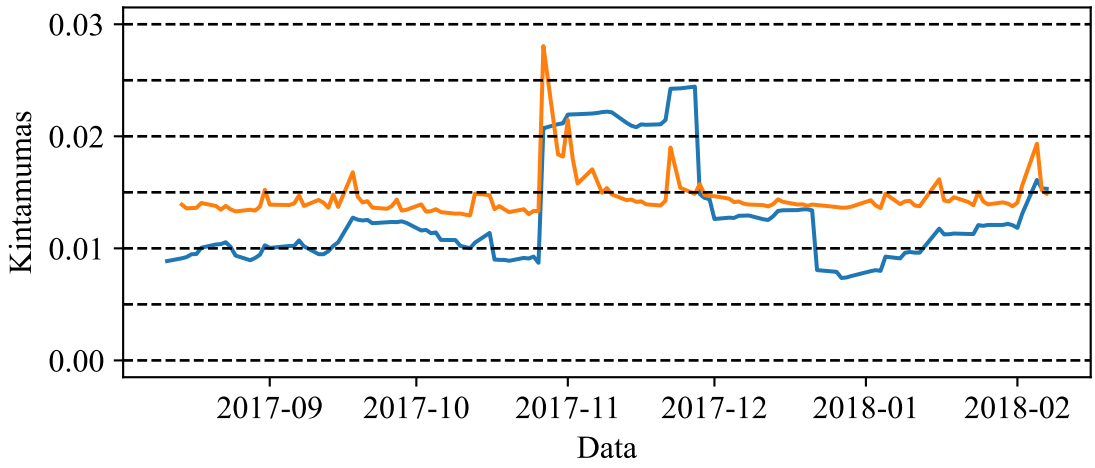
Modelis	ACC	PREC	REC	F1
$Garch^{21}_{1\_1}$	0.35	0.34	0.21	**0.19**
$ResNet^{21}_{urp+780d}$	0.38	0.22	0.23	**0.20**
$ResNet^{21}_{rp10+rqa+780d}$	0.34	0.21	0.24	0.18
$RNN^{21}_{2\times10}$	0.04	0.15	0.24	0.04
$RNN^{21}_{2\times10+rqa}$	0.13	0.17	0.24	0.12
$RNN^{21}_{2\times50}$	0.14	0.15	0.25	0.09
$RNN^{21}_{2\times50+rqa}$	0.22	0.24	0.28	0.18
$LSTM^{21}_{2\times10}$	0.21	0.20	0.26	0.16
$LSTM^{21}_{2\times10+rqa}$	0.18	0.20	0.26	0.14
$LSTM^{21}_{2\times50}$	0.26	0.20	0.26	0.17
$LSTM^{21}_{2\times50+rqa}$	0.21	0.19	0.25	0.15



30 pav. $RNN^{21}_{2\times10}$ (iliustracija viršuje) ir $RNN^{21}_{2\times10+rqa}$ (iliustracija apačioje) nuostolių ir tikslumo grafikai. Iš to, kad modelių tikslumas smarkiai diverguoja, nors nuostoliai konverguoja, galima daryti išvadą, kad modeliai neišmoko prognozuoti kintamumo.

6.2. Modelių įvertinimas su paklaida

6.1 skyrelyje pateikti rezultatai vertina, kaip gerai modeliams sekasi identifikuoti intervalą (žr. 4.4 skyrelį), į kurį pakliūna kintamumas. Ne visada yra norima atspėti kintamumo rėžį. Kartais yra naudingiau atspėti kintamumo reikšmę su paklaidą. Pavyzdžiui, kai kintamumo reikšmė yra lygi 0.51%, o modelis prognozuoja, kad kintamumas paklius į intervalą $[0\%, 0.5\%)$ modelis bus laikomas neteisingu, nors iš tikrųjų jis buvo labai arti tiesos. Ši problema, taip pat, yra įžvelgiama paveikslėlyje nr. 31.



31 pav. CERN kompanijos akcijų kintamumo prognozavimas naudojant GARCH(1, 1) modelį. Oranžine spalva pažymėta GARCH(1, 1) prognozuojama reikšmė, mėlyna – apskaičiuotas 21 dienos kintamumas. Iš kreivių matoma, kad prognozuojamas kintamumas, kartais būna už linijos ir ne visada pakliūna į tą patį intervalą su apskaičiuotu kintamumu. Dėl to modelio tikslumas įvertinamas prasčiau.

Siekiant geriau suprasti modelių efektyvumą, modelių rezultatai buvo dar kartą įvertinti, tik šį kart su galima paklaida lygia **0.5%** (reikšmė tarp gretimų kategorijų). F1 metrikos buvo atvaizduojamos lentelėje nr. 9. Šioje lentelėje RNN modelių $F1^{21}$ metrikos yra pažymėtos brūkšniu, nes šių modelių nepavyko apmokyti (30 pav.). Išsamios metrikos pateiktos Priede A lentelėse nr. 10, 11, 12, 13 ir 14.

9 lentelė. Kintamumo prognozavimo su paklaida F1 vertinimo metrikos.

Modelis	$F1^1$	$F1^2$	$F1^5$	$F1^{10}$	$F1^{21}$
$Garch_{1\_1}$	0.72	0.72	0.72	0.69	0.58
$ResNet_{urp+780d}$	0.82	0.78	0.72	0.62	0.57
$ResNet_{rp10+rqa+780d}$	0.83	0.80	0.72	0.62	0.54
RNN_{2x10}	0.98	0.96	0.88	0.72	–
$RNN_{2x10+rqa}$	0.98	0.96	0.88	0.75	–
RNN_{2x50}	0.98	0.96	0.90	0.75	–
$RNN_{2x50+rqa}$	0.98	0.96	0.87	0.70	–
$LSTM_{2x10}$	0.98	0.96	0.88	0.74	0.48
$LSTM_{2x10+rqa}$	0.98	0.96	0.88	0.76	0.49
$LSTM_{2x50}$	0.97	0.96	0.84	0.66	0.50
$LSTM_{2x50+rqa}$	0.98	0.96	0.82	0.69	0.46

Iš lentelėje nr. 9 matomų duomenų galima įžvelgti kintamumą būsiantį anksčiau nei po 10 dienų, tiksliausiai prognozuoja RNN modeliai, tačiau tolimesnį kintamumą geriausiai prognozuoja GARCH(1, 1) modelis.

Išvados ir rekomendacijos

Šiame darbe buvo bandoma prognozuoti į kokį intervalą pakliūtų S&P500 [8] akcijų kainų kintamumas būsiantis po 1, 2, 5, 10 ir 21 darbo dienos. Kintamumo prognozavimui buvo naudojami istoriniai duomenys: 5 metų akcijų uždarymo kainų istorija renkama kas dieną, dienos pabaigoje. Iš istorinių duomenų buvo sudarytos klasikinės ir spalvinės rekurentinės diagramos, išryškinant apytiksliai 10% juodų pikselių. Be to, klasikinėms rekurentinėms diagramoms papildomai buvo apskaičiuotos RQA metrikos. Rekurentinės diagramos kartu su RQA metrikomis buvo panaudotos apmokant konvoliucinius ir rekurentinius neuroninius tinklus. Šių modelių rezultatai buvo lyginami su GARCH(1, 1) modelio rezultatais. Kadangi GARCH(1, 1) modeliai prognozuoja skaitinę kintamumo reikšmę, ši reikšmė buvo prilyginta tiems patiems intervalams, kuriuos bando nuspėti konvoliuciniai ir rekurentiniai modeliai.

Įvertinus modelių kokybę buvo padarytos pagrindinės išvados:

1. Visi darbe nagrinėti modeliai prognozuoja kintamumo intervalą, būsiantį po 1, 2, 5 ir 10 dienų, tiksliau nei GARCH(1, 1).

2. Kintamumo intervalą po 21 dienos tiksliausiai prognozuoja ResNet modelis, apmokytas spalvinėmis rekurentinėmis diagramomis ir 780 dienų kintamumo istorija.

3. Kintamumo intervalą, būsiantį po 1, 2, 5 ir 10 dienų tiksliausiai prognozuoja rekurentiniai neuroniniai tinklai.

4. RQA įvestis rekurentiniams neuroniniams tinklams nedaro įtakos.

5. Kintamumo prognozavimas artimesnei dienai yra tikslesnis nei tolimesnei.

Be to, siekiant pagrindinių tikslų, buvo rasti šalutiniai pastebėjimai:

1. Konvoliuciniai neuroniniai tinklai apmokyti tik rekurentinėmis diagramomis, prognozuoja kintamumą prasčiau nei tokie patys modeliai apmokyti priimti ir rekurentinėms diagramoms sudaryti reikalingas laiko eilutes (žr. 2).

2. Norint rekurentinėms diagramoms pritaikyti duomenų augmentaciją, tai galima padaryti iškerpant jos dalį palei diagramos centrinę įstrižainę. Taip nėra sugadinamos diagramos savybės (žr. 5.1.4).

Taip pat buvo suformuluotos šios rekomendacijos:

1. Siekiant prognozuoti kintamumą būsiantį anksčiau nei po 10 dienų, verta naudoti rekurentinius neuroninius tinklus, nes šie modeliai duoda tiksliausius rezultatus.

2. Kintamumo prognozavimui po 21 dienos verta naudoti ResNet modelį su spalvinėmis rekurentinėmis diagramomis ir laiko eilučių istorija, nes šie modeliai duoda tiksliausius rezultatus.

3. Apmokant modelius rekurentinėmis diagramomis visada verta į modelio įvestį pridėti ir diagramoms sudaryti panaudotas laiko eilutes, nes tokie modeliai duoda tiksliausius rezultatus.

Ateities tyrimų gairės

- Darbe buvo pasirinkta sudaryti rekurentines diagramas visada išryškinant 10% visų juodų pikselių. Ateityje verta ištirti, kaip šio procento pasirinkimas lemia kintamumo prognozavimo tikslumą.
- Šiame darbe, apmokant konvoliucinius neuroninius tinklus, buvo naudojama tik viena architektūra – ResNet. Verta pabandyti prognozuoti kintamumą ir su kitomis architektūromis.
- Darbo metu kryžminė validacija buvo panaudota tik su dirbtiniais duomenimis. Verta ją panaudoti ir su finansiniais duomenimis.
- Darbo metu buvo skaičiuojamas 21 dienos kintamumas (žr. 2.1 ir 4.2 skyrelius). Verta ištirti ir kitokio dydžio kintamumą.
- ResNet modeliams apmokyti buvo naudojami 44415 rekurentinių diagramų paveikslėlių. Verta pabandyti naudoti daugiau duomenų.

Literatūros šaltiniai

[1] arch. https://github.com/bashtage/arch. Tikrinta: 2024-01-07.

[2] Bash. https://www.gnu.org/software/bash/. Tikrinta: 2024-01-07.

[3] debian. https://www.debian.org/. Tikrinta: 2024-01-07.

[4] Imagenet. https://image-net.org/index.php. Tikrinta: 2024-01-07.

[5] joblib. https://joblib.readthedocs.io/en/stable/. Tikrinta: 2024-01-07.

[6] Mif vu pst. https://mif.vu.lt/itwiki/hpc. Tikrinta: 2024-01-07.

[7] Python. https://www.python.org/. Tikrinta: 2024-01-07.

[8] S&P 500 stock data.
https://www.kaggle.com/datasets/camnugent/sandp500.
Tikrinta: 2023-10-01.

[9] vast.ai global gpu market. https://vast.ai/. Tikrinta: 2024-01-07.

[10] Dima Alberg, Haim Shalit, and Rami Yosef. Estimating stock market volatility using asymmetric garch models. *Applied Financial Economics*, 18(15):1201–1208, 2008.

[11] Tadas Meškauskas Algimantas Juozapavičius. *Vaizdų ir signalų analizė ir apdorojimas*. TEV, 2011.

[12] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8, 12 2021.

[13] Luana Barros, Gabriel Soares, Suzete Correia, Gabriel Duarte, and Silvana Costa. Classification of recurrence plots of voice signals using convolutional neural networks. Sociedade Brasileira de Telecomunicações, 2020.

[14] Choo Wei Chong, Muhammad Idrees Ahmad, and Mat Yuso Abdullah. Performance of garch models in forecasting stock market volatility. *Journal of forecasting*, 18(5):333–343, 1999.

[15] Hung-hsiang Chou. Chinese oracle bones. *Scientific American*, 240(4):134–149, 1979.

[16] Christian L. Dunis and Xuehuan Huang. Forecasting and trading currency volatility: An application of recurrent neural regression and model combination. *Journal of Forecasting*, 21:317–354, 2002.

[17] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32:2627–2636, 8 1998.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. 8 2020.

[20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[21] Nima Hatami, Yann Gavet, and Johan Debayle. Classification of time-series images using deep convolutional neural networks. 10 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[24] Xixi Li, Yanfei Kang, and Feng Li. Forecasting with time series imaging. *Expert Systems with Applications*, 160:113680, 12 2020.

[25] Norbert Marwan, M. Carmen Romano, Marco Thiel, and Jürgen Kurths. Recurrence plots for the analysis of complex systems, 1 2007.

[26] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Otávio A B Penatti and Milton F S Santos. Pattern recognition letters human activity recognition from mobile inertial sensors using recurrence plots, 2017.

[30] Linas Petkevičius. Di sąvokų žodynėlis.
http://klevas.mif.vu.lt/~linp/page/savokos.html.
Tikrinta: 2024-01-07.

[31] Tobias Rawald, Mike Sips, and Norbert Marwan. Pyrqa—conducting recurrence quantification analysis on very long time series e ciently. *Computers & Geosciences*, 104:101–108, 2017.

[32] Peter J. Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:73–79, 1 2011.

[33] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning?, 2016.

[34] Andrew L. Turner and Eric J. Weigel. Daily stock market volatility: 1928–1989. *Management Science*, 38:1586–1609, 11 1992.

[35] Efthymios Tzinis, Georgios Paraskevopoulos, Christos Baziotis, and Alexandros Potamianos. Integrating recurrence dynamics for speech emotion recognition. 11 2018.

[36] Jr. Charles L. Webber and Norbert Marwan. *Understanding Complex Systems Recurrence Quantii cation Analysis Theory and Best Practices*. 2015.

[37] Marc Wenninger, Sebastian P. Bayerl, Jochen Schmidt, and Korbinian Riedhammer. Timage – a robust time series classification pipeline. 9 2019.

[38] Sanjay Yadav and Sanyam Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. pages 78–83. Institute of Electrical and Electronics Engineers Inc., 8 2016.

[39] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[40] Chun Xia Zhang, Jun Li, Xing Fang Huang, Jiang She Zhang, and Hua Chuan Huang. Forecasting stock volatility and value-at-risk based on temporal convolutional networks. *Expert Systems with Applications*, 207, 11 2022.

Priedai

A. Kintamumo prognozavimo su leidžiama paklaida vertinimo metrikos

Visos žemiau pateiktos lentelės sudarytos leidžiant modeliui padaryti 5% paklaidą (žr. 6.2 skyrelį).

10 lentelė. Kintamumo prognozavimo po 1 dienos įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{1}_{1\_1}$	0.89	0.76	0.70	0.72
$ResNet^{1}_{urp+780d}$	0.93	0.82	0.83	0.82
$ResNet^{1}_{rp10+rqa+780d}$	0.93	0.83	0.84	0.83
RNN^{1}_{2x10}	0.99	0.97	0.98	0.98
$RNN^{1}_{2x10+rqa}$	0.99	0.97	0.98	0.98
RNN^{1}_{2x50}	0.99	0.97	0.98	0.98
$RNN^{1}_{2x50+rqa}$	0.99	0.97	0.98	0.98
$LSTM^{1}_{2x10}$	0.99	0.97	0.98	0.98
$LSTM^{1}_{2x10+rqa}$	0.99	0.97	0.98	0.98
$LSTM^{1}_{2x50}$	0.99	0.97	0.98	0.97
$LSTM^{1}_{2x50+rqa}$	0.99	0.98	0.98	0.98

11 lentelė. Kintamumo prognozavimo po 2 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{2}_{1\_1}$	0.89	0.76	0.71	0.72
$ResNet^{2}_{urp+780d}$	0.92	0.78	0.80	0.78
$ResNet^{2}_{rp10+rqa+780d}$	0.92	0.80	0.81	0.80
RNN^{2}_{2x10}	0.99	0.96	0.97	0.96
$RNN^{2}_{2x10+rqa}$	0.99	0.96	0.97	0.96
RNN^{2}_{2x50}	0.99	0.96	0.97	0.96
$RNN^{2}_{2x50+rqa}$	0.99	0.96	0.97	0.96
$LSTM^{2}_{2x10}$	0.99	0.96	0.97	0.96
$LSTM^{2}_{2x10+rqa}$	0.99	0.96	0.97	0.96
$LSTM^{2}_{2x50}$	0.99	0.95	0.96	0.96
$LSTM^{2}_{2x50+rqa}$	0.99	0.95	0.96	0.96

12 lentelė. Kintamumo prognozavimo po 5 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{5}_{1\_1}$	0.89	0.76	0.70	0.72
$ResNet^{5}_{urp+780d}$	0.89	0.73	0.75	0.72
$ResNet^{5}_{rp10+rqa+780d}$	0.89	0.72	0.74	0.72
RNN^{5}_{2x10}	0.95	0.85	0.91	0.88
$RNN^{5}_{2x10+rqa}$	0.96	0.86	0.91	0.88
RNN^{5}_{2x50}	0.96	0.90	0.91	0.90
$RNN^{5}_{2x50+rqa}$	0.96	0.85	0.92	0.87
$LSTM^{5}_{2x10}$	0.96	0.87	0.91	0.88
$LSTM^{5}_{2x10+rqa}$	0.96	0.88	0.91	0.88
$LSTM^{5}_{2x50}$	0.95	0.82	0.89	0.84
$LSTM^{5}_{2x50+rqa}$	0.95	0.79	0.89	0.82

13 lentelė. Kintamumo prognozavimo po 10 dienų įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{10}_{1\_1}$	0.88	0.74	0.68	0.69
$ResNet^{10}_{urp+780d}$	0.86	0.65	0.65	0.62
$ResNet^{10}_{rp10+rqa+780d}$	0.85	0.64	0.65	0.62
RNN^{10}_{2x10}	0.89	0.70	0.81	0.72
$RNN^{10}_{2x10+rqa}$	0.92	0.75	0.81	0.75
RNN^{10}_{2x50}	0.92	0.75	0.81	0.75
$RNN^{10}_{2x50+rqa}$	0.90	0.68	0.80	0.70
$LSTM^{10}_{2x10}$	0.90	0.77	0.80	0.74
$LSTM^{10}_{2x10+rqa}$	0.91	0.82	0.80	0.76
$LSTM^{10}_{2x50}$	0.89	0.64	0.76	0.66
$LSTM^{10}_{2x50+rqa}$	0.89	0.66	0.78	0.69

14 lentelė. Kintamumo prognozavimo po 21 dienos įvertinimo metrikos.

Modelis	ACC	PREC	REC	F1
$Garch^{21}_{1\_1}$	0.83	0.63	0.57	0.58
$ResNet^{21}_{urp+780d}$	0.82	0.59	0.61	0.57
$ResNet^{21}_{rp10+rqa+780d}$	0.81	0.55	0.59	0.54
RNN^{21}_{2x10}	0.44	0.62	0.55	–
$RNN^{21}_{2x10+rqa}$	0.60	0.49	0.59	–
RNN^{21}_{2x50}	0.54	0.57	0.52	–
$RNN^{21}_{2x50+rqa}$	0.74	0.59	0.59	–
$LSTM^{21}_{2x10}$	0.71	0.65	0.56	0.48
$LSTM^{21}_{2x10+rqa}$	0.68	0.57	0.57	0.49
$LSTM^{21}_{2x50}$	0.73	0.52	0.59	0.50
$LSTM^{21}_{2x50+rqa}$	0.69	0.52	0.57	0.46