KLAIPĖDOS UNIVERSITETO
JŪROS TECHNOLOGIJŲ IR GAMTOS MOKSLŲ FAKULTETO
INFORMATIKOS IR STATISTIKOS KATEDRA

PROGRAMINĖS ĮRANGOS, SKIRTOS NESTACIONARIŲ LAIKO EILUČIŲ TYRIMUI, PROJEKTAVIMAS IR PROGRAMAVIMAS

Magistro baigiamasis darbas

Autorius JMNTS18 gr. studentas Audrius Domarkas

Vadovas doc. dr. Dalia Baziukė

Klaipėda, 2020

Pildo bakalauro / magistro baigiamojo darbo autorius

.....
(bakalauro / magistro baigiamojo darbo autoriaus vardas, pavardė)

.....
(bakalauro / magistro baigiamojo darbo pavadinimas lietuvių kalba)

Patvirtinu, kad bakalauro / magistro baigiamasis darbas parašytas savarankiškai, nepažeidžiant kitiems asmenims priklausančių autorių teisių, visas baigiamasis bakalauro / magistro darbas ar jo dalis nebuvo panaudotas Klaipėdos universitete ir kitose aukštosiose mokyklose.

.....
(bakalauro / magistro baigiamojo darbo autoriaus parašas)

Sutinku, kad bakalauro / magistro baigiamasis darbas būtų naudojamas neatlygintinai 5 m. Klaipėdos universiteto studijų procese.

.....
(bakalauro / magistro baigiamojo darbo autoriaus parašas)

Pildo bakalauro / magistro baigiamojo darbo vadovas

Bakalauro / magistro baigiamąjį darbą ginti .....
(įrašyti – leidžiu arba neleidžiu)

....
(data) *(bakalauro / magistro baigiamojo darbo vadovo vardas, pavardė ir parašas)*

Pildo katedros / centro, kuruojančios studijų programą, administratorius (sekretorius)

Baigimasis darbas įregistruotas katedroje (centre).....
(data)

.....
(katedros sekretorės vardas, pavardė ir parašas)

Pildo katedros / centro, kuruojančios studijų programą, vedėjas

Bakalauro / magistro baigiamąjį darbą ginti .....
(įrašyti – leidžiu arba neleidžiu)

....
(data) *(bakalauro / magistro baigiamojo darbo vadovo vardas, pavardė ir parašas)*

Recenzentu (-ais) skiriu.....

.....
(įrašyti recenzento (ų) vardą, pavardę)

....
(data) *(katedros / centro vedėjo vardas, pavardė ir parašas)*

Domarkas A. Programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, projektavimas ir programavimas. Techninių informacinių sistemų inžinerijos magistro studijų programos baigiamasis darbas. Darbo vadovė doc. dr. Dalia Baziukė, Klaipėdos universitetas: Klaipėda, 2020. – 69 p.

Raktažodžiai: nestacionarūs laiko eilučių duomenys, neuroninis tinklas, SRNN (Simple Recurrent Neural Network) LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit).

SANTRAUKA

Apdorojant nestacionarius laiko eilučių duomenis statistiniais metodais jie privalo būti stacionarizuojami. Tačiau taip ne tik prarandamos svarbios duomenų ypatyvės, bet ir nuvertinama prognozės neužtikrintumo tikimybė. Rekurentiniais neuroniniais tinklais galima analizuoti nestacionarizuotus duomenis, tačiau nėra aišku, kuris iš trijų dažniausiai naudojamų tinklų – SRNN, LSTM ar GRU, – geriausiai tinka kiekvienu konkrečiu nestacionarių laiko eilučių duomenų atveju. Šiame magistro darbe aprašomas programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui panaudojant trijų tipų rekurentinius neuroninius tinklus, projektavimas ir programavimas, pateikiamas programinis produktas, kuriuo atliktas 50 nestacionarių laiko eilučių duomenų aibių tyrimas bei pristatomi šio tyrimo rezultatai. **Darbo hipotezė:** SRNN, LSTM ir GRU rekurentiniais neuroniniais tinklais galima prognozuoti nestacionarių laiko eilučių duomenis jų nestacionarizavus. **Darbo problema**: kaip atlikti nestacionarių laiko eilučių duomenų tyrimą jų nestacionarizuojant ir panaudojant trijų tipų rekurentinius neuroninius tinklus. **Darbo objektas**: programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, projektavimas ir programavimas. **Darbo tikslas**: suprojektuoti ir suprogramuoti įrangą nestacionarių laiko eilučių prognozės tikslumui vertinti, kai taikomi trijų tipų neuroniniai tinklai. **Darbo uždaviniai**: suformuluoti reikalavimus programinei įrangai, skirtai nestacionarių laiko eilučių tyrimui, ir apibrėžti jos architektūrą; parašyti programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, kodą; atlikti kompiuterinį eksperimentą su atrinktų duomenų aibių duomenimis panaudojant pasirinktas programinės įrangos nuostatas. **Rezultatai:** Taikant numatytuosius neuroninių tinklų hiperparametrus analizuotus finansinius duomenis tiksliausiai prognozuoja LSTM neuroninis tinklas, o prasčiausiai SRNN. Vidutinės sMAPE reikšmės taikant numatytuosius hiperparametrus yra LSTM: 0,53 (53 %), GRU: 0,77 (77 %), SRNN: 1,43 (143 %). Individualiai kiekvienai duomenų aibei paderinus neuroninio rinklo hiperparametrus prognozės tikslumą galima pagerinti 70 % – 90 %. Vidutinės sMAPE reikšmės taikant individualiai nustatytus hiperparametrus yra LSTM: 0,16 (16 %), GRU: 0,06 (6 %), SRNN: 0,28 (28 %). Geriausi visų trijų neuroninių tinklų rezultatai buvo pasiekiami keičiant šiuos hiperparametrus: optimizavimo funkciją „*adagrad*“ naudojant kartu su duomenų sumaišymu (*shuffle* = *true*), epochų skaičiaus

padidinimu iki 100 ar 200 (*Epochs = 200*) bei paketo dydžio sumažinimu iki 10-20 (*Batch size = 20)*. Paketo dydžio pakeitimas į kartotinį skaičiui 5 pagerino analizuotų finansinių duomenų prognozavimą todėl, kad neuroninio tinklo svorių matrica yra atnaujinama pateikus neuroniniam tinklui duomenų imčių skaičių, aprašomą hiperparametru „*batch size*“, o visi analizei naudoti duomenys buvo surinkti darbo dienomis.

Domarkas A. Design and Development of Software for Non-stationary Time Series Analysis. Master's Thesis of Technical Information Systems Engineering study program. Research adviser Assoc. Prof. Dr. Dalia Baziukė, Klaipėda University: Klaipėda, 2020. - 69 p.

Keywords: Non-stationary Time Series Data, Neural Network, SRNN (Simple Recurrent Neural Network) LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit).

SUMMARY

When processing non-stationary time series data by statistical methods, they must be stationarized. However, hereby not only important data features are lost, but also the likelihood of forecast uncertainty is underestimated. Recurrent neural networks can analyze non-stationary data, but it is not clear which of the three most commonly used networks — SRNN, LSTM, or GRU — is best suited for each specific case of non-stationary time series data. This master's thesis describes the design and development of software for nonstationary time series analysis using three types of recurrent neural networks, presents the software and the results of the study of 50 non-stationary time series data sets. **Hypothesis**: SRNN, LSTM and GRU recurrent neural networks predict nonstationary time series data without their stationaryization.**The problem of the work**: how to forecast non-stationary time series data without stationaryization using three types of recurrent neural networks. **Object of the work**: design and development of software for non-stationary time series analysis. **Aim of the work**: to design and to program software for estimating the accuracy of non-stationary time series prediction when using three types of neural networks. **Tasks of the work**: to formulate requirements for software for non-stationary time series analysis and to define its architecture; to write a code for software for non-stationary time series analysis; perform a computer experiment with the data of the selected data sets using the selected software settings. **Results**: Using the default neural network hyperparameters, the financial data analyzed are most accurately predicted by the LSTM neural network and worst by the SRNN. Mean sMAPE values using the default hyperparameters are LSTM: 0.53 (53%), GRU: 0.77 (77%), SRNN: 1.43 (143%). By adjusting the neural set hyperparameters individually for each data set, the prediction accuracy can be improved by 70% - 90%. Mean sMAPE values using individually determined hyperparameters are LSTM: 0.16 (16%), GRU: 0.06 (6%), SRNN: 0.28 (28%). The best results for all three neural networks were obtained by changing the following hyperparameters: using the adagrad optimization function in combination with shuffle = true, increasing the number of epochs to 100 or 200 (Epochs = 200), and reducing the packet size to 10-20 (Batch size = 20). Changing the packet size to a multiple of 5 improved the prediction of the analyzed financial data because the neural network weight matrix is updated by

providing the neural network with the number of data samples described by the batch size hyperparameter and all data used for analysis were collected on weekdays.

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

SANTRUMPOS IR TERMINAI

ADF (angl. *Augmented Dickey-Fuller test*) – *papildytas Dickey-Fuller testas,* vienetinės šaknies testas laiko eilučių stacionarumui nustatyti [1].

ARMA (angl. *Autoregressive Moving Average*) – autoregresinis slenkančio vidurkio modelis. Naudojamas apibūdinti silpnai stacionarias stochastines laiko eilutes dviejų polinomų – autoregresijos ir slenkančiojo vidurkio, – atžvilgiu [2].

ARIMA (angl. *Autoregressive Integrated Moving Average*) – autoregresinis integruotas slenkančio vidurkio modelis. Tai regresinės analizės forma, kuria įvertinama vieno kintamojo įtaka kitiems kintamiesiems. Modelis naudojamas prognozavimui, panaudojant laiko eilučių verčių skirtumus, o ne faktines vertes [3].

GRU (angl. *Gated Recurrent Unit*) – valdomas rekurentinis vienetas. Tai rekurentinio neuroninio tinklo porūšis, neturintis neurono būsenos išsaugojimo atminties. Jis gali sureguliuoti įvesties svorius, todėl išvengiama sprogstančio ar dingstančio gradiento problemos [4].

Laiko eilučių duomenys (angl. *Time Series Data*) – 1) duomenys, surinkti nuosekliai, nustatytais laiko intervalais ir indeksuoti laiko žymėmis [5]; 2) reikšmių, kurias kintamasis įgyja skirtingu laiku, rinkinys [6]; 3) skaitinių duomenų seka, kuria aprašoma konkretaus proceso evoliucija [7]; 4) konkrečios veiklos generuojami skaičiai, kurie fiksuojami reguliariais laiko intervalais ir po to analizuojami. Laiko eilučių duomenys dažniausiai naudojami tokių veiklų, kaip pvz.: pardavimų, užsakymų, pajamų ir kt., tyrimams [8].

LSTM (angl. *Long Short Term Memory*) – ilga trumpalaikė atmintis. Tai rekurentinio neuroninio tinklo porūšis, skirtas modeliuoti laiko eilučių sekas ir jų tolimas priklausomybes. Kiekvienas LSTM neuronas turi tris loginius elementus: įėjimo, išėjimo bei užmiršimo. Šių loginių elementų paskirtis – išsaugoti informaciją, sustabdant ir atstatant jos srautą. Tokia konstrukcija šiam neuroniniam tinklui leidžia efektyviai mokytis tada, kai tinklui paduodami įvykiai atskirti ilgais laiko tarpais [9].

SRNN (angl. *Simple Recurrent Neural Network*) – paprastas rekurentinis neuroninis tinklas. Tai neuroninis tinklas, skirtas nuoseklių duomenų sekų x(1)...x(n) apdorojimui, kai rezultatas priklauso nuo gaunamų duomenų eilės tvarkos. Šis neuroninis tinklas turi didelį trūkumą – išnykstantį arba sprogstantį gradientą [10]. SRNN tinklai naudojami laiko eilučių duomenų, garso, paveiksliukų ir vaizdo įrašų apdorojimui [11].

Stacionarumas (angl. *Stationarity*) – tai laiko eilutės duomenų charakteristika, kai jų vidurkis, dispersija ir autokoreliacija laiko atžvilgiu nekinta [12].

TURINYS

ĮVADAS

Temos aktualumas. Laiko eilučių duomenys yra neatsiejama šiuolaikinio gyvenimo dalis – tai beveik visi finansų, prekybos, orų stebėjimo, sveikatos apsaugos, demografiniai ir kitų mokslo, pramonės bei ūkio šakų renkami ir analizuojami duomenys. Tokių duomenų analizė ir prognozavimas yra svarbūs uždaviniai: analizė padeda suprasti istorinius duomenis, o prognozavimas – modeliuoti ateities scenarijus. Kaip teigia George E. P. Box et al. (2016), laiko eilučių prognozavimas yra ekonomikos, verslo, gamybos, pramoninės veiklos planavimo, kontrolės, valdymo ir optimizavimo procesų pagrindas [13].

Prognozuojant laiko eilučių duomenis būtina atsižvelgti į jų vidines struktūras, tokias kaip stacionarumas, autokoreliacija, tendencija ar sezoniniai pokyčiai [14]. Absoliuti dauguma realaus gyvenimo reiškinių ir procesų generuoja nestacionarius duomenis, kaip teigia David F. Hendry et al. (2016) „laiko eilučių duomenų stacionarumas yra išimtis, o nestacionarumas – norma“ [15, p. 10]. Mokslinėje literatūroje pateikiama nemažai metodų, skirtų nestacionarių duomenų apdorojimui juos stacionarizuojant [16] [17] [14] [18] [15], tačiau prognozuojant taip apdorotus duomenis susiduriama su problema: nors prognozės neužtikrintumo tikimybė išauga, ji yra nuvertinama [19]. Tai reiškia, kad visi planai bei strategijos, kurie remiasi prognoze, taip pat tampa nepatikimi. Pavyzdžiui, Philip R. Christensen et al. (2018) pateikia pasaulinio ir regioninio ekonominio augimo vienam gyventojui iki 2100 m. prognozių įvertinimo tyrimo rezultatus: egzistuoja virš 35% tikimybė, jog klimato kaitos įtaka ekonomikai bus didesnė, nei numatyta griežčiausiame iš galimų scenarijų [20].

Rozaida Ghazali et al. (2009), Changqing Cheng et al. (2015), David F. Hendry et al. (2016) ir kiti tyrėjai taip pat pažymi, kad duomenų nestacionarumas turi ir pranašumų, kurie prarandami stacionarizavus duomenis: pavyzdžiui, nestacionariuose duomenyse lengviau įžvelgti ilgalaikius, patvarius bei priežastinius ryšius tarp kintamųjų, išryškinti klaidingas koreliacijas, panaudoti testus, vertinančius duomenų prognoze paremtos strategijos teisingumą dar prieš pradedant ją vykdyti [21] [22] [15]. Taigi, nestacionarių laiko eilučių duomenų prognozė atsižvelgiant į jų nestacionarumą yra aktuali užduotis, kuri sprendžiama tobulinant duomenų analizės įrankius.

Laiko eilučių analizės ir prognozavimo uždaviniams spręsti tradiciškai pasitelkiami įvairūs statistiniai metodai ir modeliai: slenkančio vidurkio algoritmų modeliai (ARMA, ARIMA), atsitiktinio klaidžiojimo ir tendencijos (trendo) modeliai, vektorių autoregresijos modeliai, kointegruoti ir priežastiniai modeliai ir kiti [7]. Šie metodai ir modeliai yra gerai ištyrinėti ir plačiai taikomi, bet ir turi tam tikrų apribojimų, pavyzdžiui: negali būti analizuojami duomenys, jei jie yra nepilni ar sugadinti; analizuojamos tik tiesinės priklausomybės, atmetant sudėtingus netiesinius ryšius tarp kintamųjų; analizuojami duomenys su vienu kintamuoju, nors didžioji dalis realių laiko

eilučių duomenų turi kelis kintamuosius; prognozių tikslumas mažėja ilgėjant ateities laiko intervalams, tačiau daugelis realių reiškinių reikalauja būtent ilgalaikių prognozių ir t. t. [23]. Kaip teigia Lijana Stabingienė (2014), laiko eilučių statistinės analizės pagrindu sumodeliuota prognozė niekada neturi teorinio savo teisingumo patvirtinimo, nes net itin tikslios žinios apie laiko eilutės duomenis praeityje negarantuoja, kad jie tokie bus ir ateityje [24]. Todėl šiuo metu didelį tyrėjų susidomėjimą kelia metodai, paremti mašininio mokymo algoritmais, ypač rekurentiniais neuroniniais tinklais [25]. Kaip teigia Jason Brownlee (2018), rekurentiniai neuroniniai tinklai turi galimybę kiekviename laiko žingsnyje grąžinti signalą iš išėjimo sluoksnio į įėjimą, todėl geba atpažinti ir išmokti netiesinius ir laikinus ryšius tarp kintamųjų, analizuoti duomenis su daugeliu kintamųjų [23], todėl naudojami ne tik laiko eilučių duomenų prognozavimui [26], bet ir rašytinio teksto atpažinimui [27], kalbos atpažinimui [11], vaizdų atpažinimui [28] ir kitiems tikslams. Rekurentiniai neuroniniai tinklai gali prognozuoti nestacionarių laiko eilučių duomenis tokius, kokie jie yra – t. y. nestacionarizuotus [23] [26] [22].

Mokslinėse publikacijose, pavyzdžiui Felix A. Gers et al. (1999), Juliana Yim (2002), Peter G. Zhang et al. (2005), John Gamboa (2017) ir daugelio kitų mokslininkų darbuose [29] [30] [31] [32] [33] [34] dažniausiai analizuojami trijų rūšių rekurentiniai neuroniniai tinklai, kaip efektyviausiai pritaikomi laiko eilučių prognozavimui – paprastas rekurentinis neuroninis tinklas (angl. *Simple Recurrent Neural Network, SRNN*), ilga trumpalaikė atmintis (angl. *Long Short-Time Memory, LSTM*) ir valdomas rekurentinis vienetas (angl. *Gated Recurrent Unit, GRU*). Egzistuoja nemažai mokslinių darbų, kuriuose analizuojami atskirų rekurentinių neuroninių tinklų taikymo nestacionarių laiko eilučių duomenų tyrimams ypatumai. Pavyzdžiui, Zhengping Che et al. (2018) teigia, kad GRU tinklai yra efektyviausi analizuojant duomenis su keletu kintamųjų, ypač, jei duomenys nepilni [26], Khaled A. Althelaya et al. (2018) teigia, kad LSTM architektūra rodo aukštus tiek trumpalaikės, tiek ilgalaikės prognozės tikslumo rezultatus [35], Jiexi Liu et al. (2019) teigimu LSTM ir GRU yra apskritai efektyviausi prognozuojant nestacionarius laiko eilučių duomenis [25]. Tačiau pabrėžiama, kad konkretaus rekurentinio neuroninio tinklo prognozės tikslumui lemiamos įtakos turi pačių nestacionarių laiko eilučių duomenų ypatybės [36]. Tam, kad atsakyti į klausimą, koks neuroninis tinklas efektyviausiai prognozuoja konkrečius nestacionarių laiko eilučių duomenis, reikalinga programinė įranga, kurios naudotojas galėtų nustatyti turimų duomenų stacionarumą, pasirinkti neuroninį tinklą ir kiekvienu individualiu atveju įvertinti jo efektyvumą prognozuojant papildomai neapdorotus – nestacionarizuotus – duomenis. Tačiau mokslinės literatūros, nagrinėjančios laiko eilučių tyrimo ir rekurentinių neuroninių tinklų panaudojimo problematiką, analizė rodo, kad tokių programinių instrumentų nėra gausu. Galima paminėti nekomercinius laiko eilučių tyrimo programinius produktus: PAST [37], GRETL [38]. Deja, šios programos neturi galimybės prognozuoti duomenų neuroniniais tinklais. Taip pat egzistuoja komerciniai programiniai

produktai: Alyuda Forecaster [39], Attrasoft PredictorPro [40], NeuroSolutions [41] ir panašūs. Šie programiniai produktai leidžia naudotis prognozavimu neuroniniais tinklais, tačiau yra per daug universalūs, turi nebūtinų funkcijų, nėra atviro kodo. Egzistuoja taip pat programavimo kalbos ir programinės bibliotekos, pavyzdžiui, R arba Python, kurių pagalba galima sukurti reikalingą laiko eilučių duomenų prognozavimo programinę įrangą, naudojančią neuroninius tinklus, tačiau tam reikalingos specifinės žinios ir įgūdžiai. Taigi, nestacionarių laiko eilučių duomenų prognozavimui skirtos programinės įrangos projektavimas ir programavimas yra aktuali problema. Todėl šiame magistro darbe yra koncentruojamasi į tris aukščiau paminėtų rūšių rekurentinius neuroninius tinklus, aprašomas programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, projektavimas ir programavimas, pateikiamas programinis produktas, kuriuo atliktas 50 nestacionarių laiko eilučių duomenų aibių tyrimas bei pristatomi šio tyrimo rezultatai.

Darbo hipotezė: SRNN, LSTM ir GRU rekurentiniais neuroniniais tinklais prognozuojami nestacionarių laiko eilučių duomenys jų nestacionarizavus.

Darbo problema: kaip atlikti nestacionarių laiko eilučių duomenų tyrimą jų nestacionarizuojant ir panaudojant trijų tipų rekurentinius neuroninius tinklus.

Darbo objektas: programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, projektavimas ir programavimas.

Darbo tikslas: suprojektuoti ir suprogramuoti įrangą nestacionarių laiko eilučių prognozės tikslumui vertinti, kai taikomi trijų tipų neuroniniai tinklai.

Darbo uždaviniai:

1. Suformuluoti reikalavimus programinei įrangai, skirtai nestacionarių laiko eilučių tyrimui, ir apibrėžti jos architektūrą.
2. Parašyti programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, kodą.
3. Atlikti kompiuterinį eksperimentą su atrinktų duomenų aibių duomenimis panaudojant pasirinktas programinės įrangos nuostatas.

Darbo metodai. *Teoriniai*: mokslinės literatūros analizė, programinės įrangos projektavimo ir programavimo modelių analizė. *Praktiniai*: programinės įrangos projektavimas ir programavimas naudojant SDLC *Waterfall* modelį; statistinė analizė; kompiuterinis eksperimentas.

Darbo praktinis reikšmingumas. Šiame magistro darbe yra pateikiamas programinis produktas, kurio naudotojas gali savarankiškai atlikti turimų nestacionarių laiko eilučių duomenų tyrimą trimis rekurentiniais neuroniniais tinklais. Programinės įrangos naudotojo sąsajos pagalba naudotojas gali pasirinkti tiriamas duomenų aibes, nustatyti jų stacionarumą, keisti išankstinio duomenų paruošimo nuostatas, pasirinkti tyrime naudojamus neuroninius tinklus bei jų nuostatas. Programinė įranga galėtų būti naudojama studijų procese, ar kaip prototipas kuriant komercinį nestacionarių laiko eilučių duomenų analizės įrankį.

I. PROGRAMINĖS ĮRANGOS, SKIRTOS NESTACIONARIŲ LAIKO EILUČIŲ TYRIMUI, PROJEKTAVIMO IR PROGRAMAVIMO TEORINIAI PAGRINDAI

1.1. Nestacionarių laiko eilučių duomenų prognozavimo problematika

1.1.1. Laiko eilučių duomenų charakteristikos

Laiko eilučių duomenys (angl. *Time Series Data*) – tai duomenys, surinkti nuosekliai, nustatytais laiko intervalais ir indeksuoti laiko žymėmis [5]. Šių duomenų aibė aprašo tam tikro konkretaus proceso evoliuciją [7].

Laiko eilučių duomenys yra apibūdinami statistinėmis charakteristikomis.

Vidurkis yra labiausiai paplitęs centrinės tendencijos matas. Tai yra visų verčių suma, padalyta iš verčių skaičiaus. Vidurkis skaičiuojamas pagal (1) formulę [42].

$$\mu = \frac{1}{N}\sum_{j=1}^{N} x_j, \tag{1}$$

čia $\mu - verčių\ vidurkis$;

N – verčių skaičius;

x_j – vertė, kurios indeksas j.

Dispersija aprašo verčių sklaidą apie vidurkį. Atsitiktinio dydžio X dispersija gali būti skaičiuojama pagal (2) formulę [42]:

$$DX = E(X - EX)^2, \tag{2}$$

čia EX– vertės vidurkis;

Autokoreliacija yra koreliacijos koeficientas, kai koreliuoja dvi to paties kintamojo reikšmės laiku X_i ir X_{i+k} . Autokoreliacija gali būti apskaičiuota pagal (3) formulę [43]:

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}, \tag{3}$$

čia $Y_1, Y_2 Y_N$ – verčių matavimai laiko momentu X_1, X_2, X_N;

$\bar{Y}$ – verčių vidurkis;

N – matavimų skaičius.

Verčių vidurkio, dispersijos ir autokoreliacijos kitimas laiko atžvilgiu lemia laiko eilutės duomenų stacionarumą.

1.1.2. Laiko eilučių duomenų stacionarumas

Laiko eilutės duomenys yra laikomi stacionariais, jei jų vidurkis, dispersija ir autokoreliacija laiko atžvilgiu nekinta [12]. Mokslinėje literatūroje pateikiamos tokios laiko eilučių stacionarumo rūšys [44] [13]:

1. *Griežto stacionarumo* duomenų vidurkis, dispersija ir autokoreliacija laike nesikeičia.
2. *Silpno stacionarumo* duomenų vidurkis pastovus, o dispersija keičiasi.
3. *Tendencijos stacionarumo* laiko eilučių duomenų modelis vieneto šaknies neturi, tačiau turi tendenciją.
4. *Skirtumo stacionarumo* laiko eilučių duomenis galima paversti silpno stacionarumo duomenimis diferencijavimo būdu.
5. *Lokalaus stacionarumo* duomenys turi lėtai kintantį vidurkį, tačiau mažesniuose laiko intervaluose gali būti vertinami kaip stacionarūs.

Duomenų stacionarumui įvertinti gali būti naudojami grafiniai arba skaitiniai metodai.

Paprasčiausias būdas duomenų stacionarumui įvertinti yra *grafinė analizė* – pavaizdavus duomenų eilutės vertes grafike, galima vizualiai įvertinti ar duomenis sugeneravęs procesas buvo stacionarus [45].

Skaitiniai metodai testuoja nulinę hipotezę. Hipotezės testavimas yra tikimybės, kad pasirinkta hipotezė yra teisinga, nustatymas statistiniais metodais. Hipotezės testavimas atliekamas keturiais žingsniais:

- Formuluojama nulio hipotezė H_0 (kad stebėjimai yra atsitiktinumo rezultatas) ir alternatyvi hipotezė H_a (kad stebėjimai atspindi realų poveikį, kartu su atsitiktinumo faktoriumi).
- Randama testo statistika, kurią galima naudoti norint įvertinti nulio hipotezės teisingumą.
- Darant prielaidą, kad nulio hipotezė yra teisinga, apskaičiuojama *P* vertė, kuri yra tos prielaidos tikimybė. Kuo mažesnė *P* vertė, tuo mažesnė tikimybė, kad nulio hipotezė yra teisinga.
- P vertė palyginama su priimtina reikšmingumo verte, kai kada vadinama *alfa* verte. Jei *P* vertė yra mažesnė arba lygi *alfa* vertei, tai stebimas poveikis yra statistiškai reikšmingas, nulio hipotezė yra atmesta, ir galioja alternatyvi hipotezė. [46]

Populiariausi duomenų stacionarumo skaitiniai testai yra šie:

Papildytas Dickey-Fuller testas (angl. *Augmented Dickey-Fuller test, ADF*). Šis testas testuoja nulinę hipotezę, kad pasirinktų duomenų autoregresinis modelis turi vieneto šaknį. Jei nulinė hipotezė pasitvirtina, tai reiškia, jog pasirinkta duomenų aibės įrašas yra nestacionarus. Vienas iš ADF testo

rezultatų yra *p* skaičius. Jei *p* yra mažesnis už 0,05, tai galima teigti, jog duomenys yra stacionarūs, arba, kitaip sakant, neturi nuo laiko priklausančių struktūrų [47].

KPSS (Kwiatkowski-Phillips-Schmidt-Shin) testas taip pat testuoja nulinę hipotezę, kad pasirinktų duomenų autoregresinis modelis vieneto šaknies neturi. Skirtingai nuo ADF testo, KPSS testas patvirtina stacionarumą, net jei jis svyruoja apie vidurkį arba linijinę tendenciją [48].

ADF testas pasižymi tuo, kad struktūrinius lūžius duomenyse traktuoja kaip nestacionarumą. Tai reiškia, kad esant lūžiui, *ADF* testo nulinė hipotezė nebus atmesta. Todėl kaip papildomas testas naudojamas *ZA* (*Zivot-Andrews*) testas, kuris nustato endogeninį (koreliuojantį su duomenimis) struktūrinį lūžį duomenyse bei jo datą. *ZA* testo nulinė hipotezė yra ta, kad duomenys turi vieneto šaknį (t. y. yra nestacionarūs) su struktūriniu lūžiu, o alternatyvi hipotezė – kad duomenys yra stacionarūs, bet su lūžiu. Testo pagrindinis tikslas ir yra atmesti nulio hipotezę bei patvirtinti alternatyvią hipotezę [49]. *ZA* testas palengvina laiko eilučių analizę, nes nustato lūžio datą, todėl galima analizuoti, su kuo tas lūžis susijęs. Pavyzdžiui, finansiniuose duomenyse struktūriniai lūžiai gali atsirasti dėl valstybinės politikos pokyčių, karų, valiutos krizės ir pan. [50].

Dažniausiai praktikoje duomenys testuojami *ADF* testu, o *KPSS* ir *ZA* testai naudojamas kaip papildomi [45]. Galimi testavimo rezultatai pateikiami 1 lentelėje.

1 lentelė. ADF, KPSS ir ZA testų rezultatų interpretavimas

ADF	KPSS	ZA	Bendras testavimo rezultatas
Duomenys stacionarūs	Duomenys stacionarūs	Duomenys stacionarūs	Duomenys griežtai stacionarūs
Duomenys nestacionarūs	Duomenys stacionarūs	Duomenys stacionarūs	Duomenys turi tendencijos stacionarumą ir gali turėti lūžį
Duomenys nestacionarūs	Duomenys nestacionarūs	Duomenys stacionarūs	Duomenys stacionarūs, bet turi lūžį
Duomenys nestacionarūs	Duomenys nestacionarūs	Duomenys nestacionarūs	Duomenys nestacionarūs

1.1.3. Laiko eilučių duomenų prognozavimo tikslumo matavimas

Mokslinėje literatūroje [51] nurodoma, jog naudojant laiko eilutes kaip duomenų šaltinį ir atlikus duomenų paruošimą – t. y. duomenų valymą bei transformavimą į tą pačią skalę, – pagrįstai tikslinga laiko eilučių prognozavimo tikslumo vertinimui pasirinkti *vidutinės absoliučios klaidos* (angl. *Mean Absolute Error, MAE*), *absoliučios klaidos medianos* (angl. *Median Absolute Error, MdAE*) ar *vidutinės kvadratinės klaidos šaknies* (angl. *Root Mean Square Error, RMSE*) metrikas. Jos priskiriamos nuo matavimo skalės priklausančių metrikų klasei [52].

Vidutinė absoliuti klaida skaičiuojama pagal (4) formulę [51]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - f_i^{(m)}\right| = \underset{i=1,n}{mean}\left|y_i - f_i^{(m)}\right| \quad (4)$$

čia y_i – i-oji reikšmė.

$f_i^{(m)}$ – prognozuojamo modelio m pateikiama išprognozuota i-oji reikšmė.

i – į modelio įėjimą paduodamos reikšmės indeksas.

n – į modelio įėjimą paduodamų reikšmių skaičius.

$mean(\cdot)$ – matematinė operacija vidurkio apskaičiavimui.

Šis klaidos skaičiavimo metodas yra jautrus nutolusioms (angl. *Outliers*) duomenų aibės reikšmėms [52]. Mokslinėje literatūroje [53] teigiama, jog nutolusios reikšmės gali būti vertinamos kaip kraštutinis atsitiktinis duomenų nukrypimas. Vidutinė absoliuti klaida yra pateikiama kaip neneigiamas realus skaitmuo, geriausia jo reikšmė yra 0,0.

Absoliučios klaidos mediana gali būti skaičiuojama pagal (5) formulę [51]:

$$\mathrm{MdAE} = \underset{i=1,n}{median}\left|y_i - f_i^{(m)}\right| \quad (5)$$

čia y_i – i-oji reikšmė.

$f_i^{(m)}$ – prognozuojamo modelio m pateikiama išprognozuota i-oji reikšmė.

i – į modelio įėjimą paduodamos reikšmės indeksas.

n – į modelio įėjimą paduodamų reikšmių skaičius.

$median(\cdot)$ – matematinė operacija medianos apskaičiavimui.

Klaida yra apskaičiuojama imant visų absoliučių skirtumų tarp tikslo ir numatytų reikšmių vidurkį. Absoliučios klaidos mediana yra atspari nutolusioms reikšmėms. Absoliučios klaidos mediana yra išreiškiama neneigiamu realiu skaitmeniu, o geriausia jos vertė yra 0,0 [53].

Vidutinės kvadratinės klaidos šaknies skaičiavimas gali būti išreikštas (6) formule [51]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left(y_i - f_i^{(m)}\right)^2\right)} = \sqrt{\underset{i=1,n}{\mathrm{mean}}\left(\left(y_i - f_i^{(m)}\right)^2\right)} \quad (6)$$

čia y_i – i-oji reikšmė.

$f_i^{(m)}$ – prognozuojamo modelio m pateikiama išprognozuota i-oji reikšmė.

i – į modelio įėjimą paduodamos reikšmės indeksas.

n – į modelio įėjimą paduodamų reikšmių skaičius.

$mean(\cdot)$ – matematinė operacija vidurkio apskaičiavimui.

Ši metrika yra jautri didelėms klaidoms, didelėms klaidų variacijoms ir nutolusioms reikšmėms [54].Vidutinės kvadratinės klaidos šaknies reikšmė yra realus neneigiamas skaitmuo, optimali jos vertė lygi 0,0 [55].

Jei būtina palyginti skirtingus prognozavimo metodus, nuo matavimo skalės priklausančių metrikų klasei priskiriamos metrikos negali būti naudojamos [52]. Todėl skirtingų duomenų aibių prognozavimo vertinimui dažnai naudojamos procentinės išraiškos metrikos. Viena jų – *vidutinės absoliučios procentinės klaidos* (angl. *Mean Absolute Percentage Error*, MAPE) metrika. Jos reikšmė apskaičiuojama pagal (7) formulę:

$$MAPE = mean(100(y_t - f_t)/y_t) \quad (7)$$

čia y_t – stebėjimo reikšmė laiku *t.*

f_t – prognozavimo reikšmė laiku *t*.

MAPE pasižymi keliais trūkumais: a) jos reikšmė būna begalinė ar neapibrėžta, jei y_i artėja prie 0; b) nesimetriškumas 0 atžvilgiu. Todėl dažniau yra naudojama *simetriškos vidutinės absoliučios procentinės klaidos* (angl. *Symmetric Mean Absolute Percentage Error,* sMAPE) metrika, kuri yra mažiau jautri mažoms y_i reikšmėms, tačiau, mažėjant y_i reikšmei, mažėja ir atitinkama išprognozuota reikšmė f_t, todėl matuojant kyla dalybos iš skaičiaus, artimo nuliui, grėsmė [52]. sMAPE skaičiuojama pagal formulę (8):

$$sMAPE = mean(200|y_t - f_t|)/|y_t + f_t| \quad (8)$$

čia y_t – stebėjimo reikšmė laiku *t.*

f_t – prognozavimo reikšmė laiku *t*.

1.1.4. Nestacionarių laiko eilučių duomenų prognozavimo problemos

Prognozės tikslumą paveikia du pagrindiniai laiko eilučių duomenų nestacionarumo šaltiniai [15]:

- *Evoliucija*, sukeliama praeities sukrėtimų kumuliacijos (angl. *Evolution from cumulated shocks*). Šis procesas itin būdingas finansiniams ir ekonominiams duomenims, kur jį nulemia lėtas technologijų, demografijos ir prioritetų kitimas [18].
- Staigūs nenumatyti laiko eilučių duomenų poslinkiai laike, vadinami *lokacijos poslinkiais* (angl. *Location Shifts*). Jie sukelia prognozės žlugimą, nes prognozavimo klaidos yra daug didesnės nei būtų, jeigu lokacijos poslinkiai nepasireikštų.

Laiko eilutę galima vadinti *silpnai stacionaria* (angl. *Weak Stationary*), kai jos duomenų vidurkis ir dispersija yra baigtiniai ir laike nekinta. Silpnai stacionarūs duomenys neturi istorijos. Jei

ekonomikos procesai būtų stacionarūs, istoriniai stebėjimo duomenys būtų nereikalingi. Tačiau karai, epidemijos, vulkanų išsiveržimai ir kiti panašūs reiškiniai sukelia ryškius duomenų perstūmimus laike, o štai nuolat vykstančios finansinės inovacijos, mokslo bei technikos atradimai sukelia lėtą duomenų evoliuciją [15].

Mokslinėje literatūroje pateikiama nemažai metodų, skirtų nestacionarių duomenų apdorojimui juos paverčiant stacionariais. Dažniausiai naudojami:

- *Tendencijos pašalinimas* – duomenų aspekto, nulemiančio iškraipymus, pašalinimas. Yra realizuota keletas metodų, skirtų tendencijos pašalinimui. Tai linijinis ir kvadratinis metodai, *Baxter-King*, *Hodrick-Prescott* filtrai ir kiti. Universalaus tendencijos pašalinimo metodo nėra, ir kiekvienu atveju reikia atsižvelgti į tendencijos tipą [56].
- *Diferencijavimas* – skirtumo skaičiavimas tarp gretimų stebėjimų. Diferencijavimas padeda stabilizuoti laiko eilučių duomenų vidurkį. Kartu gali būti pašalinami tendencija ir sezoniškumas [57].
- *Logaritminės transformacijos* pašalina laiko eilučių duomenų dispersijos pokyčius [57].

Tačiau stacionarizuoti nestacionarūs duomenys prognozuojant sukuria kitą problemą: nors prognozės neužtikrintumo (angl. *Uncertainty*) tikimybė išauga, paprastai ji yra nuvertinama [19].

Duomenų nestacionarumas turi ir pranašumų [15]:

- Nestacionariuose duomenyse yra lengviau įžvelgti ilgalaikius ryšius tarp kintamųjų bei įžvelgti priežastinius ryšius.
- Nukrypimų kumuliacija padeda pastebėti ilgalaikius ryšius tarp kintamųjų.
- Nustatyti lokacijos poslinkiai labiau išryškina patvarius ryšius, nes tik tokiais ryšiais susiję kintamieji lieka tokie ir įvykus poslinkiams.
- Lokacijos poslinkiai pakeičia kintamųjų koreliaciją ir išryškina *beprasmę koreliaciją*, kai iš tikro kintamieji nėra susiję. Į tai atsižvelgus pagerėja prognozės tikslumas.
- Praeityje įvykusių lokacijos poslinkių nustatymas leidžia panaudoti testus, vertinančius duomenų prognoze paremtos strategijos teisingumą dar prieš pradedant ją vykdyti.

Kai kurių rūšių neuroniniai tinklai yra visiškai neatsparūs nestacionarių procesų generuojamiems duomenims, ir tokius duomenis prognozuoja klaidingai, tačiau rekurentiniai neuroniniai tinklai dėl galimybės išmokti netiesines funkcijas šiuo metu yra perspektyviausi prognozuojant nestacionarius laiko eilučių duomenis [25].

Apibendrinant galima teigti, kad laiko eilučių duomenys, kurių aibė aprašo konkretaus proceso evoliuciją, dažniausiai yra nestacionarūs – t. y. jų vidurkis, dispersija ir autokoreliacija kinta laike. Pagrindiniai nestacionarumo šaltiniai yra lėtai vykstanti praeities sukrėtimų kumuliacijos sąlygota evoliucija ir staigūs lokacijos poslinkiai. Prognozuojant nestacionarius duomenis jie dažnai apdorojami paverčiant juos stacionariais: pašalinama tendencija, taikomas diferencijavimas arba logaritminės transformacijos. Tačiau tokiu atveju nepagrįstai nuvertinama prognozės neužtikrintumo galimybė bei prarandami kai kurie duomenų nestacionarumo teikiami privalumai. Taigi, nestacionarių duomenų prognozavimas atsižvelgiant į jų nestacionarumą yra aktuali užduotis, kuri sprendžiama tobulinant duomenų analizės įrankius, o tam labiausiai pritaikomi rekurentiniai neuroniniai tinklai.

1.2. Rekurentinių neuroninių tinklų teoriniai pagrindai

Dirbtiniais neuroniniais tinklais vadinamos informacijos apdorojimo sistemos, kurių veikimas paremtas biologinio (žmogaus smegenų) neurono veikimo principu [58]. Dėl galimybės mokytis neuroniniai tinklai taikomi labai plačiai – klasifikuojant, optimizuojant, prognozuojant įvairių mokslo ir pramonės šakų duomenis. Galimybė mokytis reiškia, kad panaudojant turimus duomenis ir mokymo algoritmus, neuroninis tinklas atranda duomenų vidines struktūras ir geba atpažinti jas naujuose duomenyse [24].

1.2.1. Rekurentinių neuroninių tinklų vieta neuroninių tinklų klasifikacijoje

Neuroninių tinklų prigimtis yra daugialypė, todėl juos klasifikuoti sudėtinga. Mokslinėje literatūroje neuroniniai tinklai analizuojami pagal įvairius kriterijus:

Jungčių tipas

Neuroniniai tinklai su tarpsluoksnėmis jungtimis. Turi sujungimus tarp neuronų, esančių gretimuose sluoksniuose. Pažymėtina, kad daugiasluoksniai tinklai turi mažiausiai vieną tarpsluoksnį sujungimą tarp gretimų sluoksnių. Rekurentiniai tinklai yra su visiškai tarpsluoksniais sujungimais.

Neuroniniai tinklai su jungtimis sluoksnio viduje. Tokio tinklo pavyzdžiu gali būti Adaptyvaus rezonanso teorijos neuroninis tinklas.

Neuroniniai tinklai su neuronais, turinčiais jungtį į save. Turi ryšius, kurie prasideda ir baigiasi sujungimu prie to paties neurono. Tai sukuria grįžtamąjį ryšį pačiam neuronui (suteikiantį savęs slopinimą arba savęs sužadinimą).

Neuroniniai tinklai su viršsluoksnėmis jungtimis. Šių neuroninių tinklų jungtys sieja ne greta esančius sluoksnius. Tai yra, jos „peršoka“ mažiausiai vieną sluoksnį.

Neuroniniai tinklai su visų rūšių jungtimis. Tokie neuroniniai tinklai turi tarpsluoksnes jungtis, jungtis sluoksnio viduje ir neuronus su jungtimi į save [59].

Simetriškumas

Simetriška arba dvipusė jungtis leidžia signalui sklisti bet kuria kryptimi, todėl sukuriamas grįžtamasis ryšys. Simetriškas neuroninis tinklas turi išskirtinai tik simetriškas jungtis ir vieną ar daugiau simetriškų svorio matricų.

Asimetriška arba vienpusė jungtis leidžia signalui sklisti viena kryptimi. Asimetriškas neuroninis tinklas turi vieną ar daugiau asimetriškų jungčių [60].

Jungčių lygmuo:

Pirmojo lygmens neuroniniai tinklai – tai neuroniniai tinklai, kurių skirtingų įėjimų neuronai nėra apjungiami panaudojant apjungimo funkciją. Tačiau tam, kad į neuroninio tinklo įėjimą

patenkanti informacija būtų panaudota efektyviau, gali būti panaudotos aukštesniojo lygmens jungtys.

Aukštesniojo ar aukšto lygmens jungtys apjungia įėjimus iš skirtingų neuronų panaudodamos apjungimo funkciją, kuri dažniausiai yra daugyba. Apjungiamų įėjimų skaičius vadinamas neuroninio tinklo lygmeniu. Neuroninis tinklas, kurio aukščiausias jungties lygmuo yra Ω, vadinamas Ω lygmens neuroniniu tinklu [59].

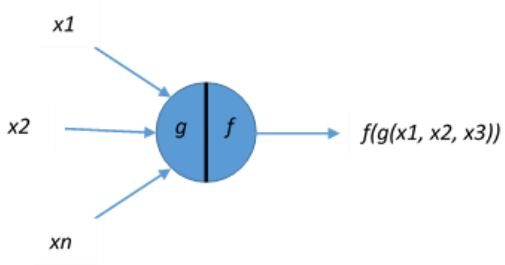
Sinchroniškumas

Sinchroniniuose neuroniniuose tinkluose visų elementų išėjimas apskaičiuojamas akimirksniu. Tai įmanoma, jei tinklo topologijoje nėra ciklų.

Asinchroniniai neuroniniai tinklai skirtingu stochastiniu būdu parinktu laiko momentu apskaičiuoja kiekvieno bloko aktyvumą nepriklausomai nuo visų kitų blokų [61].

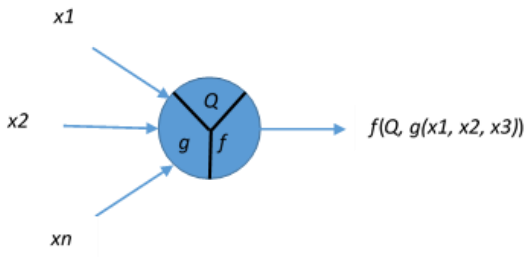
Būsenos išsaugojimas

Neuroninis tinklas be išsaugotos neurono būsenos (1 pav.). Jį sudaro du elementai: primityvi *n* argumentų integravimo funkcija *g* (dažniausiai tai sumavimo funkcija), kuri užtikrina, kad *n* argumentų yra sumažinama į vieną argumentą, o taip pat aktyvavimo funkcija *f*.



1 pav. Neuroninis tinklas be išsaugotos neurono būsenos pagal R. Rojas [61]

Neuroninis tinklas su išsaugota neurono būsena (2 pav.) turi trečią elementą *Q*, kuriame po kiekvieno skaičiavimo yra išsaugoma neurono būsena. Išsaugota būsena *Q* gali keisti neurono išvestį [61].



2 *pav. Neuroninis tinklas su išsaugota neurono būsena pagal R. Rojas [61]*

Rekurentinis neuroninis tinklas – tai neuroninių tinklų rūšis, kurios pagrindinė savybė – galimybė kiekviename laiko žingsnyje grąžinti signalą iš išėjimo sluoksnio į įėjimą [62]. Rekurentinis

neuroninis tinklas yra arba simetriškas, arba turi vieną ar daugiau ciklinių jungčių. Pavyzdžiui, neuroninis tinklas su neuronais, turinčiais jungtį į save, yra rekurentinis, kadangi neurono jungtis į save yra ciklinė jungtis [59]. Rekurentiniai tinklai gali turėti viršsluoksnes, aukštesniojo lygmens jungtis, gali turėti arba ne išsaugotą neurono būseną.

1.2.2 Rekurentinių neuroninių tinklų rūšys

Paprastas rekurentinis neuroninis tinklas (angl. *Simple Recurrent Neural Network,* SRNN) – tai neuroninis tinklas, skirtas nuoseklių duomenų sekų x(1)...x(n) apdorojimui. SRNN gali apdoroti ilgas duomenų sekas, o pateikiamas rezultatas priklauso nuo gaunamų duomenų eilės tvarkos. Šis neuroninis tinklas turi didelį trūkumą – išnykstantį arba sprogstantį gradientą [10]. SRNN tinklai naudojami laiko eilučių duomenų, garso, paveiksliukų ir vaizdo įrašų apdorojimui [63] [64] [11].

Hopfildo neuroninis tinklas (angl. *Hopfield Network,* HN) – tai visiškai susietas neuroninis tinklas, turintis simetrišką ryšių matricą [12]. Tai reiškia, jog kiekvienas tinklo mazgas yra susietas su visais mazgais, išskyrus save patį [65]. Tai, kad mazgas nesusietas su savimi, leidžia išvengti poveikio savo būsenai per atgalinį ryšį. Šis tinklas vadinamas tinklu su asociatyvia atmintimi. Kadangi Hopfildo tinklas visada konverguoja į stabilią būseną, ši stabili būsena gali būti traktuojama kaip šablonas. Hopfildo tinklas yra naudojamas trūkstamos informacijos atstatymui [11].

Ilga trumpalaikė atmintis (angl. *Long Short Term Memory,* LSTM) – tai specifinė rekurentinio neuroninio tinklo architektūra, skirta modeliuoti laiko eilučių sekas ir jų tolimas priklausomybes tiksliau, nei tai daro SRNN. LSTM gali išmokti 1000 laiko žingsnių seką, išlaikydamas pastovią išėjimo klaidą [66]. Šis neuroninis tinklas sukonstruotas taip, kad sugebėtų išvengti išnykstančio arba sprogstančio gradiento, būdingo paprastam rekurentiniam neuroniniam tinklui. Kiekvienas LSTM neuronas turi tris loginius elementus: įėjimo, išėjimo bei užmiršimo. Jų paskirtis – išsaugoti informaciją, sustabdant ir atstatant jos srautą. Tokia konstrukcija šiam neuroniniam tinklui leidžia efektyviai mokytis tada, kai tinklui paduodami įvykiai atskirti ilgais laiko tarpais [67], nors, lyginant su daugiasluoksniu perceptonu, apdorojant laiko eilučių duomenis, LSTM rodo prastesnius prognozavimo rezultatus [9].

Valdomas rekurentinis vienetas (angl. *Gated Recurrent Unit,* GRU), kaip ir LSTM, yra atsparus išnykstančiam arba sprogstančiam gradientui. Tačiau jo konstrukcija paprastesnė, jis lengviau mokomas, tai yra, jo vidinei būsenai atnaujinti reikia atlikti mažiau skaičiavimų. Vietoj trijų LSTM loginių elementų – įėjimo, užmiršimo ir išėjimo, – valdomas rekurentinis vienetas turi du: atnaujinimo ir išvalymo. Skirtingai negu LSTM, GRU neturi jokios neurono būsenos išsaugojimo atminties. Jis gali būti greičiau apmokomas negu LSTM, ir su mažesniu duomenų kiekiu [4] [11].

Neuroninė Tiuringo mašina (angl. *Neural Turing Machine*, NTM) – tai rekurentinio tinklo porūšis, kurio architektūrą sudaro du pagrindiniai komponentai: neuroninio tinklo valdiklis ir atminties bankas. Kaip ir daugumoje neuroninių tinklų, valdiklis sąveikauja su išoriniu pasauliu per įvesties ir išvesties vektorius. Skirtingai nuo SRNN, jis taip pat sąveikauja su atminties matrica, naudodamas pasirinktines skaitymo ir rašymo operacijas. Sąveikavimą su matrica galima optimizuoti gradientinio nusileidimo būdu. Analogiškai kaip įprastoje Tiuringo mašinoje, šio neuroninio tinklo skaitymo ir rašymo į atmintį operacijas galima apibūdinti kaip operacijas įrašymo ir skaitymo galvutėmis. Neuroninė Tiuringo mašina, turinti LSTM neuroninio tinklo valdiklį, gali išmokti paprastus algoritmus, tokius kaip kopijavimas ir rūšiavimas [68].

Dvikryptis neuroninis tinklas (angl. *Bidirectional Recurrent Neural Network,* BiRNN) yra sudarytas iš dviejų rekurentinių neuroninių tinklų, skaitančių įėjimo duomenis skirtingomis kryptimis. Pavyzdžiui, vienas rekurentinis neuroninis tinklas gali laiko eilutes skaityti laiko didėjimo kryptimi, o kitas – laiko mažėjimo. Kiekvieno laiko momento išėjimas priklausys nuo abiejų rekurentinių tinklų paslėptų sluoksnių būsenos. Dvikryptis neuroninis tinklas leidžia rasti išėjimo priklausomybę ne tik nuo visų anksčiau pateiktų, bet ir nuo būsimų įėjimo duomenų [69].

Rezervuaro neuroninis tinklas (angl. *Echo State Network,* ESN) turi tik vieną paslėptą sluoksnį, vadinamą rezervuaru. Rezervuaro viduje ryšiai tarp neuronų yra pastovūs ir atsitiktiniai, tačiau ryšiai su išoriniu sluoksniu yra apmokomi. Rezervuaro būsena priklauso nuo ankstesnės rezervuaro būsenos, o taip pat nuo ankstesnių įėjimo ir išėjimo signalų. Šie neuroniniai tinklai yra greiti, kadangi turi tik vieną paslėptą sluoksnį [70].

Boltzmano mašina (angl. *Boltzmann Machine*, BM) yra simetriškai sujungtų, į neuronus panašių mazgų tinklas, kurio mazgai gali stochastiniu būdu priimti sprendimą įsijungti ar išsijungti. Boltzmano mašina turi paprastą apmokymo algoritmą, kuris leidžia aptikti sudėtingus apmokymo duomenų dėsningumus. Tinklo, turinčio daug požymių detektorių sluoksnių, apmokymas yra labai lėtas; greičio problema yra išspręsta apribotoje Boltzmano mašinoje, turinčioje tik vieną požymių detektorių sluoksnį. Boltzmano mašina gali būti priskiriama stochastiniam rekurentiniam neuroniniam tinklui, ir gali būti panaudota dviem visiškai skirtingiems skaičiavimo uždaviniams – paieškai ir apmokymui [71]. Šio neuroninio tinklo struktūra labai panaši į Hopfildo neuroninį tinklą, tačiau dalis jo neuronų naudojami duomenų įvedimui, o dalis – kaip paslėpti neuronai. Įėjimo neuronai tinklo veikimo metu tampa išėjimo neuronais [72] [11].

1.2.3 Rekurentinių neuroninių tinklų charakteristikos

Dėl galimybės kiekviename laiko žingsnyje grąžinti signalą iš išėjimo sluoksnio į įėjimą rekurentiniai neuroniniai tinklai yra ypač efektyvūs tais atvejais, kai konteksto supratimas lemia

rezultatą, todėl jų panaudojimas laiko eilučių tyrimams laikomas itin perspektyviu. Laiko eilučių tyrimams dažniausiai naudojami rekurentiniai neuroniniai tinklai yra SRNN, LSTM ir GRU [29] [30] [31] [32] [33]. Toliau pateikiamos šių tinklų charakteristikos.

1.2.3.1 Paprastas rekurentinis neuroninis tinklas, SRNN

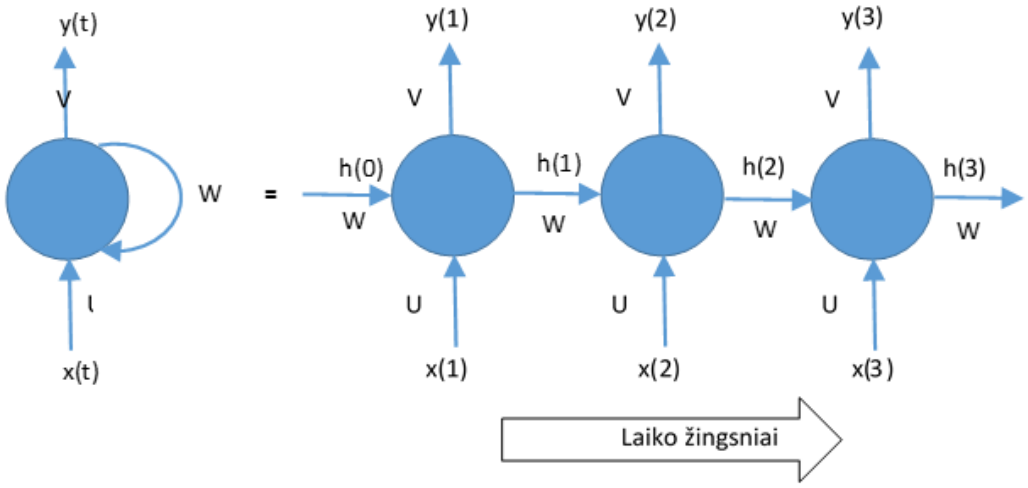
Tradiciškai daugiasluoksniuose tiesioginio sklidimo neuroniniuose tinkluose laikoma, jog visos į įėjimą paduodamos eilutės yra viena nuo kitos nepriklausančios. Tačiau apdorojant laiko eilutės duomenis, tekstą, garsą ar vaizdą, turi reikšmę duomenų padavimo eilės tvarka [23]. Priklausomybė nuo ankstesnių duomenų vadinama ilgalaike tendencija. Rekurentinio neuroninio tinklo neuronuose ši tendencija atspindima paslėptoje būsenoje, arba atmintyje, kurioje saugoma visos ankstesnės informacijos suvestinė. Paslėptos būsenos reikšmę apibrėžia (9) formulė:

$$h_t = \varphi(h_{t-1}, x_t), \tag{9}$$

čia h_t ir h_{t-1} – paslėptos būsenos žingsnyje t ir $t-1$ reikšmė;

x_t – įėjimo reikšmė laiko momentu t.

Rekurentinio neuroninio tinklo celę taip pat galima atvaizduoti grafiškai (3 pav.)



3 pav. Rekurentinio neuroninio tinklo celės schema pagal A. Gulli [73]

Galima pastebėti, jog rekurentinio neuroninio tinklo celė t laiko momentu savo įėjime gauna reikšmę x_t, ir išėjime išduoda reikšmę y_t. Dalis y_t reikšmės (paslėpta būsena h_t) paduodama atgal į celės įėjimą tam, kad ją galima būtų panaudoti kitame žingsnyje $t+1$. Tradicinio neuroninio tinklo atveju jo parametrai saugomi svorių matricoje, tačiau rekurentinio neuroninio tinklo parametrai nustatomi trimis svorių matricomis – U, V ir W, kurios atitinka neuroninio tinklo įėjimą, išėjimą ir paslėptą būseną. Dešinėje paveiksliuko pusėje pavaizduota rekurentinio neuroninio tinklo su trimis

sluoksniais skleistinė. Svorių matricos *U, V* ir *W* yra naudojamos visų trijų žingsnių metu, kadangi kiekviename žingsnyje naudojamos tos pačios operacijos.

Rekurentinio neuroninio tinklo skaičiavimus taip pat galima aprašyti lygtimis. Tinklo būsena *t* laiko momentu yra aprašoma vektoriaus h_t reikšme, kuri yra lygi hiperboliniam tangentui svorių matricos *W* ir paslėptos būsenos h_{t-1} sandaugos laiko momentu *t-1* sumai su sandauga svorių matricos *U* su įėjimo reikšme x_t laiko momentu *t* (10):

$$h_t = \tanh(Wh_{t-1} + Ux_t), \tag{10}$$

čia h_t – paslėpta būsena laiko momentu *t*;
W – svorių matrica;
h_{t-1} – paslėpta būsena laiko momentu *t-1*;
U – svorių matrica;
x_t – įėjimo reikšmė laiko momentu *t*.

Laiko momentu *t* išėjimo vektorius y_t yra lygus svorių matricos *V* ir paslėptos būsenos h_t laiko momentu *t* sandaugos *softmax* funkcijai (11):

$$y_t = softmax(Vh_t), \tag{11}$$

čia y_t – išėjimo vektorius laiko momentu *t*;
V – svorių matrica;
h_t – paslėpta būsena laiko momentu *t*.

Hiperbolinio tangento funkcija pasirinkta todėl, kad jos antroji išvestinė labai lėtai artėja prie nulio, ir taip ši funkcija leidžia išvengti nykstančio gradiento [73].

Paprasto rekurentinio neuroninio tinklo mokymas

Sakoma, kad kompiuterinė programa tam tikros užduočių klasės *T* ir kokybės mato *P* atžvilgiu mokosi patirties *E* pagrindu, jei kokybė *T* užduotyse, išmatuota su *P* pagalba, auga augant patirčiai *E* [74]. Rekurentinio neuroninio tinklo, kurio įėjimas u_t, būsena x_t laiko momentu *t* gali būti aprašyta (12) formule:

$$x_t = F(x_{t-1}, u_t, \theta), \tag{12}$$

čia x_t – neuroninio tinklo būsena laiko momentu *t*;
x_{t-1} – neuroninio tinklo būsena laiko momentu *t-1*;
u_t – neuroninio tinklo įėjimas.

Arba parametrizuota (13) formule:

$$x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b, \tag{13}$$

čia x_t – neuroninio tinklo būsena laiko momentu *t*;

W_{rec} – rekurentinė svorių matrica;

σ – elementarioji funkcija;

x_{t-1} – neuroninio tinklo būsena laiko momentu *t-1*;

W_{in} – įėjimo svorių matrica;

u_t – neuroninio tinklo įėjimas laiko momentu *t*.;

b – postūmis.

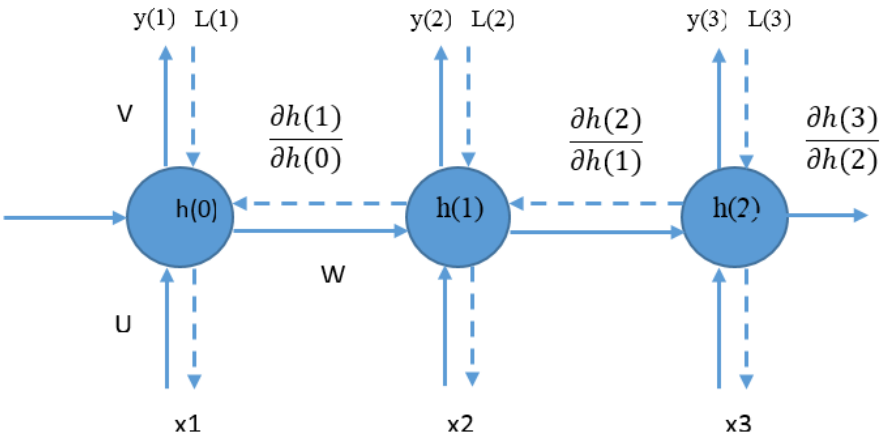
Šiuo atveju, modelio parametrai yra nustatomi rekurentine svorių matrica W_{rec}, postūmiu b ir įėjimo svorių matrica W_{in}, surinkta į θ bendruoju atveju. x_0 yra nustatoma naudotojo, prilyginta nuliui arba išmokta, o σ yra elementarioji funkcija. Tinklo efektyvumą atspindintys nuostoliai išreikšti (14) formule:

$$\varepsilon = \Sigma_{1 \le t \le T} \varepsilon_t \,, \tag{14}$$

čia ε – tinklo efektyvumą atspindintys nuostoliai;

ε_t – tinklo efektyvumą atspindintys nuostoliai laiko momentu *t*.

Vienas iš būdų apskaičiuoti būtinus gradientus yra atgalinio sklidimo laiku panaudojimas. Rekurentiniame neuroniniame tinkle kiekviename žingsnyje yra naudojami vienodi parametrai, tad kiekviename išėjime gradientas priklauso ne tik nuo tuometinio žingsnio, bet ir nuo visų ankstesnių žingsnių. Šis procesas vadinamas *atgaliniu sklidimu laiku* (angl. *Backpropagation through Time*, *BPTT*), ir yra pagal A. Gulli [73] pavaizduotas 4 pav.



4 pav. Trisluoksnio rekurentinio neuroninio tinklo schema pagal A. Gulli [73]

Informacijos eiga tiesioginio sklidimo metu pavaizduota ištisinėmis rodyklėmis, o atgalinio sklidimo metu – punktyrinėmis. Tiesioginio informacijos sklidimo metu neuroninis tinklas sukuria prognozes, kurios lyginamos su žymėmis tam, kad kiekviename laiko žingsnyje apskaičiuoti praradimus L_t. Atgalinio informacijos sklidimo metu kiekviename laiko žingsnyje apskaičiuojami

parametrų *U, V* ir *W* praradimų funkcijos gradientai ir gradientų suma panaudojama parametrų atnaujinimui [73].

Dingstančio ir sprogstančio gradiento problema

W matricos, kurioje užkoduoti ilgalaikių priklausomybių svoriai, praradimų funkcijos gradientas pateiktas (15) lygtyje:

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W}, \tag{15}$$

čia L – praradimai;

W – svorių matrica;

L_t – praradimai laiko momentu *t*.

Ši matrica yra dingstančio ir sprogstančio gradiento priežastis. *Sprogstančiu gradientu* vadinamas staigus gradiento padidėjimas neuroninio tinklo mokymo metu. *Dingstančiu gradientu* vadinamas sprogstančiam gradientui atvirkščias reiškinys, kai gradientas greitai artėja prie 0. Ir sprogstančio, ir dingstančio gradiento atveju neuroniniam tinklui yra neįmanoma mokytis [75].

Kiti du praradimų funkcijos gradientai matricų *U* ir *V* atžvilgiu taip pat yra sumuojami kiekviename laiko žingsnyje.

Praradimų funkcijos gradientą paskutiniame laiko žingsnyje (*t*=3), panaudojus sudėtingos funkcijos diferencijavimo taisyklę, galime išskaidyti į tris komponentes (16).

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial y_3} * \frac{\partial y_3}{\partial h_2} * \frac{\delta h_2}{\delta W}, \tag{16}$$

čia L – praradimų funkcija;

W – svorių matrica;

h – paslėpta būsena.

Paslėptos būsenos gradientą h_2 atžvilgiu *W* galima pateikti suma kiekvieno paslėptos būsenos gradiento ankstesnio atžvilgiu (17).

$$\frac{\partial L_3}{\partial W} = \sum_{t=0}^{2} \frac{\partial L_3}{\partial y_3} * \frac{\partial y_3}{\partial h_2} * \frac{\partial h_2}{\partial h_t} * \frac{\partial h_t}{\partial W}, \tag{17}$$

čia L – praradimų funkcija;

W – svorių matrica;

h – paslėpta būsena.

Paslėptos būsenos gradientą ankstesnio atžvilgiu galima išskaidyti į dabartinio paslėptos būsenos gradiento ir ankstesnio gradiento sandaugą (18).

$$\frac{\partial L_3}{\partial W} = \sum_{t=0}^{2} \frac{\partial L_3}{\partial y_3} * \frac{\partial y_3}{\partial h_2} * \left[\prod_{j=t+1}^{2} \frac{\partial h_j}{\partial h_{j+1}} \right] * \frac{\partial h_t}{\partial W}, \quad (18)$$

čia L – praradimų funkcija;

W – svorių matrica;

h – paslėpta būsena.

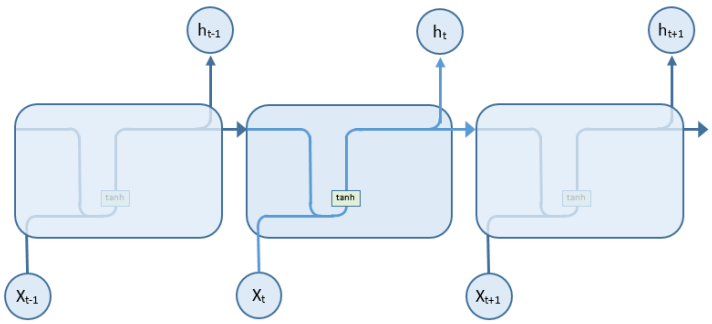
Analogiškai apskaičiuojami praradimų funkcijų L_1 ir L_2 gradientai W atžvilgiu, o po to jų suma naudojama W atnaujinimui.

Tuo atveju, kai atskiri paslėptos būsenos gradientai ankstesnio gradiento atžvilgiu mažesni už 1, signalui sklindant atgal per kelis laiko žingsnius, gradientų sandauga tampa vis mažesnė, ir tai sukelia gradientų išnykimą. Dėl nykstančio gradiento efekto nutolusiuose laiko žingsniuose gradientai nedaro įtakos mokymo procesui, todėl tinklas negali išmokti ilgalaikių priklausomybių. Iš kitos pusės, jei gradientai didesni už 1, jų sandauga greitai auga, ir pasireiškia sprogstamasis gradientas. Jo reikšmė perpildo kintamąjį, ir mokymo procesas baigiasi klaida.

Siekiant išvengti išnykstančių gradientų taikomi įvairūs būdai: tinkamas matricos W inicializavimas, aktyvavimo funkcijos *ReLU* panaudojimas vietoje aktyvavimo funkcijos *tanh* ar sluoksnių apmokymas mokymo be mokytojo būdu. Tačiau efektyviausia yra panaudoti *ilgos trumpalaikės atminties* ar *valdomo rekurentinio vieneto* neuroninio tinklo architektūras, kurios ir yra specialiai sukurtos tam, kad išvengti nykstančio ir sprogstančio gradientų.

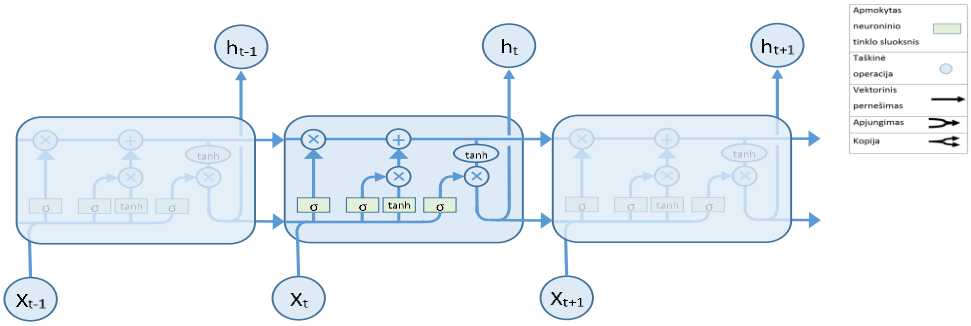
1.2.3.2 Ilgos trumpalaikės atminties neuroninis tinklas, LSTM

Visi rekurentiniai neuroniniai tinklai turi neuroninio tinklo pasikartojančių modulių grandinę. Standartiniuose rekurentiniuose neuroniniuose tinkluose šių pasikartojančių modulių struktūra yra labai paprasta, pavyzdžiui, sudaryta iš vieno *tanh* sluoksnio (pagal Ch. Ollah [76] pavaizduota 5 pav.).



5 pav. Rekurentinio neuroninio tinklo schema pagal Ch. Ollah [76]

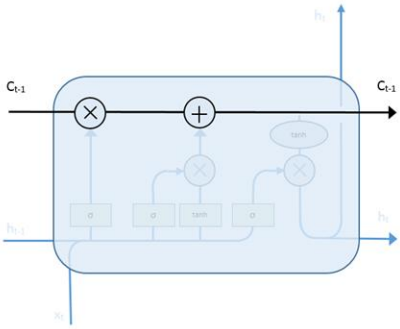
LSTM tinklai taip pat sudaryti iš grandine sujungtų neuroninio tinklo modulių, tačiau patys moduliai yra žymiai sudėtingesni. Vietoje vieno, juose yra keturi tam tikru būdu sujungti sluoksniai (6 pav.).



6 pav. LSTM neuroninio tinklo schema pagal Ch. Ollah [76]

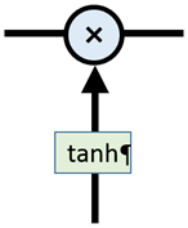
Sutartiniai schemų žymėjimai

Pagrindinis LSTM tinklo elementas – celės būsena (angl. *Cell State*). Ji žymima horizontalia linija celės schemos viršutinėje dalyje (7 pav.).



7 pav. LSTM celės būsenos žymėjimas pagal Ch. Ollah [76]

Informacija laisvai praeina pro celės būseną, tačiau gali būti joje ir ištrinta. Tai atliekama *loginiuose elementuose* (angl. *Gates*), sudarytuose iš neuroninio tinklo sigmoidinio sluoksnio bei taškinio sudauginimo operacijos. Tai suteikia galimybę praleisti informaciją priklausomai nuo tam tikrų sąlygų. Loginio LSTM elemento žymėjimas pagal Ch. Ollah [76] yra pateiktas 8 pav.



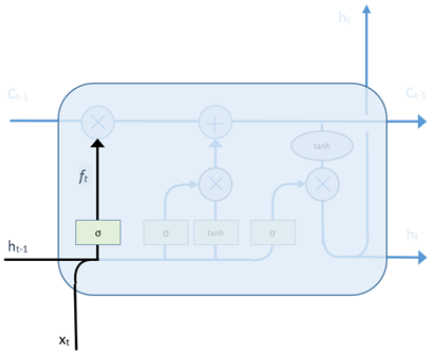
8 pav. LSTM loginis elementas pagal Ch. Ollah [76]

Sigmoidinis elementas pateikia 0...1 intervalo reikšmę, kuri nurodo kokią kiekvieno bloko informacijos dalį elementas turi praleisti toliau į tinklą. Čia 0 reiškia „nepraleisti nieko“ o 1 –

„praleisti viską“. LSTM celė turi tris sigmoidinius elementus, kurie suteikia galimybę apsaugoti ir kontroliuoti celės būseną.

LSTM celės veiklos analizė

Pirmame žingsnyje *užmiršimo loginio elemento sluoksnis* (angl. *Forget Gate Layer*) nusprendžia, kokią informaciją iš celės būsenos galima ištrinti (9 pav.). Atsižvelgdamas į įėjimus h_{t-1} ir x_t, jis kiekvienam celės būsenos skaičiui C_{t-1} pateikia 0...1 diapazono skaičių, čia 0 reiškia „nepraleisti nieko“, o 1 – „praleisti viską“.



9 pav. LSTM užmiršimo loginio elemento schema pagal Ch. Ollah [76]

LSTM užmiršimo elemento veikla gali būti aprašyta (19) lygtimi.

$$f_t = \sigma(W_t * [h_{t-1}, x_t] + b_i), \tag{19}$$

čia f_t – užmiršimo elemento funkcija laiko momentu *t*;

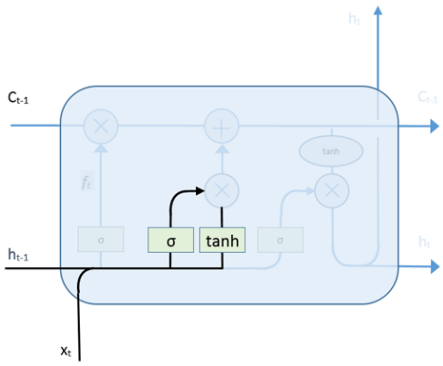
W_t – svorių matrica laiko momentu *t*;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*;

b – postūmis.

Antrame žingsnyje yra parenkama informacija, kuri bus išsaugota celės būsenoje (10 pav)



10 pav. LSTM celės būsenoje išsaugomos informacijos pasirinkimo proceso schema pagal Ch. Ollah [76]

Šis žingsnis turi dvi dalis. Pirmojoje *įvesties loginio elemento sluoksnis* (angl. *Input Gate Layer*) pasirenka reikšmes i_t, kurios bus atnaujintos. Šis procesas aprašomas (20) lygtimi:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i), \quad (20)$$

čia i_t – reikšmės, kurios bus atnaujinamos laiko momentu *t*;

σ – elementarioji funkcija;

W – svorių matrica;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*;

b – postūmis.

Antrojoje tanh sluoksnis sukuria naujų reikšmių vektorių (C_t)̃, kuriuo celės būsena bus papildyta. Jo išraiška pateikiama (13) lygtyje.

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_C), \quad (21)$$

čia $\tilde{C}_t$ – naujų reikšmių vektorius laiko momentu *t*, tuo vektoriumi celės būsena bus papildyta;

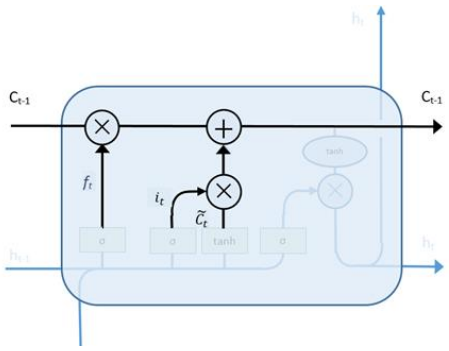
W – svorių matrica;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*;

b – postūmis.

Trečiajame žingsnyje ankstesnė celės būsena C_{t-1} keičiama į naują būseną C_t. Ankstesnė celės būsena C_{t-1} yra dauginama iš f_t reikšmės, pamirštant tai, ką buvo pasirinkta pamiršti. Tada prie celės būsenos yra pridedama i_t*C_t (11 pav.):



11 pav. LSTM celės būsenos pakeitimo proceso schema pagal Ch. Ollah [76]

Matematinė proceso išraiška pateikiama (22) lygtimi.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (22)$$

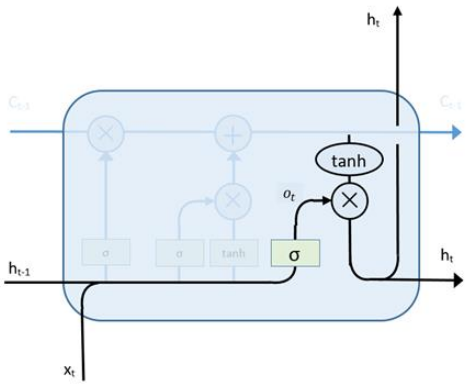
čia C_t – celės būsena laiko momentu *t*;

C_{t-1} – celės būsena laiko momentu *t-1*;

i_t – reikšmės, kurios bus atnaujintos;

$\tilde{C}_t$ – naujų reikšmių vektorius laiko momentu *t*, tuo vektoriumi celės būsena bus papildyta.

Ketvirtajame žingsnyje bus pasirenkamos išvesties reikšmės. Jos remsis celės būsena, bet bus filtruojamos (12 pav.). Pirmiausia, sigmoidinis sluoksnis parenka, kuri celės būsenos dalis bus pateikiama išvestyje (23):



12 pav. LSTM išvesties informacijos pasirinkimo proceso schema pagal Ch. Ollah [76]

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{23}$$

čia o_t – išvestyje pateikiama celės būsenos dalis;

σ – elementarioji funkcija;

W – svorių matrica;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*;

b – postūmis.

Tada celės būsena praeina pro *tanh* sluoksnį ir yra keičiama į diapazone -1...1 esančias reikšmes, o po to dauginama iš sigmoidinio elemento (24):

$$h_t = o_t * \tanh(C_t), \tag{24}$$

čia h_t - išvesties reikšmės laiko momentu *t*;

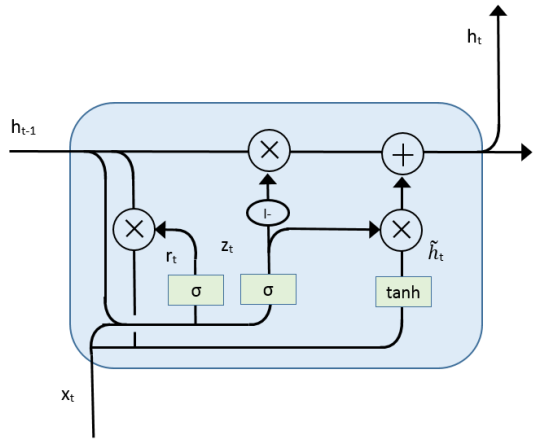
o_t – išvestyje pateikiama celės būsenos dalis;

C_t – celės būsena laiko momentu *t*;

Tokiu būdu išvestį pasiekia tik pasirinktos reikšmės.

1.2.3.3 Valdomas rekurentinis vienetas, GRU

Valdomas rekurentinis vienetas yra LSTM tinklo variantas, pasižymintis savybe kartu valdyti pamiršimo faktorių ir pasirinkimą ar atnaujinti vieneto būseną. GRU schema pagal Ch. Ollah [76] pateikiama 13 pav.



13 pav. GRU schema pagal Ch. Ollah [76]

GRU neuroninio tinklo struktūra yra paprastesnė ir jis yra greitesnis už LSTM. Prognozuojant laiko eilučių duomenų sekas šie abu tinklai pateikia panašaus tikslumo rezultatus [77].

GRU atnaujinimo elemento reikšmė z_t skaičiuojama pagal (25) formulę:

$$z_t = \sigma(W_z * [h_{t-1}, x_t]), \tag{25}$$

čia z_t - valdomo rekurentinio vieneto atnaujinimo ventilio reikšmė;

σ – elementarioji funkcija;

W_z – valdomo rekurentinio vieneto atnaujinimo ventilio svorių matrica;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*.

Pakartotinės kelties elemento reikšmė skaičiuojama pagal (26) formulę [76]:

$$r_t = \sigma(W_r * [h_{t-1}, x_t]), \tag{26}$$

čia r_t - valdomo rekurentinio vieneto pakartotinos kelties ventilio reikšmė;

σ – elementarioji funkcija;

W_r – valdomo rekurentinio vieneto pakartotinos kelties ventilio svorių matrica;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*.

Išėjimo reikšmė $\tilde{h}_t$ laiko momentu *t* skaičiuojama pagal (27) formulę:

$$\tilde{h}_t = \tanh(W * [r_t * h_{t-1}, x_t]), \quad (27)$$

čia $\tilde{h}_t$ – išėjimo reikšmė laiko momentu *t*;

W – svorių matrica;

r_t - valdomo rekurentinio vieneto pakartotinos kelties ventilio reikšmė;

h_{t-1} – įėjimas laiko momentu *t-1*;

x_t – įėjimas laiko momentu *t*.

Išėjimo *h* reikšmė laiko momentu *t* skaičiuojama pagal (28) formulę [76]:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \quad (28)$$

čia h_t – išėjimo reikšmė laiko momentu *t*;

z_t - valdomo rekurentinio vieneto atnaujinimo ventilio reikšmė;

h_{t-1} – įėjimas laiko momentu *t-1*;

$\tilde{h}_t$ – išėjimo reikšmė laiko momentu *t*.

Apibendrinant galima konstatuoti, kad sudėtingoje neuroninių tinklų taksonomijos sistemoje rekurentiniai tinklai dėl savo savybių (simetriškumo, ciklinių jungčių) užima ypatingą vietą. Rekurentinių tinklų galimybė kiekviename laiko žingsnyje grąžinti signalą iš išėjimo sluoksnio į įėjimą daro juos ypač efektyvius laiko eilučių tyrimams. Rekurentiniai neuroniniai tinklai turi keletą rūšių, iš kurių dažniausiai naudojamos laiko eilučių prognozavimui yra SRNN, LSTM ir GRU. SRNN tinklas pasižymi sprogstančio arba išnykstančio gradiento problema, todėl negali išmokti aikių priklausomybių. LSTM ir GRU tinklai pasižymi sudėtinga specifine architektūra, kuri leidžia ne tik išvengti sprogtsančio arba išnykstančio gradiento bei modeliuoti ilgas laiko eilučių sekas ir jų tolimas priklausomybes, bet ir prognozuoti žymiai tiksliau.

1.3. Programinės įrangos kūrimo projekto veiklos modeliai

Viena iš pagrindinių programų architektūros sąvokų yra *Programų kūrimo gyvavimo ciklas* (angl. *Software Development Life Cycle*, SDLC). Tai yra koncepcinis pagrindas, aprašantis visą programinės įrangos kūrimo projekto veiklą nuo planavimo iki priežiūros. Šis procesas apima kelis modelius, kurių kiekvienas numato jam būdingas užduotis ir veiklas. Egzistuoja nemažai *Programų kūrimo gyvenimo ciklo modelių*, kiekvienas iš kurių gali būti pasirinktas atsižvelgiant į kuriamai programinei įrangai keliamus reikalavimus, programavimo terminus, programuojančios įmonės vidinę kultūrą, užsakovo reikalavimus ir kita. Svarbiausi modeliai yra išvardinti 2 lentelėje [78] [79] [80] [81] [82] [83] [84]:

2 lentelė. Svarbiausi Programų kūrimo gyvenimo ciklo modeliai

Modelis	Apibūdinimas	Privalumai	Trūkumai	Panaudos atvejai
Paslankusis modelis (angl. *Agile model*)	Daugelio iteracijų modelis, kiekvienos iš kurių metu klientas turi galimybę įvertinti rezultatą. Projekto metu vykdytojas gali greitai prisitaikyti prie pasikeitusių kliento reikalavimų.	• Modelis sutrumpina projekto kūrimo laiką. • Programos kūrimo metu yra vykdomi kūrimo proceso pakeitimai. • Projektas yra dalinamas į mažas dalis. • Dėka galimybės keisti projektą, rizikos yra nedidelės.	• Dėl nuolatinių pakeitimų sunku įvertinti projekto kaštus. • Dėl nuolatinių pakeitimų projektas gali vėluoti. • Ne kiekvienos programų kūrimo įmonės vidinė kultūra leidžia dirbti pagal *Paslankųjį modelį*.	• Projekto rengimo metu būtini nuolatiniai pakeitimai. • Tam, kad projektas būtų pradėtas, būtinas tik minimalus planavimas.
Iteratyvusis modelis (angl. *Iterative model*)	Modelis, kuriame programos gyvenimo ciklo pradžioje dėmesys sutelkiamas supaprastintam programos variantui, kuris projekto eigoje sudėtingėja, įgydamas vis gilesnių ir platesnių funkcijų.	• Kai kurios funkcijos gali būti greitai sukurtos programos gyvenimo ciklo pradžioje. • Gali būti naudojamas lygiagretus programos kūrimas. • Kūrimo procesas yra lengvai išmatuojamas. • Rizikas valdyti lengva, nes didelės rizikos užduotys vykdomos pirmiausia.	• Procesas reikalauja daugiau resursų negu *Waterfall* metodas. • Galimi programos architektūros ar dizaino kūrimo nesklandumai, nes ne visi reikalavimai aiškūs programos kūrimo pradžioje. • Netinka mažiems projektams. • Programos kūrimo procesą sunku valdyti. • Rizika gali būti nesuvaldyta net pasiekus galutinį etapą. • Rizikos įvertinimui reikalingi aukštos kvalifikacijos specialistai.	• Projektas yra didelės apimties. • Pagrindiniai reikalavimai yra apibrėžti, tačiau detalės gali kisti. • Kūrėjų komanda mokosi dirbti su nauja technologija.

2 lentelė (tęsinys). Svarbiausi Programų kūrimo gyvenimo ciklo modeliai.

Modelis	Apibūdinimas	Privalumai	Trūkumai	Panaudos atvejai
Spiralinis modelis (angl. *Spiral model*)	Modelis, kuriame yra naudojamai *Waterfall* ir *Iteratyviojo* modelio elementai. Modelyje dėmesys koncentruojamas į rizikų analizę. Didžiausias sunkumas šiuo modeliu paremtame programų kūrimo procese – teisingai pasirinkti momentą, kai pradedama kita kūrimo proceso spiralė.	• Programos procesas yra tiksliai dokumentuotas, tačiau, atsiradus pokyčiams, dokumentaciją galima pakeisti. • Net vėlesniuose etapuose galima vykdyti projekto pakeitimus ir pridėti naujų funkcijų. • Modelis puikiai tinka dideliems projektams.	• Kūrimo procesas yra ganėtinai brangus. • Programos kūrimo rizikų kontrolei būtina pasirinkti aukštos kvalifikacijos specialistus. • Nepriimtinas mažiems projektams. • Kūrimo modelis yra sudėtingesnis už kitus SDLC modelius. • Projekto kūrimo laiko valdymas sudėtingas, nes spiralių skaičius nežinomas.	• Programos užsakovas neturi aiškių reikalavimų programai. • Projekto eigoje gali iškilti poreikis dideliems pakeitimams. • Projekto eigoje produktas turi būti nuolat vertinamas užsakovo. • Projektas didelis ir ilgalaikis.
V formos modelis (angl. *V-shaped model*)	Išplėstinis *Waterfall* modelis, kuriame taikomi labai griežti reikalavimai, ir prie kiekvieno kito žingsnio pereinama tik pabaigus ankstesnįjį.	• Kiekvienas žingsnis turi aiškius tikėtinus rezultatus, todėl valdyti projektą yra lengva. • Tinka mažiems projektams su aiškiais ir nekintančiais reikalavimais. • Programinės įrangos defektai nustatomi ankstyvoje stadijoje.	• Modelis nelankstus. • Jei projekto eigoje vykdomi pakeitimai, programinės įrangos reikalavimų ir testų dokumentus būtina atnaujinti.	• Tinka projektams, kuriuose būtinas kruopštus testavimas. • Tinka mažiems ir vidutinio dydžio projektams su aiškiais reikalavimais.
Waterfall modelis (angl. *Waterfall model*)	Kaskadinis modelis, kuriame programos kūrimas nuosekliai pereina analizės, projektavimo, realizacijos, testavimo, diegimo ir palaikymo žingsnius. Modelis yra griežtai dokumentuotas.	• Modelis yra paprastas ir lengvai suprantamas. • Tinka mažiems ir vidutinio dydžio projektams, kurių reikalavimai yra aiškūs. • Programos kūrimo užduotis lengva prioretizuoti. • Modelio įgyvendinimo resursai yra minimalūs.	• Programą galima eksploatuoti tik įvykdžius paskutinį kūrimo žingsnį. • Didelė neapibrėžtumo rizika. • Netinka sudėtingiems ir objektiškai orientuotiems projektams. • Netinka ilgalaikio programos kūrimo projektams. • Vieno žingsnio progresą sunku išmatuoti.	• Projektas yra gerai dokumentuotas. • Projekto apibrėžtys yra nekintančios. • Reikalavimai yra vienareikšmiai.

Vieningo SDLC modelio nėra, tačiau kiekvienas modelis turi pagrindines stadijas [78]:

- *Planavimas ir reikalavimų analizė*. Šio etapo tikslas yra apibrėžti reikalavimus sistemai. Jo metu projekto vykdytojai ir užsakovai turi sutarti dėl projekto užduočių.

- *Projekto architektūros dizainas*. Šiame etape projekto vykdytojai turi sukurti projekto architektūrą. Joje apibrėžiamos projekto metu naudojamos technologijos, projekto vykdytojo komandos krūvis, laiko terminai bei projekto biudžetas.
- *Kūrimas ir programavimas*. Aktyvūs programavimo darbai, kurių metu kuriamas algoritmas ir vykdomas kodavimas.
- *Testavimas*. Šiame etape nustatomos ir taisomos algoritmavimo ir kodavimo klaidos bei neatitikimai dokumentacijai. Procesas kartojamas, kol pašalinamos didžiausios problemos.
- *Diegimas*. Programa perduodama užsakovui, su kuriuo aktyviai dirba programos palaikymo komanda. Užsakovas mokomas, konsultuojamas, jam teikiami programos atnaujinimai.

SDLC modelio panaudojimas programinio produkto kūrimo metu leidžia standartizuoti procesą, optimizuoti darbą, suderinti užsakovo ir vykdytojo interesus bei lūkesčius ir išvengti programinės įrangos architektūros kūrimo, programavimo, testavimo, diegimo ir priežiūros klaidų.

Išanalizavus programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, programavimo ir projektavimo teorinius pagrindus galima daryti šiuos apibendrinimus:

- Laiko eilučių duomenų stacionarumo įvertinimas yra svarbus pradinis tyrimo etapas, kuriam naudojama grafinė analizė bei vieneto šaknies testai: ADF, KPSS, ZA. Tiek grafinės analizės įrankis, tiek visi trys testai yra realizuoti šiame magistro darbe suprojektuotoje ir suprogramuotoje programinėje įrangoje.
- Nestacionarių laiko eilučių duomenų prognozavimo tikslumui vertinti naudojamos įvairios metrikos: priklausančios nuo matavimo skalės (MAE, MdAE, RMSE) arba nepriklausančios nuo matavimo skalės (MAPE, sMAPE). Šiame magistro darbe suprojektuotoje ir suprogramuotoje programinėje įrangoje galima pasirinkti vieną iš trijų rekurentinių neuroninių tinklų. Kadangi prognozės tikslumui naudojant skirtingus neuroninius tinklus palyginti nuo matavimo skalės priklausančių metrikų klasei priskiriamos metrikos negali būti naudojamos, o MAPE turi keletą trūkumų, pasirinkta naudoti sMAPE metriką.
- Prognozuojant nestacionarius laiko eilučių duomenis tradiciniais statistiniais metodais jie įprastai apdorojami paverčiant juos stacionariais: pašalinama tendencija, taikomas diferencijavimas arba logaritminės transformacijos. Tačiau tokiu atveju nepagrįstai nuvertinama prognozės neužtikrintumo galimybė bei prarandami kai kurie duomenų nestacionarumo teikiami privalumai. Mokslinės literatūros analizė patvirtina šiame magistro darbe keliamą hipotezę: rekurentinių tinklų galimybė kiekviename laiko žingsnyje grąžinti signalą iš išėjimo sluoksnio į įėjimą daro juos efektyvius prognozuojant nestacionarių laiko eilučių duomenis jų prieš tai nestacionarizavus.

- Mokslinėje literatūroje dažniausiai analizuojami rekurentiniai neuroniniai tinklai, naudojami prognozuojant laiko eilučių duomenis, yra SRNN, LSTM ir GRU. Šie trys neuroniniai tinklai yra realizuoti šiame magistro darbe suprojektuotoje ir suprogramuotoje programinėje įrangoje su galimybe palyginti jų prognozių tikslumą.
- Šiame magistro darbe projektuojant ir programuojant programinę įrangą, skirtą nestacionarių laiko eilučių tyrimui, buvo keliami nuoseklumo, vienareikšmiškumo, apibrėžtumo ir aiškios dokumentacijos reikalavimai, todėl panaudotas *Waterfall* modelis.

II. PROGRAMINĖS ĮRANGOS, SKIRTOS NESTACIONARIŲ LAIKO EILUČIŲ TYRIMUI, PROJEKTAVIMO IR PROGRAMAVIMO REZULTATAI

2.1. Aparatinės ir programinės įrangos charakteristikos

Įgyvendinant šiame magistro darbe iškeltą tikslą buvo naudojamas kompiuteris su 2,60 GHz greičio Intel Core i5-3230M procesoriumi. Kompiuteryje įdiegtos atminties dydis – 12 GB, naudojama Windows 7 64 bitų operacinė sistema.

Taip pat buvo panaudotas atviro kodo *Python* (versija 3.6.10) interpretatorius, įdiegtas panaudojant *Anaconda* (versija 4.5.11) skirstymo paketą.

Anaconda aplinkoje buvo įdiegta atviro kodo mašininio mokymo biblioteka *Tensorflow* (versija 2.1.0), atviro kodo mašininio mokymo biblioteka *Scikit-learn* (versija 0.22.2), atviro kodo programavimo aplinka *Spyder* (versija 4.1.3) ir pagalbinės bibliotekos.

2.2 Programinio produkto planavimas ir reikalavimų analizė

Programines įrangos reikalavimai buvo sukurti pasinaudojant *Volere* šablonu [85]. *Volere* šablonas pasirinktas dėl to, kad leidžia mažiausiomis laiko sąnaudomi ir aiškiausiai apibrėžti bet kurios šiuolaikinės mokslo, verslo ir programinės įrangos sistemų reikalavimus. Teisingai suformuluoti ir išgryninti reikalavimai leidžia aiškiau suprasti problematiką ir sukurti sistemą, kuri geriausiai atitinka naudotojų lūkesčius [86].

1. Projekto paskirtis

1a. Naudotojo problema. Prognozuojant nestacionarias laiko eilutes iškyla tam tinkamos programinės įrangos klausimas. Laiko eilučių duomenims analizuoti yra sukurtos programos, iš kurių populiariausios yra šios:

- *PAST* – programinė įranga, skirta mokslinei duomenų analizei. Turi duomenų apdorojimo, braižymo, statistines, ekologinės analizės, laiko eilučių analizės bei kitas funkcijas [37].
- *GRETL* – atviro kodo ekonometrinei analizei skirta programinė įranga. Suteikia naudotojui galimybę analizuoti įvairiais formatais pateikiamus duomenis. Programinėje įrangoje realizuoti statistiniai laiko eilučių analizės ir prognozavimo metodai [38].

Deja, šios programos neturi galimybės prognozuoti duomenų neuroniniais tinklais.

- *Alyuda Research* programiniai produktai *Alyuda Neurointellicence*, *Alyuda Neurosignal*, *Alyuda Forecaster* skirti tyrėjams, analizuojantiems mokslinius, inžinierinius bei finansinius duomenis [39].
- *Attrasoft Predictor* yra Attrasoft Inc. programinis produktas, pasižymintis galimybe prognozuoti duomenis, kurių apimtis viršija vieną terabaitą [40].
- *NeuroSolutions* yra programinis produktas, leidžiantis naudotojui analizuoti duomenis pasinaudojant vedlio pagalba sumodeliuotais neuroniniais tinklais [41].

Šie programiniai produktai yra komerciniai, išbaigti bei universalūs, tačiau turi daug nebūtinų funkcijų.

- *Python* ir *R* programavimo kalba galima atlikti matematinius, statistinius skaičiavimus bei duomenų prognozavimą arba kurti išbaigtus programinius produktus.
- *MathWorks* komercinis produktas *Matlab*, kurio pagalba galima atlikti laiko duomenų statistinius tyrimus ir prognozavimą.

Tai programavimo kalbos ir programinės bibliotekos, kurių pagalba galima sukurti reikalingą laiko eilučių duomenų prognozavimo programinę įrangą, tačiau tam reikalingos dalykinės kompetencijos, kurias nebūtinai turi duomenis analizuojantys tyrėjai.

Su šiame magistro darbe kuriama programine įranga galima būtų identifikuoti duomenų nestacionarumą ir prognozuoti duomenis tinkamai pasirinktu neuroniniu tinklu. Gebėjimas nustatyti duomenų nestacionarumą ir parinkti tokių duomenų prognozavimui tinkamą neuroninį tinklą yra būtinas siekiant atlikti kuo tikslesnį laiko eilučių duomenų prognozavimą. Todėl tokią laiko eilučių duomenų analizę ir prognozavimą atliekanti programinė įranga yra aktuali.

1b. Projekto tikslai. Nestacionarių laiko eilučių tyrimo įrankis reikalingas tam, kad naudotojas galėtų atlikti laiko eilučių duomenų stacionarumo analizę bei jų prognozavimą rekurentiniais neuroniniais tinklais.

2a. Produkto naudotojas

Prieš pradedant kurti projektą būtina apibrėžti projekto naudotojus. Šio produkto naudotojas gali būti asmuo, siekiantis prognozuoti nestacionarius laiko eilučių duomenis rekurentiniais neuroniniais tinklais. Galimi programos naudotojai:

- Duomenų analitikas;
- Verslininkas;
- Investuojantis privatus asmuo;
- Laiko eilučių duomenų tyrimą atliekantis studentas.

Produkto naudotojui yra reikalingos vidutinio lygio statistinės ir neuroninių tinklų žinios bei mokėjimas dirbti su *Microsoft Access* programa. Naudotojo lytis ir amžius nereglamentuojami,

produktas skirtas lietuvių kalbą mokantiems asmenims, jis nebus pritaikytas negalią turintiems žmonėms.

2b. Naudotojo dalyvavimas. Kuriant programinį produktą būtina pasitelkti potencialius naudotojus, nes neturint atgalinio ryšio iš naudotojo pusės projektas gali žlugti. Šiame projekte naudotojas gali prisidėti prie produkto kūrimo suteikdamas ekonometrinio pobūdžio konsultacijas bei konsultacijas dėl naudotojo sąsajos patogumo. Konsultacijų trukmė – 1-2 valandos per savaitę.

3. Realizavimo aplinka. Programinė įranga turi veikti *Windows 7* arba *Windows 10* operacinėje sistemoje. *Python* kalba parašyta programinė įranga turi veikti *Python* interpretatoriuje.

4. Naudojami programiniai produktai:

- *Miniconda* programinės įrangos diegimo paketas;
- *Microsoft Office 2016 Professional Plus* programinis paketas;
- *Python 3.6.10* programinės kalbos interpretatorius;
- *Python* interpretatoriuje naudojamos programinės bibliotekos:
  - *arch 4.13* statistinių funkcijų programinė biblioteka,
  - *numpy 1.18.4* duomenų masyvų apdorojimo biblioteka,
  - *pandas 1.0.3* duomenų analizės ir valdymo biblioteka
  - *pyodbc 4.0.30* programų biblioteka, skirtą ODBC duomenų bazių pasiekimui,
  - *scikit-learn 0.22.2* mašininio mokymo funkcijų programinė biblioteka,
  - *Spyder 4.1.3* integruota programavimo aplinka
  - *tensorflow 2.1.0* mašininio mokymo programinė biblioteka.

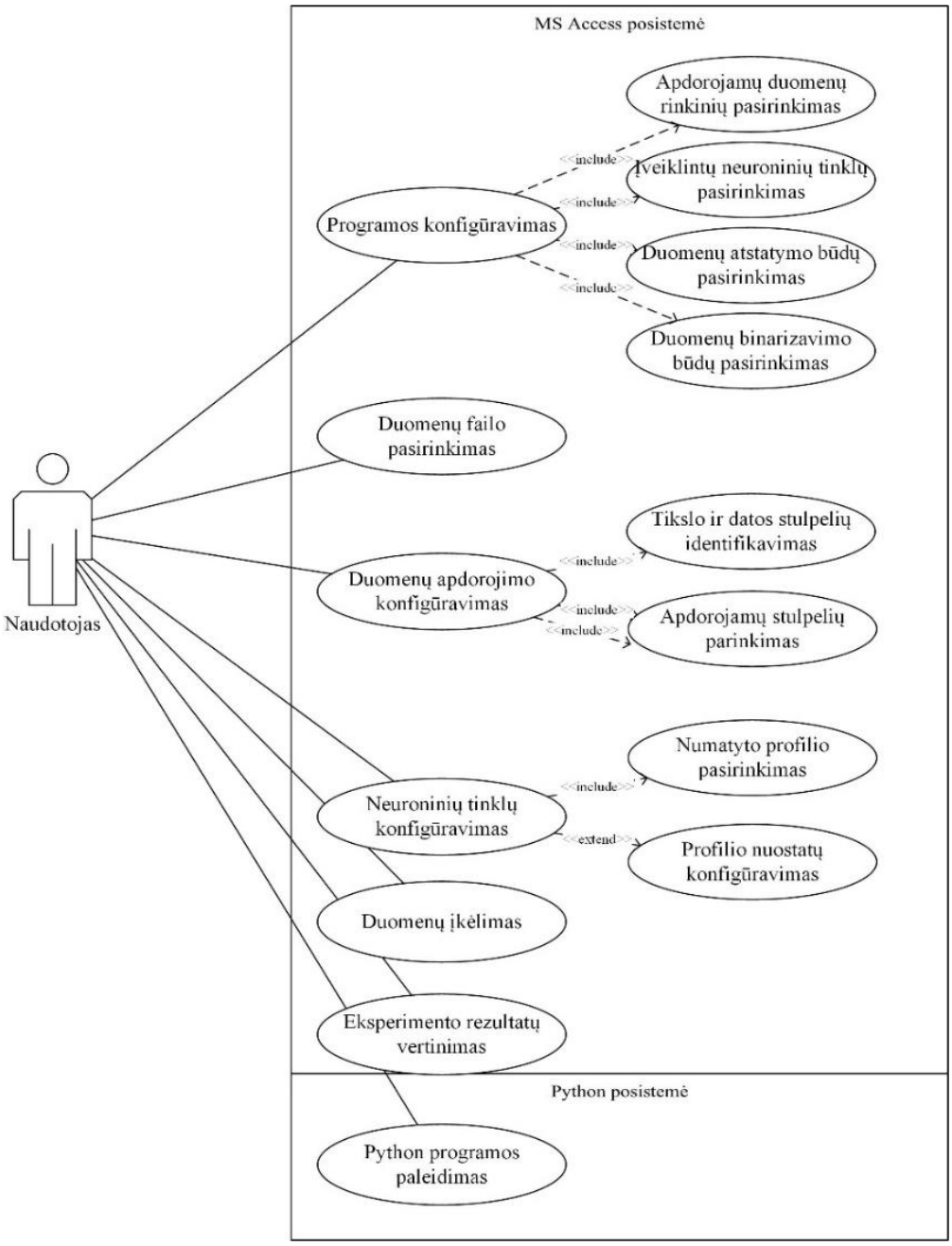
5. Numatoma naudotojo darbo vieta – biuro arba namų kompiuterizuota darbo vieta.

6. Pavadinimai ir apibrėžimai:

- Projektuojama programinė įranga vadinsis „Tyrimai“.
- Programinės įrangos *Microsoft Office* pavadinimas priklauso firmai *Microsoft*.
- Kitų naudojamų programinių ir aparatinių produktų pavadinimai priklauso atitinkamiems tų produktų autorių teisių savininkams.

7. Susiję faktai ir prielaidos. Kuriama programinė įranga „*Tyrimai*“ sudaryta iš *Python* programavimo kalba parašytos dalies, apimančios apie 800 programos eilučių kodo ir komentarų, bei *Microsoft Access* duomenų bazės valdymo sistemoje realizuotos dalies, kuri sudaryta iš duomenų bazės, SQL užklausų, skriptų, formų bei apie 1200 programos eilučių *Visual Basic for Application* programavimo kalbos kodo ir komentarų.

8. Produkto panaudos atvejai. Programinis produktas projektuojamas atsižvelgiant į naudotojo poreikius. Programinio produkto panaudos diagrama pateikiama 14 pav.



14 pav. Programinio produkto panaudos diagrama

8a. Panaudos atvejų sąrašas

Panaudos atvejis 1: programos konfigūravimas.

Aktorius: naudotojas.

Aprašymas: apdorojamų duomenų aibių pasirinkimas. Įveiklintų neuroninių tinklų pasirinkimas. Duomenų atstatymo būdų pasirinkimas. Duomenų binarizavimo būdų pasirinkimas.

Panaudos atvejis 2: duomenų failo pasirinkimas.

Aktorius: naudotojas.

Aprašymas: pasirenkamas duomenų failas, jo parametrai (pavadinimas, vieta diske) išsaugomi Microsoft Access duomenų bazėje.

Panaudos atvejis 3: duomenų apdorojimo konfigūravimas.

Aktorius: naudotojas.

Aprašymas: tikslo ir datos kintamųjų pasirinkimas. Apdorojamų kintamųjų pasirinkimas.

Panaudos atvejis 4: neuroninių tinklų konfigūravimas.

Aktorius: naudotojas.

Aprašymas: numatyto profilio pasirinkimas. Profilio konfigūravimas.

Panaudos atvejis 5: duomenų aibės įkėlimas.

Aktorius: naudotojas.

Aprašymas: pasirinkta duomenų aibė įkeliama į Microsoft Access duomenų bazę.

Panaudos atvejis 6: kompiuterinio eksperimento nuostatų sukūrimas.

Aktorius: naudotojas.

Aprašymas: sukuriamas naujas kompiuterinis eksperimentas, įrašomas jo pavadinimas ir aprašymas.

Panaudos atvejis 7: kompiuterinio eksperimento vykdymas.

Aktorius: naudotojas.

Aprašymas: vykdomas kompiuterinis eksperimentas, kurio metu pagal numatytas nuostatas išanalizuojamas pasirinktų duomenų aibių stacionarumas ir prognozuojami duomenys.

Panaudos atvejis 8: eksperimento rezultatų vertinimas.

Aktorius: naudotojas.

Aprašymas: naudotojas peržiūri ir įvertina kompiuterinio eksperimento rezultatus.

9. Funkciniai reikalavimai:

Funkcinis reikalavimas 1: programinės įrangos nuostatoms ir duomenims saugoti turi būti sukurta *Microsoft Access* duomenų bazė.

Atitikimo kriterijus: *Microsoft Access* duomenų bazė turi būti sukurta *Microsoft Access* 2016 duomenų bazių valdymo sistemoje.

Funkcinis reikalavimas 2: Microsoft Access duomenų bazių valdymo sistemoje turi būti realizuotas duomenų aibių įkėlimas iš tekstinių failų į *Microsoft Access* duomenų bazę.

Atitikimo kriterijus: failuose saugomi duomenys turi turėti kintamųjų antraštes, o kintamieji turi būti atskirti „tab“ simboliu arba kableliu. Faile saugomos duomenų aibės kintamųjų skaičius neturi viršyti 32, o įrašų skaičius neturi būti didesnis kaip 5000.

Paaiškinimas: Duomenų aibės kintamųjų ir įrašų skaičiaus apribojimai įvesti, nes empiriniu būdu buvo nustatyta, kad programinėje įrangoje naudojamos duomenų stacionarumą nustatančios funkcijos intensyviai naudoja kompiuterinius resursus, ir didesnių duomenų apimčių panaudojimas gali sutrikdyti programos darbą.

Funkcinis reikalavimas 3: Microsoft Access duomenų bazių valdymo sistemos pagalba turi būti įmanoma sukonfigūruoti *Python* kalba parašytos programinės dalies veikimo nuostatas, duomenų aibių paruošimo nuostatas bei neuroninių tinklų nuostatas.

Atitikimo kriterijus: *Microsoft Access* duomenų bazėje turi būti išsaugotos programos nuostatos, leidžiančios abejoms programos dalims sąveikauti.

Funkcinis reikalavimas 4: Python programinės dalies pagalba turi būti įmanomas pasirinktų duomenų aibių paruošimas, jų apdorojimas neuroniniais tinklais bei eksperimento rezultatų įrašymas į *Microsoft Access* duomenų bazę tolesniam įvertinimui.

Atitikimo kriterijus: programos direktorijų medis turi atitikti nustatytą šabloną.

Funkcinis reikalavimas 5: Microsoft Access duomenų bazė turi būti padalinta į dvi dalis – priekinę ir galinę (angl. *Front end, Back end*).

Atitikimo kriterijus: abi duomenų bazės dalys turi būti saugomos vienoje direktorijoje.

Paaiškinimas: priekinėje duomenų bazės dalyje turi būti saugomos nuorodos į galinės duomenų bazės lenteles, SQL užklausos, formos, programiniai skriptai ir moduliai. Galinėje duomenų bazės dalyje turi būti saugomos tyrimo projekto nuostatų, programos nuostatų, duomenų paruošimo nuostatų duomenis saugančios lentelės, pagalbinius duomenis saugančios lentelės bei įkeltus duomenis saugančios lentelės. Toks duomenų bazės turinio atskyrimas pasirinktas siekiant užtikrinti duomenų bazės saugumą, našumą bei patikimumą [87].

Funkcinis reikalavimas 7: nuostatų saugojimui skirtų lentelių pavadinimai turi prasidėti priešdėliu „*tbl*“, duomenų aibėms saugoti skirtų lentelių pavadinimai – priešdėliu „*_tbl*“.

Funkcinis reikalavimas 8: duomenų bazių valdymo sistemoje esantis programinis kodas turį būti realizuotas formų programiniuose moduliuose bei modulyje „*Helper*“.

Funkcinis reikalavimas 9: Microsoft Access formoje „*Programos nuostatos*“ turi būti įmanoma sukonfigūruoti šias nuostatas:

- Įvesti ir išsaugoti *Miniconda* virtualios aplinkos vietą diske;
- Įvesti ir išsaugoti Miniconda programinio paketo diegimo vietą diske;
- Parinkti mokymo ir testavimo duomenų apimčių santykį;

- Įjungti ar išjungti Tensorflow bibliotekos vidinį atsitiktinių skaičių generatorių kartu su Python vykdymo terpės atsitiktinių skaičių generatoriumi. Atsitiktinių skaičių išjungimas būtinas eksperimento pakartojamumui užtikrinti;
- Pasirinkti tyrime naudojamus neuroninius tinklus;
- Pasirinkti tyrime naudojamas duomenų aibes;
- Parinkti trūkstamų duomenų aibių reikšmių atstatymo metodus pagal duomenų tipą;
- Parinkti duomenų binarizavimo metodus pagal duomenų tipą. Kadangi duomenų aibės duomenų binarizavimas dažnai atliekamas dviem metodais, kurie vykdomi nuosekliai vienas po kito, konfigūracijoje numatytas pirminio ir antrinio duomenų binarizavimo nuostatų pasirinkimas;
- Parinkti naudojamas stacionarumo nustatymo funkcijas.

Funkcinis reikalavimas 10: *Microsoft Access* formoje *„Duomenų nuostatos“* turi būti įmanoma sukonfigūruoti šias nuostatas:

- Įkelti pasirinkto tekstinio duomenų failo pavadinimą ir atributų sąrašą;
- Keisti kiekvieno atributo duomenų tipą;
- Peržiūrėti į *Microsoft Access* duomenų bazę įkeltų duomenų lentelės atributų pavadinimus;
- Išjungti atributo duomenų panaudojimą eksperimentuose;
- Įvertinti atribute esančių kategorinių, skaitinių, datos ir tuščių reikšmių kiekius skaitine ir procentine išraiška;
- Peržiūrėti kiekvieno atributo duomenų pirmas 10 eilučių;
- Galimybė po *Python* kalba parašytos programos įvykdymo formoje *Duomenų nuostatos* matyti kiekvieno atributo stacionarumo požymį.

Funkcinis reikalavimas 11: *Microsoft Access* formoje *„Eksperimento nuostatos*“ turi būti įmanoma sukonfigūruoti šias nuostatas:

- Sukurti naują eksperimentą;
- Įvesti eksperimento pavadinimą;
- Įvesti eksperimento aprašymą;
- Įvykdyti pasirinktą eksperimentą;
- Peržiūrėti reikšmingiausius eksperimento rezultatus ir duomenų vizualizacijas.
- Paleisti Python programavimo kalba parašytą praogramą.

Funkcinis reikalavimas 12: kiekvienas iš šių programinėje įrangoje naudojamų neuroninių tinklų *Microsoft Access* duomenų bazės valdymo sistemoje turi turėti tam tinklui konfigūruoti skirtą naudotojo sąsają, sudarytą iš Microsoft Access duomenų formos ir joje esančių valdiklių.

Funkcinis reikalavimas 13: *Microsoft Access* formose, kuriose bus galima konfigūruoti neuroninius tinklus, turi būti įmanoma sukonfigūruoti šias nuostatas:

- Sukurti naują profilį;
- Įvesti profilio pavadinimą;
- Įvesti profilio aprašymą;
- Pasirinkti numatytąjį profilį;
- Nustatyti pasirinkto neuroninio tinklo nuostatas.

10. Nefunkciniai reikalavimai

10a. Naudotojo sąsaja. Programos naudotojo sąsaja turi būti paprasta ir funkcionali, sukurta standartinių *Microsoft Acces* formų pagalba. Formos turi būti atidaromos įrankių juostoje patalpintų valdiklių pagalba. Vykdant kompiuterinį eksperimentą, turi būti matomas *Python* terpės langas, kurio uždarymas pasibaigus eksperimentui turi būti patvirtinamas „*Enter*“ klavišo nuspaudimu.

10b. Programinio produkto įdiegimo reikalavimai. Programinis produktas naudotojui turi būti pateikiamas viename *ZIP* faile „*tyrimai.zip*“. Išskleidus archyvą numatytoje disko vietoje, turi būti sukuriama programos veikimui reikalinga direktorijų struktūra. Programos veikimui būtina papildoma programinė įranga turi būti parsiųsta iš tos programinės įrangos gamintojų svetainių ir įdiegta atskirai.

11. Rizikos. Kiekvienas projektas gali turėti rizikas, todėl jos turi būti suformuluotos ir įvertintos. Šiame projekte galima įžvelgti šias rizikas:

Rizika 1: programavimo klaida

Aprašymas: kiekviename projekte yra programavimo klaidų. Kai kurios gali būti sunkiai pastebimos, nes pasireiškia retai pasitaikančiose situacijose. Kai kurios programavimo klaidos gali iškraipyti programos veiklos rezultatus, kitos – sustabdyti programos veikimą arba net negrįžtamai sugadinti programinę įrangą.

Būdas išvengti: programavimo klaidos taisomos programinės įrangos testavimo metu arba ją eksploatuojant. Siekiant sparčiai pašalinti programavimo klaidas, programos kūrėjui būtina turėti tamprų ryšį su programos naudotoju.

Rizika 2: algoritmo klaida

Aprašymas: algoritmo klaidos padaromos programos projektavimo metu ir dažniausiai yra taisomos sunkiau negu programavimo klaidos.

Būdas išvengti: kaip ir programavimo klaidos, algoritmo klaidos išryškėja ir yra taisomos programos testavimo ir naudojimo metu. Kruopštus programinės įrangos testavimas mažina tikimybę algoritmo klaidai išryškėti programos eksploatavimo metu.

Rizika 3: duomenų bazės gedimas

Aprašymas: neteisingas programinis kodas, duomenų bazės perkrova, kompiuterio operatyvios atminties trūkumas ir kitos priežastys gali lemti *Microsoft Access* duomenų bazės gedimą. Jis gali pasireikšti programos sulėtėjimu, nuolatiniais klaidos pranešimais ar kitais požymiais.

Būdas išvengti: turi būti reguliariai atliekama duomenų bazės profilaktika *Microsoft Access* duomenų bazės valdymo sistemos priemonėmis, reguliariai diegiami *Microsoft Office* programinės įrangos atnaujinimai. Reguliarus duomenų bazės archyvinis kopijavimas leidžia išvengti nemalonių duomenų bazės gedimo pasekmių [88].

2.3 Projekto architektūros dizainas

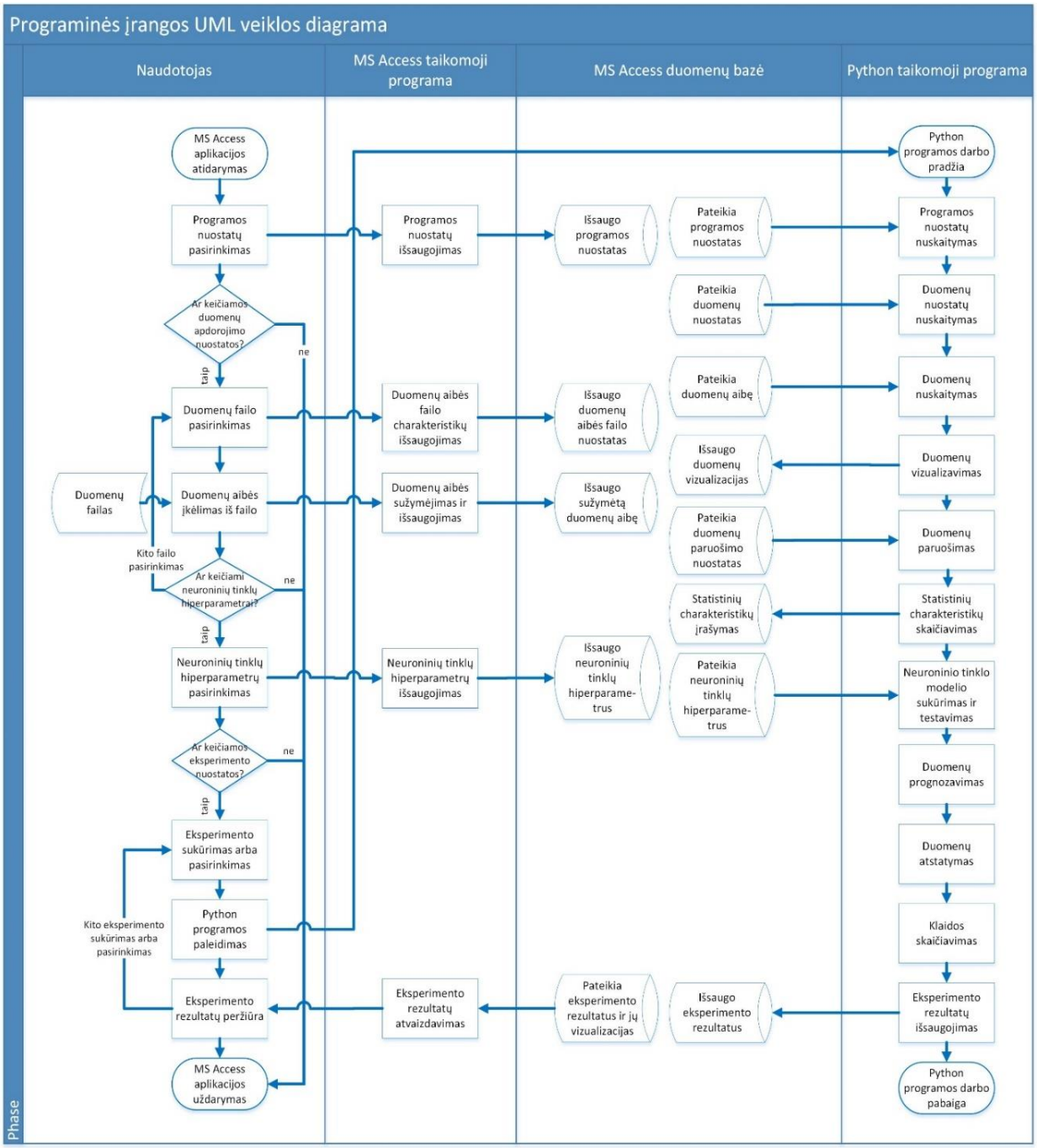
2.3.1 Programos komponenčių tarpusavio sąveika

Programinis produktas sudarytas iš dviejų dalių: 1) *Microsoft Access 2016* duomenų bazių valdymo sistemos pagrindu suformuotos sistemos su *Visual Basic for Application (VBA)* kodu, ir 2) *Python* programavimo kalba parašytos programos. *MS Access* duomenų bazės valdymo sistema pasirinkta dėl to, kad turi tamprias sąsajas su kitomis *Microsoft Office* programomis, yra lengvai valdoma. Pagrindinius duomenų apdorojimus vykdančiai programinės įrangos realizavimui pasirinkta *Python* programavimo kalba, kadangi ji optimizuota duomenų valdymui, yra lengvai įsisavinama, ir turi programines bibliotekas statistiniam duomenų apdorojimui. Taip pat *Python* kalba yra realizuotos šiuo metu efektyviausios neuroninių tinklų bibliotekos *Tensorflow* ir *Keras*.

Programinės įrangos projektavimui ir programavimui buvo pasirinktas *Waterfall* metodas. Tokį pasirinkimą lėmė tai, jog projekto apimtis yra nedidelė, reikalavimai aiškūs ir vienas kitam neprieštarauja, projektuoja ir programuoja vienas žmogus.

Projektuojama programa yra dinaminė sistema, kurią yra patogu pavaizduoti veiklos diagrama (15 pav.). Šioje diagramoje matomas veiklos pasidalinimas tarp programinės įrangos objektų. Objektai yra keturi: 1) Naudotojas, 2) *Microsoft Access* duomenų valdymo sistemoje realizuota taikomoji programa, 3) *Microsoft Access* duomenų bazė ir 4) *Python* taikomoji programa. Kiekvieno iš šių objektų vykdomų veiksmų ir ryšių vaizdavimui numatytas takelis. Programos darbą inicijuoja naudotojas, atidarydamas *Microsoft Access* aplikaciją. Jos pagalba naudotojas pasirenka duomenų aibę, į *Microsoft Access* duomenų bazę įkelia jo atributų sąrašą, ir šio duomenų aibės duomenis įrašo į *Microsoft Access* duomenų bazę, tuos veiksmus galimai pakartodamas. Naudotojas *Microsoft*

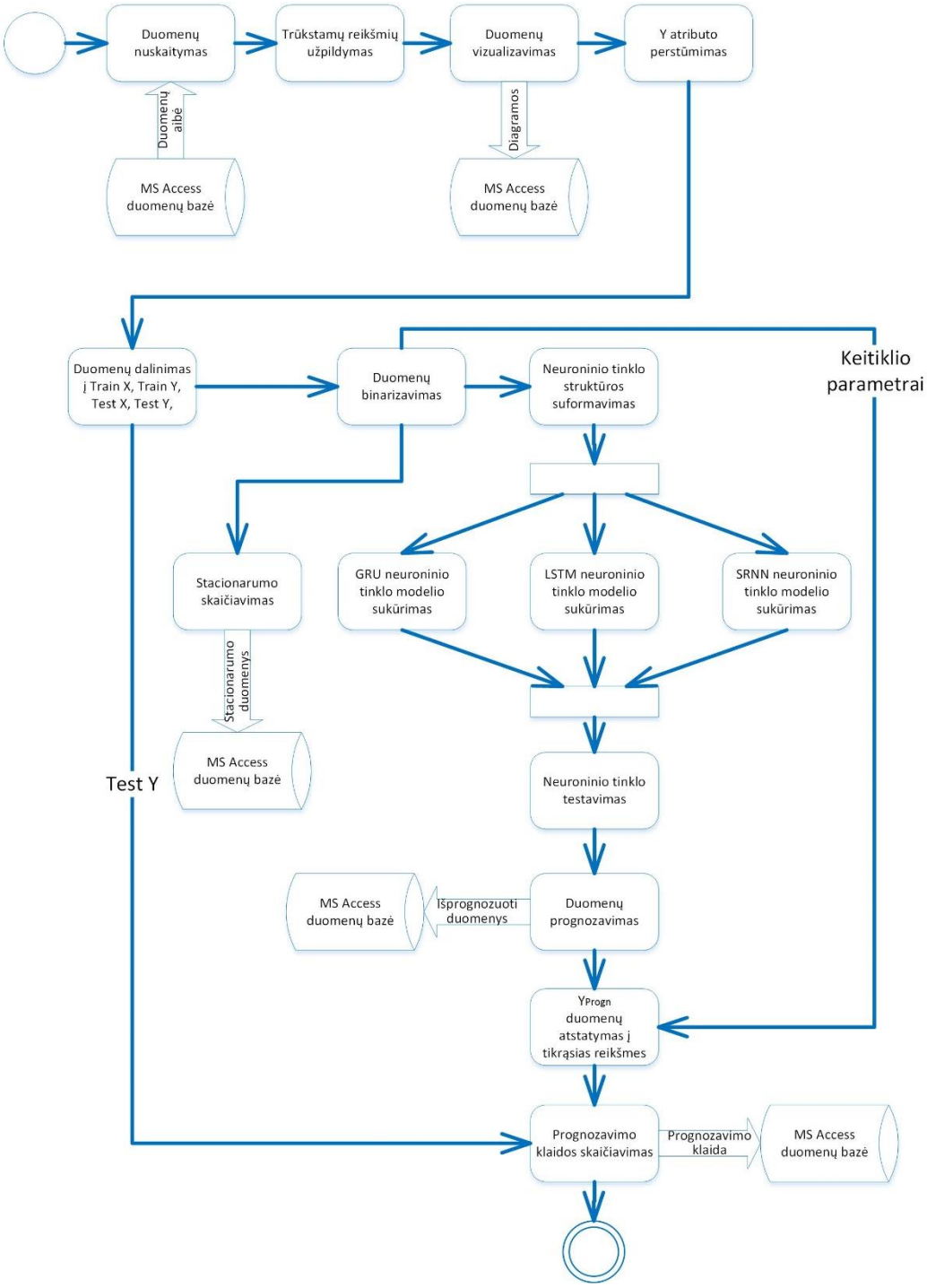
Access aplikacijos pagalba taip pat gali parinkti programos bei naudojamų neuroninių tinklų nuostatas.



15 pav. Programinės įrangos UML veiklos diagrama

Po to naudotojas turi galimybę paleisti *Python* taikomąją programą, kuri iš duomenų bazės įkelia neuroninių tinklų hiperparametrus, taip pat duomenų aibės duomenis bei nuostatas, reguliuojančias tų duomenų apdorojimą. Programa atstato trūkstamas duomenų aibės reikšmes, vizualizuoja bei binarizuoja duomenų aibės duomenis, skaičiuoja statistines charakteristikas, transformuoja duomenis į formatą, tinkamą pateikti į neuroninio tinklo įėjimą, sukuria neuroninio tinklo modelį ir jam pateikia duomenis. Po to, kai neuroninis tinklas atlieka duomenų prognozę, jos

rezultatai lyginami su testiniais duomenimis, ir apskaičiuojama klaidos reikšmė, kuri tolesnei jos analizei ir vertinimui yra įrašoma į *Microsoft Access* duomenų bazės lentelę.*Python* taikomosios programos veiklos diagrama pateikiama 16 pav.



16 pav. Python programavimo kalba parašytos programos UML veiklos diagrama

Programos veikla prasideda duomenų nuskaitymu iš *Microsoft Access* duomenų bazės. Toliau iš duomenų atributų išskiriama *Test y* duomenų komponentė, kuri bus panaudota neuroninio tinklo prognozavimo klaidos skaičiavimui. Kituose žingsniuose, pasinaudojant *Microsoft Access* duomenų bazėje saugomomis programos nuostatomis, yra atstatomos duomenų aibės duomenų trūkstamos

reikšmės ir duomenų aibės kintamieji yra vizualizuojami. Tolesni vykdomas duomenų aibės binarizavimas, panaudojant *Microsoft Access* duomenų bazėje išsaugotas binarizavimo nuostatas.Kiekvienam binarizuoto duomenų aibės atributui, panaudojant programinės bibliotekos ARCH procedūras, atliekami ADF, KPSS ir ZA testai. Loginė testo rezultato reikšmė įrašoma į *Microsoft Access* duomenų bazę.

2.3.2 *Microsoft Access* duomenų bazės struktūra

Microsoft Access duomenų bazės pagrindą sudaro trys nuostatoms saugoti skirtų lentelių rinkiniai, kuriuose lentelės susietos tarpusavio ryšiais (3 lentelė).

3 lentelė. Microsoft Access duomenų bazės lentelių rinkiniai

Eil. Nr.	Lentelių rinkinio paskirtis	Lentelės pavadinimas	Lentelės paskirtis	Schema
1.	Saugoma su programos nuostatomis susijusi informacija	*tblProgram_Settings*	Saugomos pagrindinės programos nuostatos: mokymo ir testavimo duomenų santykis, atsitiktinių skaičių generatoriaus įjungimo ar išjungimo loginė reikšmė, kategorinių, skaitinių, datos ir tuščių reikšmių atstatymo bei pirminio ir antrinio binarizavimo nuostatos.	1 priedas
		tblImputation_Methods	Saugomos duomenų kategorijos ir kiekvienai iš tų kategorijų tuščių reikšmių atstatymui taikomi metodai.	
		tblNormalize_Methods	Saugomos duomenų kategorijos bei kiekvienai iš tų kategorijų binarizuoti taikomi metodai.	
2.	Saugoma su duomenų aibių nuostatomis susijusi informacija	*tblDataSettings*	Saugomos pagrindinės duomenų aibės nuostatos: duomenų aibės pavadinimas, sudarytas iš duomenų aibės failo pavadinimo ir duomenų bazėje saugomo jo nuostatų įrašo indekso; nuoroda į duomenų aibės failo vietą diske; duomenų aibės *Y* atributo pavadinimas ir jo duomenų tipas; duomenų aibės datos atributo pavadinimas ir jo duomenų tipas; faile saugomų duomenų atributus skiriančio simbolio *ASCII* kodo indeksas; *Microsoft Access* duomenų bazėje programiniu būdu sukurtos duomenų aibei saugoti skirtos lentelės pavadinimas.	2 priedas
		tblSelected_Files	Saugoma formoje „*Programos nuostatos*“ eksperimentui pasirinktų duomenų aibių informacija: duomenų aibės identifikatorius, duomenų aibės pavadinimas, duomenų aibės failo vieta diske bei *Microsoft Access* duomenų bazėje programiniu būdu sukurtos duomenų aibei saugoti skirtos lentelės pavadinimas.	

3 lentelė (tęsinys). Microsoft Access duomenų bazės lentelių rinkiniai

Eil. Nr.	Lentelių rinkinio paskirtis	Lentelės pavadinimas	Lentelės paskirtis	Schema
	Saugoma su duomenų aibių nuostatomis susijusi informacija	*tblSeparators*	Saugomas faile saugomų duomenų atributus skiriančio simbolio *ASCII* kodas bei šio kodo pavadinimas.	2 priedas
		tblColumn_Names	Saugoma su duomenų aibės atributais susijusi informacija: duomenų aibės lentelės indeksas; atributo pavadinimas duomenų aibės faile; atributo pavadinimas; į *Microsoft Access* duomenų bazę įkeltos duomenų aibės lentelės atributo pavadinimas; atributo naudojimo tyrime požymis; atributo duomenų tipas; skaitinių, kategorinių, datos ir tuščių reikšmių skaitinės ir procentinės dalys nuo visų atributo reikšmių skaičiaus.	
3.	Saugoma su neuroninių tinklų nuostatomis susijusi informacija	*tbl_Parameters_SRNN*	Saugomos atitinkamai LSTM, SRNN ir GRU neuroninių tinklų profilių nuostatos. Profilių pagalba atsiranda galimybė išsaugoti daugiau negu vieną neuroninio tinklo nuostatų kombinaciją, viena iš kurių pažymima kaip numatytoji. *Python* terpėje veikianti programa iš duomenų bazės paima būtent numatytame profilyje išsaugotas neuroninio tinklo nuostatas. Kiekvienas profilis lentelėje turi naudotojo suteiktą profilio pavadinimą, aprašymą bei numatyto profilio požymį. Lentelėje taip pat saugomos atitinkamo neuroninio tinklo nuostatos.	3 priedas
		tbl_Parameters_GRU		
		tbl_Parameters_LSTM		
		tblActivations	Saugomi neuroninių tinklų aktyvavimo funkcijų pavadinimai.	
		tblLoss_Functions	Saugomi neuroninių tinklų praradimų funkcijų pavadinimai.	
		tblInitializers	Saugomi neuroninių tinklų inicializatorių pavadinimai.	

2.3.3 *Python* programavimo kalba parašytos programinės įrangos klasių struktūra

Microsoft Access duomenų bazių valdymo sistemoje *VBA* programavimo kalba parašytas kodas yra modulinės struktūros. Tuo tarpu *Python* taikomosios programos kodo struktūra yra objektinė. Iš viso *Python* taikomosios programos kode yra realizuota 14 klasių (4 lentelė).

4 lentelė. Phyton taikomosios programos klasės

Eil. Nr.	Paskirtis	Klasės pavadinimas	Kas realizuota	UML diagrama
1.	Duomenų nuskaitymas iš disko	*cls-Selected-Models*	Pasirinktų neuroninių tinklų sąrašo įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	4 priedas
		cls_Selected_Datasets	Eksperimentui pasirinktų duomenų aibių sąrašo įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti	
		cls_Data_Settings	Duomenų aibių paruošimo nuostatų įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	
		cls_Program_Settings	*Python* kalba parašytos programos nuostatų įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	
		cls_GRU_Settings	GRU nuostatų įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	
		cls_LSTM_Settings	LSTM nuostatų įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	
		cls_SRNN_Settings	SRNN nuostatų įkėlimas iš duomenų bazės ir klasės sąsaja jam pasiekti.	
		cls_Connection_Strings	Tai bazinė klasė visoms aukščiau išvardintoms klasėms. Joje realizuotas prisijungimo prie *Microsoft Access* duomenų bazės eilutės saugojimas	
2.	Neuroninių tinklų klaidų metrikos, duomenų aibės ir neuroniniai tinklai	*cls_Metrics*	Eksperimento rezultatų vertinimui galimų panaudoti metrikų skaičiavimas ir apskaičiuotų klaidų reikšmių įrašymas į *Microsoft Access* duomenų bazę.	5 priedas
		cls_Dataset	Duomenų aibės išankstinio paruošimo metodai.	
		cls_Network_Base	Tai klasės *cls_Network* bazinė klasė, kurioje yra realizuotas neuroninių tinklų modelio sukūrimas ir testavimas.	
		cls_Connection_String	Tai klasės *cls_Dataset* bazinė klasė. Joje realizuotas *Microsoft Access* duomenų bazės prisijungimo eilutės saugojimas.	
		cls_Stationarity_Tests	Tai klasės *cls_Dataset* bazinė klasė. Joje yra realizuoti *ADF, KPSS ir ZA* stacionarumo testų metodai, taikomi duomenų aibės atributams, o taip pat metodas, į *Microsoft Access* duomenų bazę įrašantis tų testų rezultatus.	

2.3.4 Programinės įrangos naudotojo sąsaja

Naudotojo sąsaja suformuota *Microsoft Access* duomenų bazės valdymo sistemos priemonėmis. Ji sudaryta iš penkių formų ir įrankių juostos.

Įrankių juosta „*Tyrimai*“. Įrankių juostos vaizdas pateikiamas 17 pav. Įrankių juosta sudaryta iš 6 elementų. Kiekvienas valdiklis atveria atitinkamą formą (valdiklio „*Tyrimo vykdymas*“. nuspaudimas atveria formą „*Eksperimentai*“).

17 pav. Įrankių juostos valdiklių vaizdas

Kiekviena iš formų turi valdiklius ir/arba tekstinius laukus. Jų funkcijos pateikiamos 5 lentelėje.

5 lentelė. Naudotojo sąsajos formų valdikliai ir jų funkcijos

Eil. Nr.	Forma	Valdiklis / tekstinis laukelis	Funkcijos	Vaizdas
1.	Programos nuostatos	*Atnaujinti įrašą*	Atnaujina formos elementus, priskirdamas jiems *Microsoft Access* duomenų bazėje saugomas nuostatas.	6 priedas
		Išsaugoti įrašą	Formoje įvykdytus nuostatų pakeitimus įrašo į *Microsoft Access* duomenų bazę.	
		Python profilio direktorija	Leidžia naudotojui išsaugoti jo sukurto *Python* profilio (angl. *Environment*) vietą diske.	
		Miniconda diegimo direktorija	Leidžia naudotojui išsaugoti *Python* programinių bibliotekų platinimo paketo *Miniconda* įdiegimo vietą kompiuterio diske.	
		Reikšmių atstatymas	Leidžia naudotojui kiekvienam duomenų tipui parinkti trūkstamų duomenų atstatymo metodą.	
		Reikšmių binarizavimas	Leidžia kiekvienam duomenų tipui parinkti savą binarizavimo metodą. Kadangi duomenys binarizuojami dviem nuosekliai vykdomais etapais (pirminiu ir antriniu), kiekvienam etapui galima parinkti savas nuostatas.	
		Mokymo / testavimo duomenų kiekio santykis	Leidžia naudotojui išsaugoti mokymo ir testavimo duomenų kiekio santykį.	
		Įjungti / išjungti atsitiktinių skaičių generatorių	Leidžia įjungti ir (tyrimo pakartojamumui užtikrinti) išjungti programos naudojamus atsitiktinių skaičių generatorius.	

5 lentelė (tęsinys). Naudotojo sąsajos formų valdikliai ir jų funkcijos

Eil. Nr.	Forma	Valdiklis / tekstinis laukelis	Funkcijos	Vaizdas
1.	Programos nuostatos	*Naudojami neuroniniai tinklai*	Leidžia pasirinkti neuroninį tinklą (-us). Kai nepasirinktas nei vienas tinklas, tyrimo metu atliekami tik duomenų stacionarumo testai.	6 priedas
		Naudojamos duomenų aibės	Leidžia pasirinkti tyrime naudojamą duomenų aibę (-es).	
2.	Duomenų aibių nuostatos	*Pasirinkti duomenų aibę*	Leidžia naudotojui pasirinkti duomenų aibės failą ir jo aprašą įkelti į *Microsoft Access* duomenų bazę.	7 priedas
		Įkelti duomenų aibės atributus	Leidžia į vidinę formą įkelti duomenų aibės atributų parametrus.	
		Išsaugoti nuostatas	Leidžia naudotojui *Microsoft Access* duomenų bazėje išsaugoti formos valdikliais pasirinktas duomenų aibės nuostatas.	
		Importuoti failo duomenis	Leidžia iš duomenų aibės failo importuoti duomenis į *Microsoft Access* duomenų bazę. Importavimo metu duomenų atributai pervadinami, o netinkamos reikšmės pažymimos.	
		Duomenų aibės indeksas	Leidžia matyti unikalų duomenų aibės indeksą duomenų bazėje. Jis sudaromas iš failo pavadinimo ir duomenų lentelės *ID* lauko reikšmės.	
		Failo vieta diske	Leidžia matyti duomenų aibės failo vietą diske.	
		Atributų skirtukas	Leidžia pasirinkti *ASCII* simbolį, kuriuo duomenų faile atskiriami atributai.	
		Tikslo atributo pavadinimas	Leidžia naudotojui išskirti tikslo ir datos atributus bei matyti jų duomenų tipus. Išskyrimas reikalingas tam, kad likusius duomenų aibės atributus jų importavimo metu būtų galima pervadinti pavadinimais „*Var01*“ – „*VarNN*“, kur *NN* yra įgalintų duomenų atributų skaičius.	
		Tikslo atributo duomenų tipas		
		Datos atributo pavadinimas		
		Datos atributo duomenų tipas		
		Duomenų bazės lentelės pavadinimas	Leidžia peržiūrėti iš failo į *Microsoft Access* duomenų bazę importuotų duomenų lentelės pavadinimą. Lentelės pavadinimas sukuriamas iš simbolio „_“ bei duomenų aibės informaciją saugančios duomenų bazės lentelės *tblDataSettings* lauko *ID* reikšmės.	
		Atributai	Leidžia naudotojui po to, kai jis pasirenka duomenų skirtuką ir duomenų failą, matyti failo duomenų atributų pavadinimus, ir įvertinti, ar pasirinktas atributų skirtukas yra teisingas. Tokiu atveju, kai valdiklyje matomi atributų pavadinimai išdėstyti ne vertikaliai, o horizontaliai, naudotojas turi pasirinkti kitą atributų skirtuką ir pakartoti failo pasirikimą.	

5 lentelė (tęsinys). Naudotojo sąsajos formų valdikliai ir jų funkcijos

Eil. Nr.	Forma	Valdiklis / tekstinis laukelis	Funkcijos	Vaizdas
2a.	Duomenų aibės atributai (formos „Duomenų aibių nuostatos“ vidinė forma)	*Atributo pavadinimas*	Pateikia failo duomenų aibės atributo pavadinimą.	7 priedas
		Duomenų bazės lentelės atributo pavadinimas	Leidžia matyti į duomenų bazę importuotų duomenų lentelės atributo pavadinimą.	
		Duomenų tipas	Leidžia naudotojui matyti ir keisti atributo duomenų tipą.	
		Įgalintas	Leidžia naudotojui įgalinti arba išjungti duomenų atributą iš tolesnio apdorojimo.	
		Skaitinių reikšmių skaičius/dalis	Leidžia naudotojui matyti atribute esančių atitinkamai skaitinių, kategorinių, datų bei tuščių reikšmių skaičių bei procentinę dalį nuo visų atributų reikšmių skaičiaus.	
		Kategorinių reikšmių skaičius/dalis		
		Datų reikšmių skaičius/dalis		
		Tuščių reikšmių skaičius/dalis		
		Stacionarus	Leidžia matyti žymėjimą, reiškiantį, kad atributas yra stacionarus. Žymėjimas matomas tik po įvykdytos *Python* taikomosios programos procedūros, apskaičiuojančios atributų duomenų stacionarumą.	
		10 pirmų eilučių	Leidžia naudotojui, pasirinkus kurį nors atributą, matyti to atributo pirmas 10 reikšmių.	
			Valdikliai leidžia formoje pasirinkti pirmą, ankstesnę, kitą arba paskutinę duomenų aibę.	
3.	LSTM nuostatos GRU nuostatos SRNN nuostatos	*Išsaugoti nuostatas*	Leidžia duomenų bazėje išsaugoti formoje padarytus pakeitimus.	8 priedas 9 priedas 10 priedas
		Ištrinti profilį	Leidžia pašalinti profilį iš *Microsoft Access* duomenų bazės.	
		Profilio pavadinimas	Leidžia įvesti profilio pavadinimą.	
		Profilio aprašymas	Leidžia trumpai (iki 255 raidžių) apibūdinti profilį.	
		Numatytasis profilis	Leidžia pažymėti profilį kaip numatytąjį. *Python* taikomoji programa, vykdydama duomenų nuskaitymą iš duomenų bazės, naudoja tik profilio, pažymėto kaip numatytasis, duomenis. Profiliais paremtas neuroninių tinklų nuostatų organizavimas leidžia naudotojui greitai pasirinkti tyrime naudojamą neuroninio tinklo nuostatų aibę.	
		Hiperparametrų valdikliai	Leidžia parinkti neuroninio tinklo hiperparametrus.	

2.4 Programinės įrangos kodavimas

Programinės įrangos kodas pateikiamas 12 priede. Siekiant palengvinti programos skaitymą, koduojant buvo laikomasi šių nuostatų:

- Kintamųjų pavadinimai sudaromi iš priešdėlio ir vardo. Priešdėlis atspindi kintamojo tipą, vardas – kintamojo paskirtį.
- Jei kintamojo varde yra naudojami keli žodžiai, jie pagal galimybę vienas nuo kito atskiriami simboliu „_“.
- *Python* kode naudojamų klasių pavadinimai sudaryti iš priešdėlio „*cls*“ ir vardo, kurie vienas nuo kito atskiriami simboliu „_“.

Python programavimo kalba parašyta programos dalis sudaryta iš 6 failų, kuriuose saugomo programinio kodo paskirtis pateikiama 6 lentelėje.

6 lentelė. Python programavimo kalba parašytos programos modulių charakteristikos

Failo vieta diske ir pavadinimas	Failo paskirtis
C:\Tyrimai\tyrimai.accdb	*Microsoft Access* duomenų bazės priekinė dalis
C:\Tyrimai\tyrimai_be.accdb	*Microsoft Access* duomenų bazės galinė dalis
C:\Tyrimai\python\config.py	*Python* programinis modulis, kuriame saugomos įvairių rūšių duomenis iš duomenų bazės įkeliančios klasės.
C:\Tyrimai\ python\dataset.py	*Python* programinis modulis, kuriame esančios klasės paruošia duomenis prognozavimui.
C:\Tyrimai\ python\export.py	*Python* programinis modulis, kuriame easenčio kodo pagalba realizuojamas duomenų eksportavimas į diską.
C:\Tyrimai\python\main_program.py	Pagrindinis *Python* programinis modulis.
C:\Tyrimai\ python\metrics.py	*Python* programinis modulis, kuriame saugoma klasė, skaičiuojanti duomenų prognozavimo klaidas.
C:\Tyrimai\ python\network.py	*Python* programinis modulis, kuriame saugomos klasės, atsakingos už duomenų apdorojimą neuroniniais tinklais.
C:\Tyrimai\python\stationarity_tests.py	*Python* programinis modulis, būtinas už duomenų stacionarumo vertinimą atsakingų klasių saugojimui.
C:\Tyrimai\python\vizualization.py	*Python* programinis modulis, būtinas duomenų vizualizacijai realizuoti.
C:\Tyrimai\python\data	50 tekstinių duomenų failų.
C:\Tyrimai\python|doc	Su programine įranga susiję dokumentai

2.5 Programinės įrangos diegimas

Programa naudotojui pateikiama kaip nuoroda į *Zip* archyvo failą, patalpintą *Google Disk* failų saugykloje. Programos įdiegimą rekomenduojama atlikti šia tvarka:

- Parsiunčiamas ir įdiegiamas *Python* programinių bibliotekų platinimo paketas *Miniconda* (versija 4.8.2) [89]. Paketo diegimo metu naudotojas turi pasirinkti diegti paketą visiems lokalaus kompiuterio naudotojams, į direktoriją „*C:\ProgramData*“. Tam gali prireikti lokalaus kompiuterio administratoriaus teisių.
- Sukuriama *Conda* virtuali aplinka. Tuo tikslu paleidžiamas *Anaconda Prompt* (*Miniconda*) komandinės eilutės interpretatorius ir atsidariusiame lange surenkama komanda be kabučių:
  „*conda create –name xxxxx*”,
  kuri patvirtinama *Enter* klaviatūros klavišo nuspaudimu.
  Čia xxxxx yra laisvai pasirenkamas vieno žodžio virtualios aplinkos pavadinimas.
- Po to, kai komanda įvykdyta, būtina suaktyvinti virtualią aplinką. Tam komandinės eilutės interpretatoriaus langelyje būtina parašyti šią komandą be kabučių:
  „conda activate xxxxx“,
  ir patvirtinti *Enter* klaviatūros klavišo nuspaudimu.
- Virtualioje aplinkoje įdiegiama šios programų bibliotekos ir įranga:
  - *arch 4.13* [90]
  - *tensorflow* v.2.1.0 [91]
  - *numpy 1.18.4* [92]
  - *pandas* v.1.03 [93]
  - *pyodbc* v.4.0.31 [94]
  - *statsmodels* v.0.11.1 [95]
  - *scikit-learn* v.022.2 [53]
  - *spyder* v.4.1.3 [96]
  - *pyodbc 4.0.30* [94]
    Diegimo komandas galima rasti programinių bibliotekų atitinkamose internetinėse svetainėse.
- Iš *Google Disk* failų saugyklos, panaudojus turimą nuorodą, parsiunčiamas ir į lokalaus kompiuterio disko direktoriją „*C:\Tyrimai*“ išskleidžiamas diegimo failas. Taip yra išskleidžiami failai *tyrimai.accdb* ir *tyrimai_be.accdb*, žemesnio lygmens direktorijos „\data“, kurioje yra 50 tekstinių duomenų failų, direktorija „*python*“, kurioje yra

programiniai *Python* failai, direktorija „\pic“, kurioje saugomos vizualizacijos, direktorija „\doc“, kurioje patalpinta su programiniu projektu susijusi dokumentacija ir direktorija „export“ į kurią yra eksportuojami pasirinkti duomenys.

UML diegimo diagrama pateikiama 13 priede. Programinė įranga paleidžiama atidarant *Microsoft Access* duomenų bazių valdymo sistemos failą *tyrimai.accdb*.

Apibendrinant programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, projektavimo ir programavimo rezultatus, galima teigti:

- Reikalavimai programinei įrangai buvo suformuluoti panaudojant *Volere* šabloną, ir tai leido juos formalizuoti ir struktūrizuoti bei suvaldyti rizikas: programavimo klaidas, algoritmo klaidas, duomenų bazės gedimus.
- Dėl nedidelės projekto apimties ir aiškių reikalavimų projekto programinės įrangos projektavimui ir programavimui buvo pasirinktas *Waterfall* metodas.
- Dviejų programinių komponenčių, sudarytų iš *Microsoft Access* duomenų valdymo sistemos ir *Python* kalba parašytos programinės įrangos, panaudojimas leido suderinti kompaktišką *Microsoft Access* duomenų bazę, patogią *Microsoft Access* duomenų bazės valdymo sistemos formomis paremtą naudotojo sąsają ir *Python* programavimo kalba parašytos programos efektyvumą.
- Galimybė pasiekti duomenų vizualizacijas kompiuterio laikmenoje leidžia jas peržiūrėti nekartojant eksperimento, o išprognozuotų duomenų eksportas į kompiuterio laikmeną suteikia galimybę tolesnei jų analizei programinėmis priemonėmis.

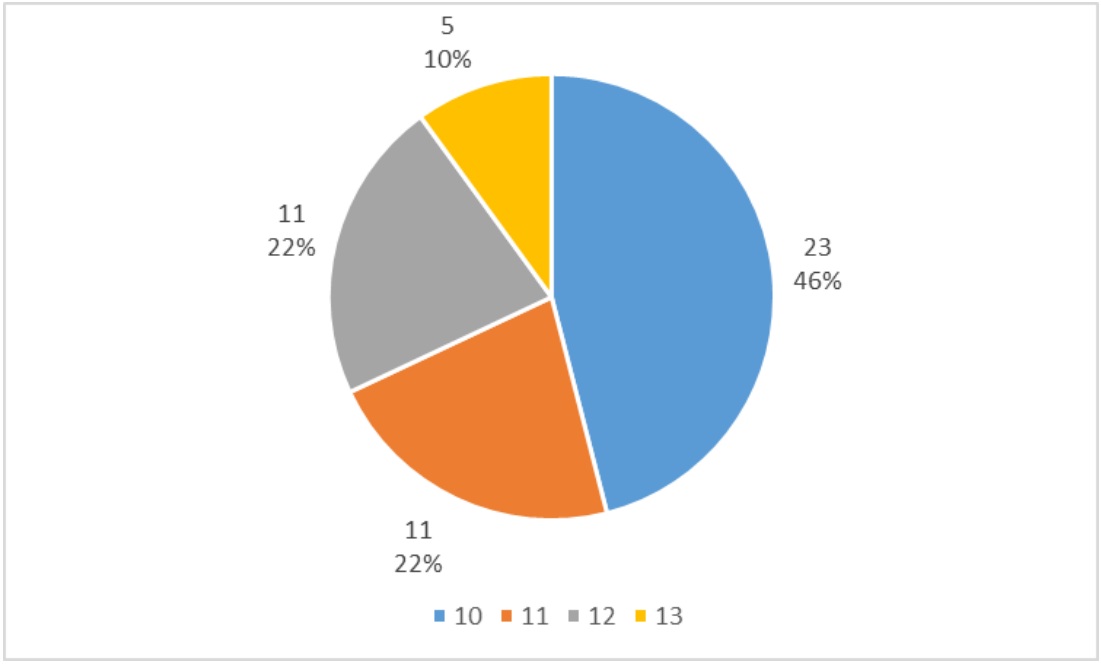
III. KOMPIUTERINIO EKSPERIMENTO REZULTATAI

Šiame magistro darbe keliamai hipotezei patikrinti bei suprojektuotos ir suprogramuotos programinės įrangos, skirtos nestacionarių laiko eilučių tyrimui, veikimo testavimui atliktas kompiuterinis eksperimentas su 50 laiko eilučių duomenų aibių. Kompiuterinis eksperimentas – tai aibė kodo paleidimų su įvairiomis įvestimis. Svarbu, kad paleidus kodą su tomis pačiomis įvestimis gaunami identiški rezultatai, t. y. išvestis yra deterministinė. Taigi, kompiuteriniu eksperimentu galima nustatyti įvestis optimaliam stochastinio proceso realizavimui. Taikant kompiuterinį eksperimentą taip pat galimi prognozių neapibrėžtumo įvertinimai [97].

3.1. Kompiuteriniame eksperimente naudotų duomenų aibių charakteristikos

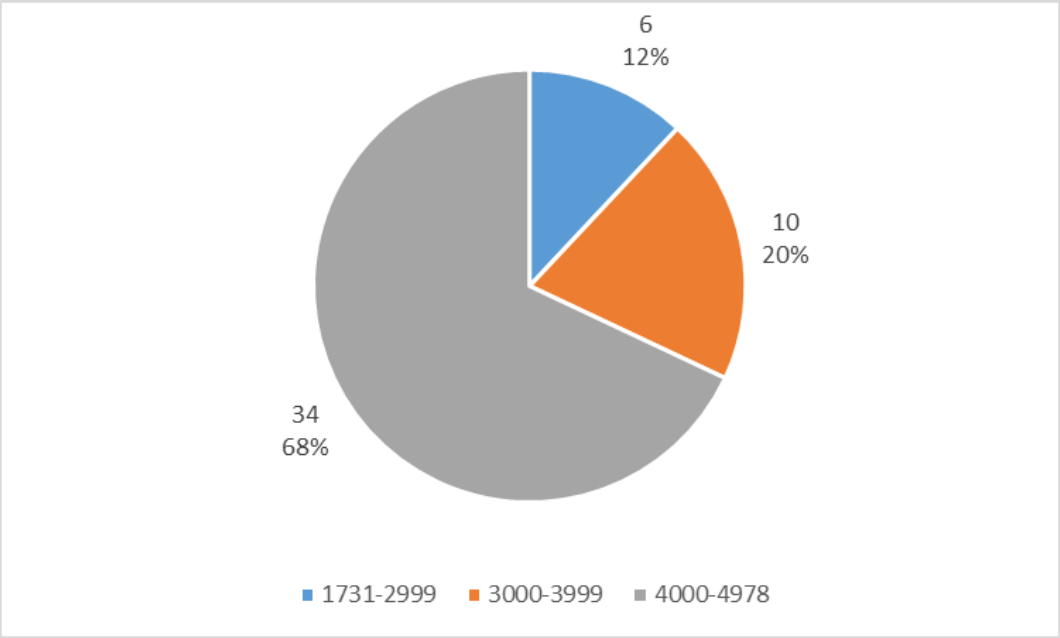
Kompiuteriniam eksperimentui buvo panaudota 50 laiko eilučių duomenų aibių, kurios buvo paimtos iš *kaggle.com* [98]. Analizuojami duomenys yra finansiniai: tai 2000 01 01 – 2019 12 31 Indijos Nacionalinės vertybinių popierių biržos akcijų duomenys. Vertybinių popierių rinkos duomenys yra plačiai analizuojami atsižvelgiant į švietimo, verslo ir asmeninius interesus. Visos duomenų eilutės yra vienos dienos lygio, o kainų ir prekybos vertės yra padalintos kiekvienos akcijos .cvs failais, taip pat pridedamas metaduomenų failas su tam tikra makro informacija apie akcijas.

Siekiant išryškinti svarbiausias duomenų statistines charakteristikas, atlikta duomenų aibių aprašomoji statistinė analizė [99] [100]. Analizuojamos duomenų aibės turi 10 – 13 atributų ir 1731 – 4978 įrašų. Pasiskirstymas pagal atributų skaičių pateikiamas 18 pav.



18 pav. Duomenų aibių pasiskirstymas pagal atributų skaičių

Pasiskirstymas pagal įrašų skaičių vaizduojamas 19 pav.



19 pav. Duomenų aibių pasiskirstymas pagal įrašų skaičių

Visų atributų stacionarumas buvo įvertintas ADF, KPSS ir ZA testais.

3.2 Kompiuterinio eksperimento eiga

50 finansinių duomenų aibių duomenys paruošti darbui su neuroniniais tinklais ir atlikti stacionarumo testai ADF, KPSS ir ZA. Kompiuterinis eksperimentas vykdytas dviem etapais, kurie apibūdinami 7 lentelėje. Iš viso atlikta 440 eksperimentų.

7 lentelė. Kompiuterinio eksperimento etapai

Etapas	Duomenų aibių skaičius	Eksperimentų skaičius	Užduotys
I.	50	50	1. Atlikti kiekvienos duomenų aibės atributų stacionarumo testus ADF, KPSS, ZA.
		150	2. Kiekvienos duomenų aibės duomenis prognozuoti trimis neuroniniais tinklais, parinkus vienodus mokslinėje literatūroje aprašomus optimalius hiperparametrus [101] [102] [103] [104] [105] [106]. 3. Prognozės tikslumą įvertinti metrika sMAPE.
II.	10	240	4. Atrinkus duomenų aibes, kurių sMAPE yra didžiausia, atlikti prognozavimą trimis neuroniniais tinklais keičiant po vieną neuroninio tinklo hiperparametrą ir ieškant optimalaus jų derinio.

Pirmajame etape kiekvienos duomenų aibės failo informacija bei duomenų aibės atributų informacija *Microsoft Access* duomenų bazės valdymo sistemos formos „*Duomenų aibių nuostatos*“ įkelta į *Microsoft Access* duomenų bazę. Duomenų aibės atributai, turintys tik vieną reikšmę, buvo pašalinti, kadangi nesuteikia informacijos neuroniniam tinklui. Taip pat buvo pašalinti atributai,

kuriuose rasta daugiau kaip 3 % nuosekliai einančių tuščių arba atributo duomenų tipą neatitinkančių reikšmių [107].

Buvo pasirinktas laiko eilučių duomenų aibės *tikslo* atributo, kuris bus reikalingas neuroninių tinklų prognozei atlikti, pavadinimas, bei *datos* atributo, kuris reikalingas atributo grafikams sukurti, pavadinimas. Po to duomenys iš failo buvo įkeliami į duomenų bazę. Ši veiksmų eilė buvo atlikta su visomis 50 duomenų aibių.

Įkėlus duomenis, formos „*Programos nuostatos*“ pagalba parinktos programos nuostatos. Kad *Microsoft Access* duomenų bazės valdymo sistemoje veikianti programa galėtų paleisti *Python* terpėje veikiančią programą ir perduoti jai parametrus bei gauti rezultatus, buvo pasirinkta *Python* programų diegimo sistemos *Miniconda* diegimo direktorija, bei *Python* profilio „Tyrimai“ direktorija. Toliau, remiantis [108], buvo parinktos duomenų aibės tuščių reikšmių atstatymo nuostatos, kurios pateikiamos 8 lentelėje.

8 lentelė. Duomenų aibės tuščių reikšmių atstatymo nuostatos

Duomenų aibės atributo duomenų rūšis	**Duomenų aibės reikšmių atstatymo metodas**
Kategorinio tipo reikšmių atstatymas	Duomenų aibės atributo dažniausia reikšmė
Skaitinio tipo reikšmių atstatymas	Duomenų aibės atributo reikšmių vidurkis
Datos tipo reikšmių atstatymas	Duomenų aibės atributo įrašo pašalinimas
Y stulpelio reikšmių atstatymas	Duomenų aibės atributo reikšmių vidurkis

Kitame žingsnyje kiekvienam duomenų tipui buvo nustatytos binarizavimo nuostatos. Kadangi duomenų binarizavimas atliekamas dviem etapais, kiekvienam etapui buvo parinktas atskiras metodas [109]. Duomenų binarizavimo nuostatos pateikiamos 9 lentelėje.

9 lentelė. Duomenų binarizavimo nuostatos

Duomenų tipas	**Pirminis reikšmių atstatymo metodas**	**Antrinis reikšmių atstatymo metodas**
Kategoriniai duomenys	Nevykdomas	Onehootencoder
Skaitiniai duomenys	Nevykdomas	Minmaxscaler [0, 1]
Datos tipo duomenys	Vertimas į savaitės dienas	Minmaxscaler [0, 1]
Y stulpelio reikšmės	Nevykdomas	Minmaxscaler [0, 1]

Neuroninio tinklo mokymo ir testavimo apimčių santykis parinktas 0,6/0,4 [110]. Siekiant, kad eksperimentą būtų galima pakartoti, išjungtas *Tensorflow* bibliotekos atsitiktinių skaičių generatorius. Eksperimentui pasirinkti LSTM, SRNN bei GRU neuroniniai tinklai ir visos 50 duomenų aibių bei šios nuostatos įrašytos į duomenų bazę.

Toliau formų „GRU nuostatos“, LSTM nuostatos“ ir SRNN nuostatos“ numatytuose profiliuose buvo nustatyti atitinkamų neuroninių tinklų numatytieji hiperparametrai. Jie pateikiami 10 lentelėje.

10 lentelė. Neuroninių tinklų hiperparametrų numatytosios reikšmės

Hiperparametro pavadinimas	**Numatytoji reikšmė**	**Paaiškinimas**
Units	50	Neuroninio tinklo paslėptų neuronų skaičius
Activation	tanh	Paslėpto sluoksnio neurono aktyvavimo funkcija
Recurrent activation	sigmoid	Įėjimo, išėjimo ir užmiršimo loginių elementų aktyvavimo funkcija
Use bias	true	Nustato, ar naudojama postūmio funkcija
Kernel initializer	glorot uniform	Inicializuoja svorių matricą, naudojamą įėjimo signalo linijinėms transformacijoms
Recurrent initializer	orthogonal	Inicializuoja svorių matricą, naudojamą rekurentinio sujungimo linijinėms transformacijoms
Bias initializer	zeros	Inicializuoja postūmio vektorių (jei jis naudojamas)
Unit forget bias	true	Užmiršimo elemento inicializacijos metu prideda prie postūmio vienetą. Naudojama, kai postūmis inicializuojamas nuliais
Loss function	mean absolute error	Funkcija, matuojanti absoliutų skirtumą tarp prognozuojamos reikšmės ir tikros reikšmės
Optimizer	adam	Algoritmas, optimizuojantis iteratyvų neuroninio tinklo svorių pasirinkimą
Epochs	50	Skaičius, nurodantis, kiek kartų duomenys praeina per neuroninį tinklą pirmyn ir atgal
Batch size	64	Nurodo, kiek laiko žingsnių telpa į vieną paketą
Shuffle	false	Nurodo, ar prieš apdorojant neuroniniu tinklu laiko eilučių įrašai yra sumaišomi atsitiktine tvarka
Verbose	0	Nurodo vieną iš 0-2 lygių, kurie apsprendžia kaip išsamiai veikimo metu neuroninis tinklas pateikia informaciją naudotojui. 2 yra išsamiausia informacija.

Įvedus neuroninių tinklų hiperparametrus, būtina formoje „Eksperimentai“ sukurti eksperimentą ir įvesti jo nuostatas – pavadinimą, aprašymą ir datą. Po to duomenys įrašomi.

Tam, kad pradėti eksperimentą, reikia spustelėti valdiklį „Vykdyti eksperimentą“. Atsivėrusiame *Python* interpretatoriaus lange galima stebėti programos pateikiamus pranešimus. Pasibaigus programos veikimui, langą galima uždaryti „*Enter*“ klavišo nuspaudimu.

3.3 Kompiuterinio eksperimento pirmojo etapo rezultatai

Pirmajame etape atlikti 50 finansinių duomenų aibių kintamųjų stacionarumo testai ADF, KPSS, ZA ir prognozės SRNN, LSTM, GRU neuroniniais tinklais tikslumo tyrimai, kurių rezultatai pateikiami 11 lentelėje.

11 lentelė. 50 finansinių duomenų aibių stacionarumo testų ir sMAPE tyrimo rezultatai

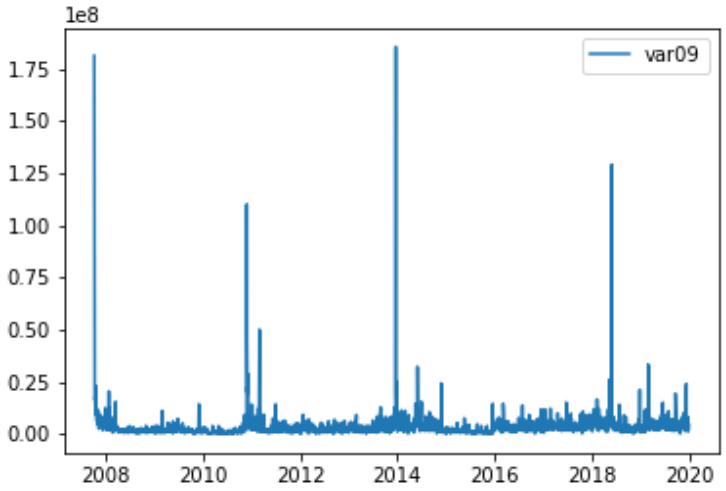
Eil. Nr.	Duomenų aibės pavadinimas	Įrašų skaičius	Atributų skaičius	Bendras stacionarumo testų rezultatas			sMAPE		
				ADF	KPSS	ZA	LSTM	GRU	SRNN
1.	ASIANPAINT_155	4978	10	nestac.	nestac.	stac.	0,2114	0,2208	0,2221
2.	AXISBANK_156	4978	11	nestac.	nestac.	stac.	0,2456	0,2752	0,1849
3.	BAJAJ-AUTO_157	2874	10	nestac.	nestac.	stac.	0,1458	0,0761	0,1005
4.	BAJAJFINSV_158	2873	12	nestac.	nestac.	stac.	0,3461	0,2993	0,3805
5.	BAJFINANCE_159	4907	11	nestac.	nestac.	stac.	0,5395	1,1271	1,1411
6.	BHARTIARTL_160	4446	13	nestac.	nestac.	stac.	0,3075	0,1472	0,8160
7.	BPCL_161	4978	10	nestac.	nestac.	stac.	0,0291	0,0785	0,0510
8.	BRITANNIA_162	4977	10	nestac.	nestac.	stac.	0,3222	0,5214	0,4851
9.	CIPLA_163	4978	10	nestac.	stac.	stac.	0,0981	0,0939	0,0934
10.	COALINDIA_164	2270	12	nestac.	nestac.	stac.	0,2488	0,1534	0,2851
11.	DRREDDY_165	4978	10	nestac.	nestac.	stac.	0,0521	0,3364	0,1017
12.	EICHERMOT_166	4973	10	nestac.	nestac.	nestac.	0,4642	0,1617	0,8182
13.	GAIL_167	4657	10	nestac.	nestac.	nestac.	0,1392	0,0708	0,0734
14.	GRASIM_168	4978	10	nestac.	nestac.	stac.	0,5649	0,1346	1,1315
15.	HCLTECH_169	4972	10	nestac.	stac.	stac.	0,1237	0,1613	0,3070
16.	HDFC_170	4978	10	nestac.	nestac.	stac.	0,2805	0,3031	0,4683
17.	HEROMOTOCO_172	4978	11	nestac.	nestac.	nestac.	0,0905	0,9473	0,7842
18.	HINDUNILVR_174	4978	11	nestac.	nestac.	stac.	0,1817	0,1976	0,4733
19.	ICICIBANK_175	4978	10	nestac.	nestac.	stac.	0,1789	0,6141	1,7924
20.	INDUSINDBK_176	4657	10	nestac.	nestac.	nestac.	0,0687	0,3320	0,6182
21.	INFY_177	4978	11	nestac.	stac.	stac.	0,2715	0,2244	0,5187
22.	INFRATEL_178	1731	13	nestac.	nestac.	nestac.	0,5930	1,5667	1,0858
23.	IOC_179	4978	10	nestac.	stac.	stac.	0,2353	0,2350	1,3231
24.	ITC_180	4978	10	nestac.	stac.	stac.	0,3230	0,3720	1,6922
25.	YESBANK_181	3588	12	nestac.	nestac.	stac.	0,1196	0,2691	0,8325
26.	JSWSTEEL_182	3666	13	nestac.	nestac.	stac.	0,1609	0,1156	0,4887
27.	KOTAKBANK_183	4657	11	nestac.	nestac.	nestac.	0,6206	0,3224	0,6165
28.	LT_184	3856	12	nestac.	stac.	stac.	0,0619	0,0598	0,0883
29.	MARUTI_185	4099	12	nestac.	nestac.	nestac.	0,1335	0,1075	0,9891
30.	MM_186	4978	10	nestac.	nestac.	nestac.	0,0958	0,0331	0,2039
31.	NESTLEIND_187	2478	12	nestac.	nestac.	nestac.	0,1448	0,5384	0,2565
32.	NTPC_188	3760	12	nestac.	nestac.	nestac.	0,4429	0,0851	0,3297
33.	ONGC_189	4978	10	nestac.	nestac.	stac.	0,1385	0,0488	0,2639
34.	POWERGRID_190	3031	12	nestac.	nestac.	nestac.	0,4388	0,1710	0,5850
35.	RELIANCE_191	4978	10	nestac.	nestac.	nestac.	0,1857	0,1291	0,4588
36.	SBIN_192	4978	10	nestac.	nestac.	stac.	0,0703	0,0403	0,2791
37.	SUNPHARMA_193	4978	10	nestac.	nestac.	stac.	0,8292	0,5289	1,8554
38.	TATAMOTORS_194	4978	11	nestac.	nestac.	nestac.	0,0371	0,3274	0,6656
39.	TATASTEEL_195	4978	11	nestac.	nestac.	nestac.	0,2708	0,1568	0,1230
40.	TCS_196	3811	12	nestac.	nestac.	nestac.	0,3285	0,3037	0,4839
41.	TECHM_197	3307	12	nestac.	nestac.	nestac.	0,0678	0,0902	0,2580
42.	TITAN_198	4978	10	nestac.	nestac.	stac.	1,2447	1,1860	1,1556

11 lentelė (tęsinys). 50 finansinių duomenų aibių stacionarumo testų ir sMAPE tyrimo rezultatai

Eil. Nr.	Duomenų aibės pavadinimas	Įrašų skaičius	Atributų skaičius	Bendras stacionarumo testų rezultatas			sMAPE		
				ADF	KPSS	ZA	LSTM	GRU	SRNN
43.	ULTRACEMCO_199	3812	12	nestac.	nestac.	nestac.	0,1800	0,5536	1,6727
44.	UPL_200	3959	13	nestac.	nestac.	nestac.	0,1232	0,2114	0,4549
45.	VEDL_201	4977	11	nestac.	nestac.	stac.	0,0426	0,3218	0,3006
46.	WIPRO_202	4978	10	nestac.	stac.	stac.	0,4437	0,3578	0,1616
47.	ZEEL_203	4978	11	nestac.	stac.	stac.	0,5869	1,0978	1,2833
48.	ADANIPORTS_339	2994	13	nestac.	stac.	stac.	0,7168	0,0686	0,5245
49.	HDFCBANK_345	3978	10	nestac.	nestac.	stac.	0,6061	0,4338	1,3175
50.	HINDALCO_346	4978	11	nestac.	nestac.	stac.	0,1380	0,4040	0,1857

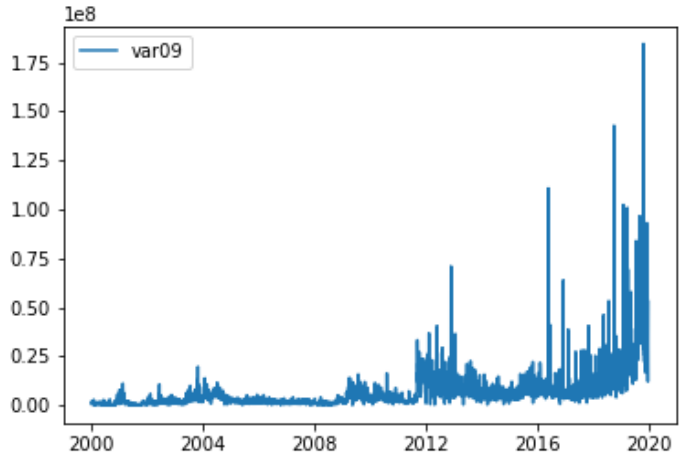
1. Duomenų stacionarumas. Duomenų aibių atributų stacionarumą, nustatytą ADF, KPSS ir ZA testais, galima peržiūrėti formoje „Duomenų nuostatos“. Formoje „Eksperimentai“ patekiama kiekvienos duomenų aibės atributų bei neuroninių tinklų veiklos vizualizacija. Kadangi stacionarumo testai atlikti kiekvienam duomenų aibės atributui, vertinant bendrą duomenų stacionarumą (11 lentelė) laikoma, kad duomenų aibės duomenys yra nestacionarūs, jei bent vieno jos atributo duomenys yra nestacionarūs. Kadangi nė vienas iš šių testų nėra universalus, atliekant eksperimentą buvo analizuojami ir grafiniai vaizdai. Keletas pavyzdžių:

- Griežto stacionarumo duomenų atributo pavyzdys – ADF, KPSS ir ZA testai rodo stacionarumą. Grafinis vaizdas pateikiamas 21 pav.



20 pav. Griežto stacionarumo duomenų pavyzdys

- Silpno stacionarumo duomenų atributo pavyzdys – ADF ir ZA testai rodo stacionarumą, o KPSS – nestacionarumą (22 pav.).



21 pav. Silpno stacionarumo duomenų pavyzdys

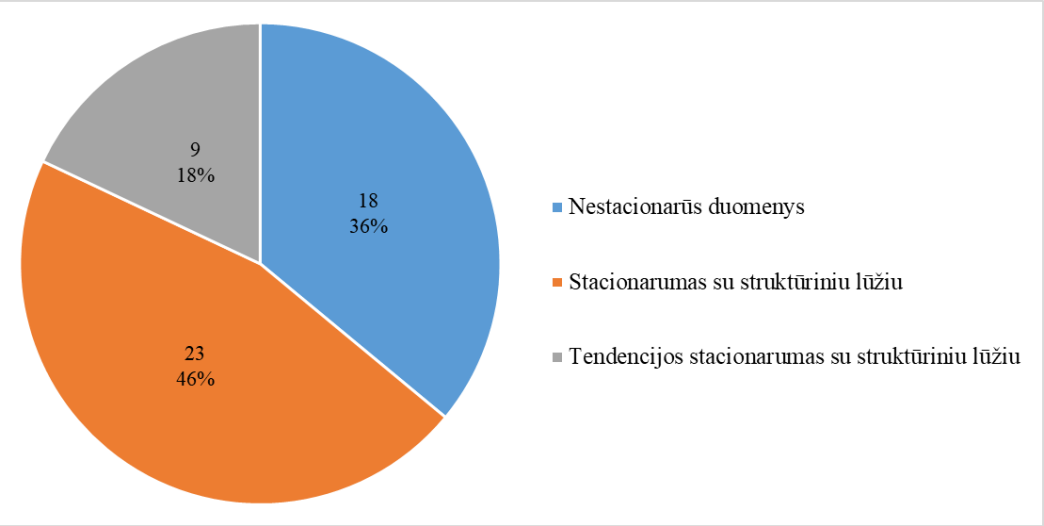
- Stacionarių duomenų su tendencija ir struktūriniu lūžiu pavyzdys – ADF testas rodo nestacionarumą, tačiau KPSS ir ZA testai rodo stacionarumą (23 pav.).

22 pav. Stacionarių duomenų su struktūriniu lūžiu pavyzdys

- Nestacionarių duomenų pavyzdys – visi testai rodo nestacionarumą (24 pav.).

23 pav. Nestacionarių duomenų pavyzdys

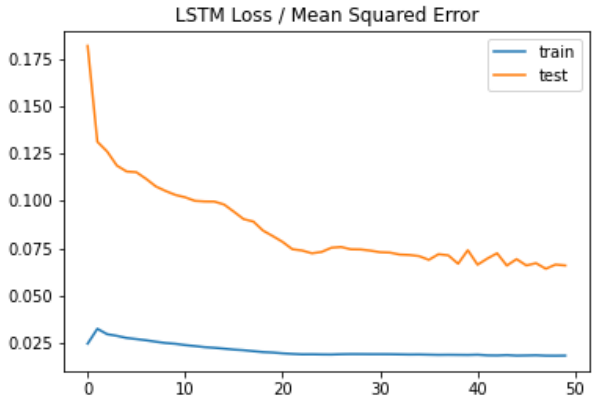
Pagal bendrą stacionarumo testų rezultatą duomenų aibes galima suskirstyti taip: 18 duomenų aibių duomenys yra nestacionarūs, 23 duomenų aibių duomenys turi struktūrinį lūžį, 9 duomenų aibių duomenys turi tendenciją ir struktūrinį lūžį. Kadangi tiriamos duomenų aibės yra finansiniai duomenys, kuriems nebūdingas, pvz., sezoniškumas, kitų nestacionarumo rūšių nenustatyta (25 pav.).



24 pav. Kompiuteriniame eksperimente naudotų duomenų aibių stacionarumas

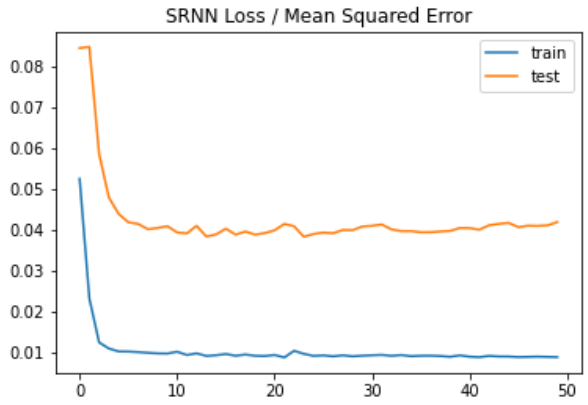
2. Tinklo mokymas. Tai rekurentinio neuroninio tinklo paslėpto neuronų sluoksnio svorių matricos suformavimas, kurios pagrindu galima prognozuoti tinklui dar nežinomus duomenis:

- Nepakankamai apmokytas tinklas (26 pav.). Nepakankamai apmokytas tinklas demonstruoja gerus rezultatus su mokymo duomenų aibe ir blogus – su testavimo. Tai galima pastebėti grafike kai mokymo praradimų kreivė yra žemiau nei testavimo praradimų kreivė, kuri turi tendenciją, rodančią, jog mokant neuroninį tinklą toliau, įmanoma pasiekti geresnį rezultatą, padidinus epochų skaičių.



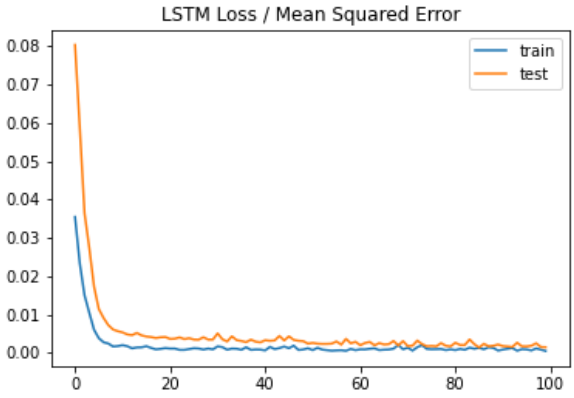
25 pav. Nepakankamai apmokyto tinklo pavyzdys: per mažas epochų skaičius

- Taip pat tinklas gali būti nepakankamai apmokytas, jei mokymo ir testavimo praradimų kreivės po kritimo tampa lygiagrečios. Tai vyksta dėl per mažo paslėptų neuronų skaičiaus (27 pav.).



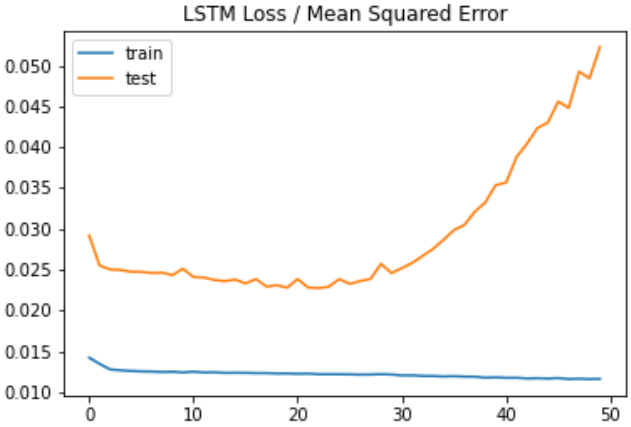
26 pav. Nepakankamai apmokyto tinklo pavyzdys: per mažas paslėptų neuronų skaičius

- Gerai apmokytas tinklas. Gerai apmokytą tinklą galima atpažinti iš grafiko, kuriame mokymo ir testavimo praradimų kreivės leidžiasi ir stabilizuojasi tame pačiame taške (28 pav.).



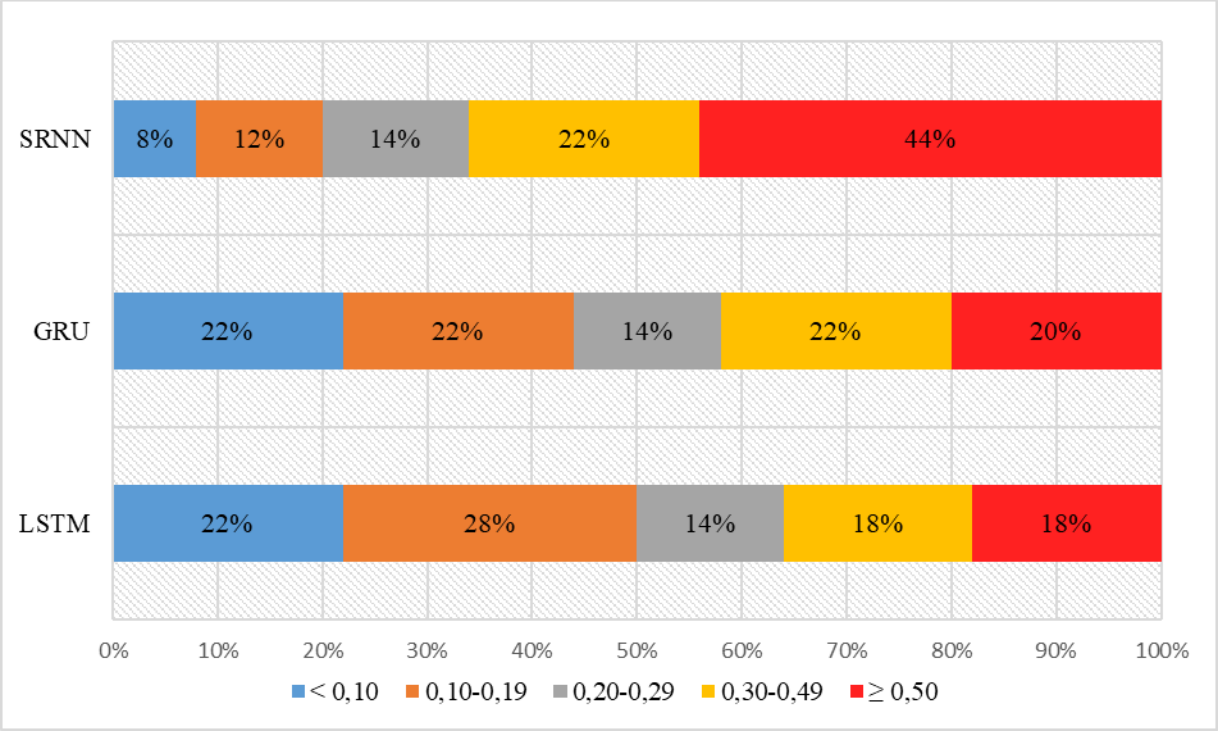
27 pav. Nepakankamai apmokyto tinklo pavyzdys: per mažas epochų skaičius

- Permokintas tinklas. Taip vadinamas tinklas, kurio apmokymo praradimų kreivė yra artėjanti prie X ašies, tuo tarpu testavimo praradimų kreivė pradžioje artėja prie X ašies, o po to ima nuo jos tolti. Tokiu atveju rekomenduojama mažinti epochų skaičių (29 pav.).



28 pav. Permokinto tinklo pavyzdys

3. Klaida sMAPE. Pirmame kompiuterinio eksperimento etape gauti sMAPE klaidos rezultatai (11 lentelė) apibendrintai pateikiami 30 pav.



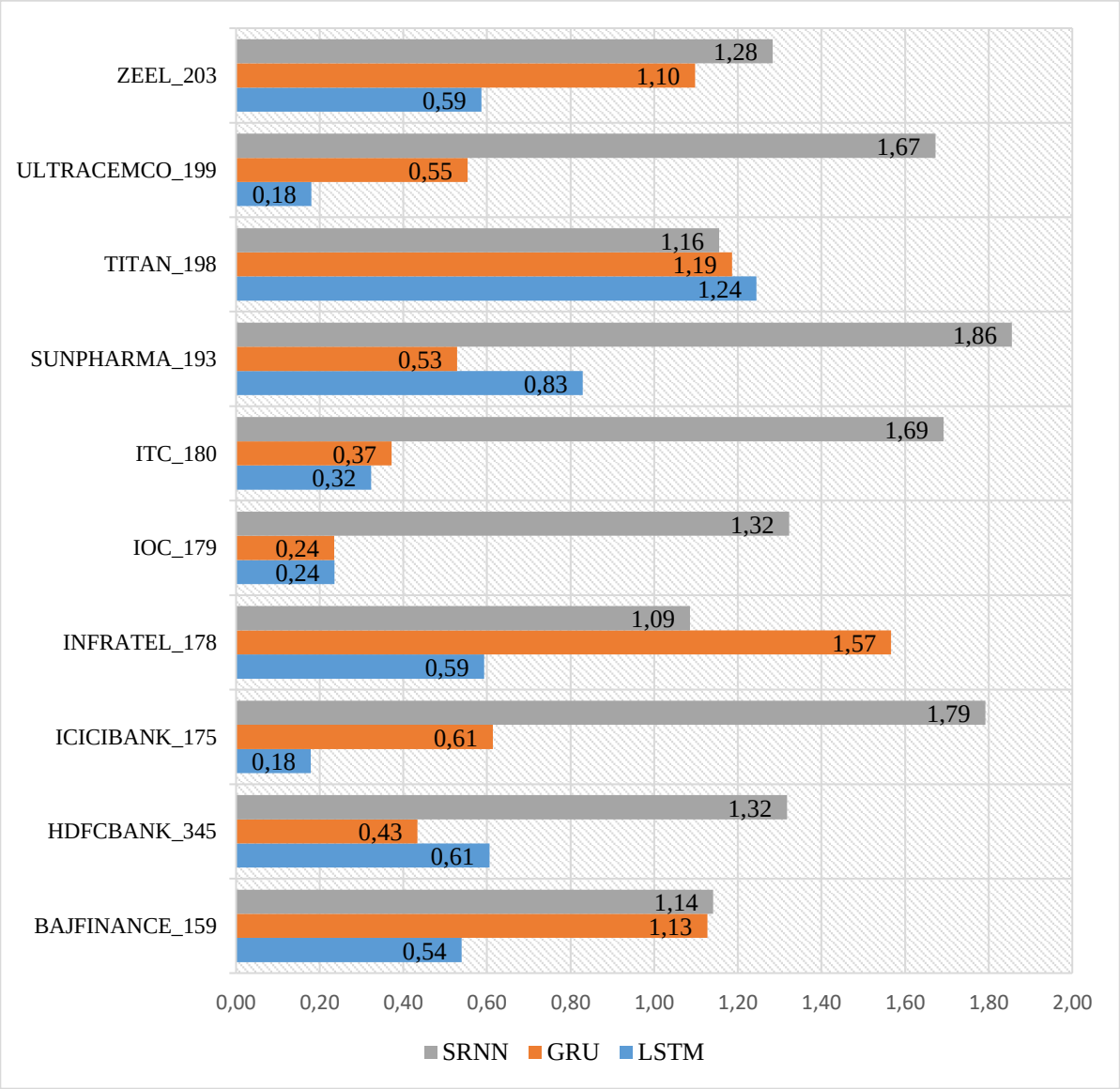
29 pav. SRNN, GRU ir LSTM tinklų sMAPE klaidų pasiskirstymas

Analizuojant grafiką pastebima:

- LSTM ir GRU prognozių tikslumas yra panašus: 22 % duomenų aibių abu tinklai prognozuoja su mažesne nei 0,10 (10 %) klaida sMAPE. Su labai didelėmis klaidomis (virš 50 %) abu tinklai atlieka atitinkamai 18 % ir 20 %.
- SRNN klaidų sMAPE pasiskirstymas ženkliai skiriasi nuo LSTM ir GRU: iki 10 % prognozės klaidų sudaro vos 8 % duomenų aibių, o virš 50 % klaida yra net 44 % duomenų aibių.
- Galima daryti išvadą, kad LSTM ir GRU labiau tinka nestacionarių finansinių laiko eilučių duomenų prognozavimui.

3.4 Kompiuterinio eksperimento antrojo etapo rezultatai

Apskaičiavus visų 50 tirtų duomenų aibių sMAPE klaidas antrajam kompiuterinio eksperimento etapui atrinkta 10 duomenų aibių, kurias prognozuodamas bet kuris iš trijų neuroninių tinklų padarė didžiausią klaidą (31 pav.).



30 pav. Didžiausios sMAPE reikšmės

Analizuojant duomenis pastebima, kad daugiausia didelių sMAPE reikšmių yra SRNN neuroninio tinklo prognozėse. Vidutinės sMAPE reikšmės yra:

- LSTM: 0,53166 (53 %)
- GRU: 0,7715 (77 %)
- SRNN: 1,43191 (143 %)

Paanalizavus duomenų aibių įrašų ir atributų skaičius (11 lentelė), pastebima, kad 2 duomenų aibės turi daugiau nei 3800 įrašų ir 10-12 atributų, 7 duomenų aibės turi virš 4900 įrašų ir 10-11

atributų, 1 duomenų aibė turi 1731 įrašą ir 13 atributų. Kaip žinoma iš šiame magistro darbe atliktos mokslinės literatūros analizės, SRNN neuroniniam tinklui būdinga nykstančio arba sprogstančio gradiento problema, dėl kurios SRNN tinklas negali išmokti ilgalaikių priklausomybių. Galima daryti išvadą, kad didelės SRNN tinklo sMAPE reikšmės yra susijusios su tiriamų duomenų aibių dydžiu. GRU ir LSTM neuroniniams tinklams nykstančio arba sprogstančio gradiento problema nebūdinga, todėl jų prognozės klaidos yra mažesnės.

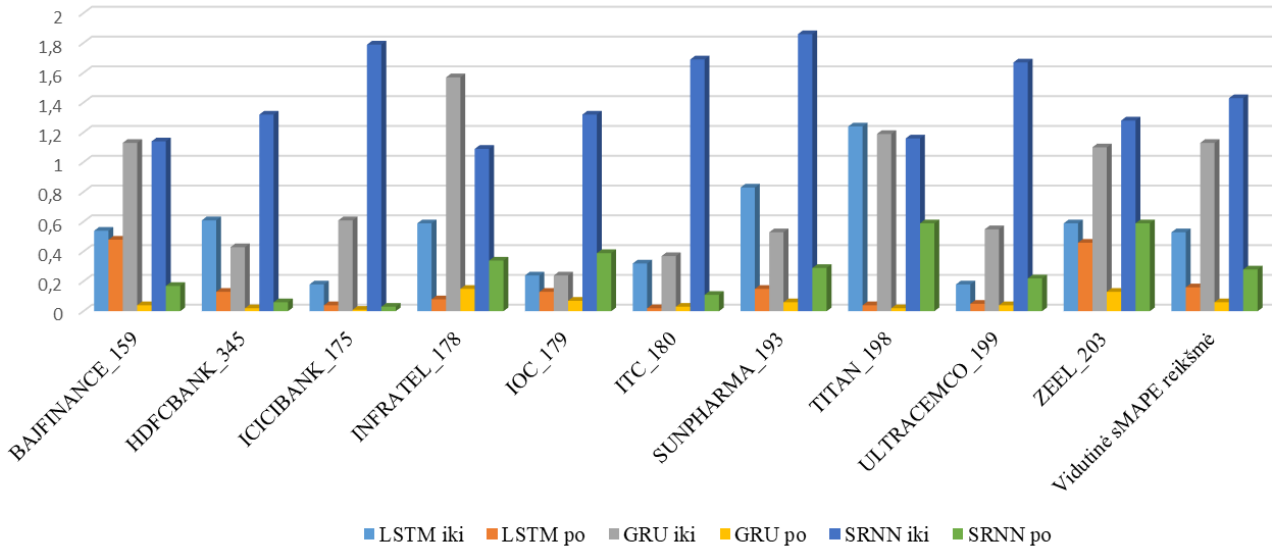
Toliau su 10 atrinktų duomenų aibių atliktas prognozavimas trimis neuroniniais tinklais keičiant po vieną neuroninio tinklo hiperparametrą ir ieškant optimalaus jų derinio. Laikytasi nuostatų:

- Kiekvieno hiperparametro įtaka buvo vertinama atskirai;
- sMAPE pokyčio vertinimui apskaičiuotas reikšmingumo lygmuo (*p* reikšmė) pasinaudojant *Microsoft Excel Analysis ToolPak*;
- Nustačius reikšmę, kuri statistiškai reikšmingai ($p < 5$ %) sumažina sMAPE, ji įrašoma į lentelę (14, 15, 16 priedai);
- Nustačius optimalų hiperparametrų derinį kiekvienai iš 10 duomenų aibei ir kiekvienam iš 3 neuroninių tinklų, atlikti galutiniai eksperimentai, kurių rezultatai pateikiami 12 lentelėje.

12 lentelė. sMAPE reikšmės iki ir po neuroninių tinklų hiperparametrų derinimo

Duomenų aibės pavadinimas	**sMAPE reikšmė iki / po hiperparametrų suderinimo**					
	LSTM		**GRU**		**SRNN**	
	iki	**po**	**iki**	**po**	**iki**	**po**
BAJFINANCE_159	0,54	0,48	1,13	0,04	1,14	0,17
HDFCBANK_345	0,61	0,13	0,43	0,02	1,32	0,06
ICICIBANK_175	0,18	0,04	0,61	0,01	1,79	0,03
INFRATEL_178	0,59	0,08	1,57	0,15	1,09	0,34
IOC_179	0,24	0,13	0,24	0,07	1,32	0,39
ITC_180	0,32	0,02	0,37	0,03	1,69	0,11
SUNPHARMA_193	0,83	0,15	0,53	0,06	1,86	0,29
TITAN_198	1,24	0,04	1,19	0,02	1,16	0,59
ULTRACEMCO_199	0,18	0,05	0,55	0,04	1,67	0,22
ZEEL_203	0,59	0,46	1,10	0,13	1,28	0,59
Vidutinė sMAPE reikšmė	0,53	0,16	1,13	0,06	1,43	0,28
Vidutinės sMAPE pokytis	69,8 %		92,3 %		80,5 %	

Analizuojant rezultatus pastebima, kad hiperparametrų įtaka neuroninių tinklų prognozės tikslumui yra labai didelė: parinkus optimalius hiperparametrus galima sumažinti sMAPE klaidą daugiau nei 90 % (32 pav.).



31 pav. Hiperparametrų įtaka neuroninių tinklų prognozės tikslumui

Analizuojant 14, 15, 16 priedų duomenis taip pat galima padaryti apibendrintas išvadas, kurių neuroninių tinklų hiperparametrų įtaka sMAPE klaidai yra didžiausia:

- Optimizavimo funkcija dažniausiai buvo keičiama į „*adagrad*“. Pastebėta, kad geriausi visų trijų neuroninių tinklų rezultatai buvo pasiekiami šią optimizavimo funkciją naudojant kartu su duomenų sumaišymu (*shuffle = true*), epochų skaičiaus padidinimu iki 100 ar 200 (*Epochs = 200*) bei paketo dydžio sumažinimu iki 10-20 (*Batch size = 20).* Paketo dydžio pakeitimas į kartotinį skaičiui 5 pagerino prognozavimą galimai todėl, kad neuroninio tinklo svorių matrica yra atnaujinama pateikus neuroniniam tinklui duomenų imčių skaičių, aprašomą hiperparametru „batch size“, o visi analizei naudoti duomenys buvo surinkti darbo dienomis, t.y. nuo pirmadienio iki penktadienio.
- Taip pat duomenų prognozavimo klaidos tikslumui įtaką padarė rekurentinio inicializatoriaus funkcijos „*variance scaling*“ nustatymas (*Recurrent initializer* = variance scaling) bei aktyvavimo funkcijos pakeitimas į „*sigmoid*“ (*Activation = sigmoid).*

Apibendrinant kompiuterinio eksperimento rezultatus galima teigti:

- Taikant numatytuosius neuroninių tinklų hiperparametrus LSTM ir GRU prognozių tikslumas yra panašus: 22 % duomenų aibių abu tinklai prognozuoja su mažesne nei

0,10 (10 %) klaida sMAPE. SRNN klaidų sMAPE pasiskirstymas skiriasi nuo LSTM ir GRU: iki 10 % prognozės klaidų sudaro vos 8 % duomenų aibių, o virš 50 % klaida yra net 44 % duomenų aibių.

- Vidutinė sMAPE taikant numatytuosius neuroninių tinklų hiperparametrus yra: LSTM: 0,53166 (53 %), GRU: 0,7715 (77 %), SRNN: 1,43191 (143 %).
- Individualiai kiekvienai duomenų aibei paderinus neuroninio rinklo hiperparametrus prognozės tikslumą galima pagerinti 70%-90%.

IŠVADOS

1. Reikalavimų programinei įrangai, skirtai nestacionarių laiko eilučių tyrimui, formulavimas pagal *Volere* šabloną leido juos formalizuoti ir struktūrizuoti, įvardinti bei suvaldyti tokias projekto rizikas, kaip programavimo klaidos, algoritmo klaidos, duomenų bazės gedimai.
2. Aiškūs reikalavimai programinei įrangai ir vaizdus architektūros pateikimas UML diagramomis leido išryškinti programinių modulių funkcijas, sklandžiai pereiti nuo koncepto prie programinio modelio, užtikrinti programavimo atkartojamumą.
3. Sukurtas programinis kodas, kurio pagalba programos naudotojas gali atlikti nestacionarių laiko eilučių duomenų analizę: įvykdyti išankstinį duomenų paruošimą, stacionarumo nustatymo testus ADF, KPSS, ZA, sukurti trijų neuroninių tinklų – SRNN, LSTM, GRU, – modelius, juos apmokyti ir išprognozuoti duomenis. Naudotojas vizualiai bei skaitiniais testais įvertinęs analizuojamų duomenų stacionarumą bei naudotojo sąsajos pagalba keisdamas pasirinktų neuroninių tinklų hiperparametrus gali siekti maksimalaus prognozavimo tikslumo.
4. Atlikus kompiuterinį eksperimentą su 50 finansinių duomenų aibių duomenimis panaudojant pasirinktas programinės įrangos nuostatas nustatyta:
    - Taikant numatytuosius neuroninių tinklų hiperparametrus analizuotus finansinius duomenis tiksliausiai prognozuoja LSTM neuroninis tinklas, o prasčiausiai SRNN. Vidutinės sMAPE reikšmės taikant numatytuosius hiperparametrus yra LSTM: 0,53 (53 %), GRU: 0,77 (77 %), SRNN: 1,43 (143 %).
    - Individualiai kiekvienai duomenų aibei paderinus neuroninio rinklo hiperparametrus prognozės tikslumą galima pagerinti 70 % – 90 %. Vidutinės sMAPE reikšmės taikant individualiai nustatytus hiperparametrus yra LSTM: 0,16 (16 %), GRU: 0,06 (6 %), SRNN: 0,28 (28 %).
    - Geriausi visų trijų neuroninių tinklų rezultatai buvo pasiekiami keičiant šiuos hiperparametrus: optimizavimo funkciją „*adagrad*“ naudojant kartu su duomenų sumaišymu (*shuffle = true*), epochų skaičiaus padidinimu iki 100 ar 200 (*Epochs = 200*) bei paketo dydžio sumažinimu iki 10-20 (*Batch size = 20)*. Paketo dydžio pakeitimas į kartotinį skaičiui 5 pagerino analizuotų finansinių duomenų prognozavimą todėl, kad neuroninio tinklo svorių matrica yra atnaujinama pateikus neuroniniam tinklui duomenų imčių skaičių, aprašomą hiperparametru „*batch size*“, o visi analizei naudoti duomenys buvo surinkti darbo dienomis.

5. Kompiuterinio eksperimento rezultatai patvirtina šio magistro darbo hipotezę: rekurentiniais neuroniniais tinklais galima prognozuoti nestacionarių laiko eilučių duomenis jų nestacionarizavus.

REKOMENDACIJOS

Šiame darbe pateikiamą programinę įrangą, skirtą nestacionarių laiko eilučių tyrimui, galima tobulinti. Galimos tobulinimo rekomendacijos:

- *Microsoft Access* duomenų bazių valdymo sistema nėra pritaikyta dideliems duomenų kiekiams ir turi gedimo riziką, todėl galima būtų duomenų bazę perkelti į SQL serverį.
- Duomenų bazės saugojimas lokaliame kompiuteryje nėra patogus, kadangi apriboja vienu metu dirbančių naudotojų skaičių, apsunkina jos rezervinių kopijų darymo procesą. Rekomenduojama duomenų bazę perkelti į „debesį“.
- *Microsoft Access* duomenų bazių valdymo sistemos formomis suformuota naudotojo sąsaja yra patogi, tačiau, duomenų bazę perkėlus į „debesį“, būtų tikslinga naudotojo sąsają sukurti remiantis internetine naršykle. Tam galima būtų panaudoti *Python* paremtą atviro kodo žiniatinklio karkasą *Django*.
- Sukurtos programinės įrangos *Python* kalba parašytą dalį būtų tikslinga perkelti į *Google Colaboratory* žiniatinklio prieglaudą. Tai leistų panaudoti suteikiamus vaizdo procesorių pajėgumus, smarkiai pagreitinančius neuroninio tinklo skaičiavimą.

LITERATŪRA

[1] S. Glen, „What is the Augmented Dickey Fuller Test?," 2020. [Tinkle]. Available: https://www.statisticshowto.com/adf-augmented-dickey-fuller-test/. [Kreiptasi 10 05 2020].

[2] S. Glen, „ARMA model," 2020. [Tinkle]. Available: https://www.statisticshowto.com/arma-model/. [Kreiptasi 10 05 2020].

[3] J. Chen, „Autoregressive Integrated Moving Average (ARIMA)," 2020. [Tinkle]. Available: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp. [Kreiptasi 10 05 2020].

[4] J. Chung, C. Gulcehre, K. Cho ir Y. Bengio, „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014. [Tinkle]. Available: https://arxiv.org/abs/1412.3555. [Kreiptasi 16 02 2020].

[5] Merriam-Webster, „Definition Of Time Series By Merriam-Webster," Merriam-Webster, 2020. [Tinkle]. Available: https://www.merriam-webster.com/dictionary/time%20series. [Kreiptasi 06 03 2020].

[6] „Time Series Analysis," Statistics Solutions, 2020. [Tinkle]. Available: https://www.statisticssolutions.com/time-series-analysis/. [Kreiptasi 12 05 2020].

[7] K. Lukoševičiūtė, Chaotinių procesų rekonstravimo bei algebrinių sekų modeliai laiko eilučių prognozavime, Kaunas: Kauno technologijos universitetas, 2012.

[8] „Time series," Cambridge University Press, 2020. [Tinkle]. Available: https://dictionary.cambridge.org/dictionary/english/time-series. [Kreiptasi 12 05 2020].

[9] F. Gers, „Long Short-Term Memory in Recurrent Neural Networks," 2001. [Tinkle]. Available: https://www.researchgate.net/profile/Felix_Gers/publication/2562741_Long_Short-Term_Memory_in_Recurrent_Neural_Networks/links/5759410a08ae9a9c954e77f5.pdf. [Kreiptasi 20 01 2020].

[10] R. Pascanu, T. Mikolov ir B. Yoshua, „On the difficulty of training Recurrent Neural Networks," *arXiv,* 2013.

[11] A. Graves, A. Mohamed ir G. Hinton, „Speech recognition with deep recurrent neural networks," įtraukta *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013.

[12] D. Montgomery, Introduction to time series analysis and forecasting, New Jersey: John Wiley & Sons. Inc ., 2008.

[13] G. E. Box, G. M. Jenkins, G. C. Reinsel ir G. M. Ljung, Time Series Analysis: Forecasting and Control, New York: Wiley, New York, 2016.

[14] G. E. Box, Time series analysis: forecasting and control, New Jersey: Prentice-Hall , International, Inc., 1994.

[15] D. Hendry ir F. Pretis, „All Change! The Implications of Non-stationarity for Empirical Modelling, Forecasting and Policy," 2016. [Tinkle]. Available: https://www.oxfordmartin.ox.ac.uk/downloads/briefings/All_Change.pdf. [Kreiptasi 01 05 2020].

[16] J. W. Kantelhard, S. Zschiegnera ir E. Koscielny, „Multifractal detrended fluctuation analysis of nonstationary time series," *Physica A: Statistical Mechanics and its Applications,* t. 316, nr. 1-4, pp. 87-114, 2002.

[17] W. A. Barnett ir F. Jawadi, Nonlinear Modeling of Economic and Financial Time-Series, Bingley : Emerald Group Publishing Limited, 2010.

[18] P. Cízek, W. Härdle ir R. Weron, „Nonstationary Models for Time Series," Humboldt-Universitet zu Berlin, 2005. [Tinkle]. Available: http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/tutorials/xegbohtmlnode37.html. [Kreiptasi 12 05 2020].

[19] F. J. H. Don, „Forecast Uncertainty in Economics,“ *Predictability and Nonlinear Modelling in Natural Sciences and Economics,* t. 1, nr. 1, pp. 568-580, 1994.

[20] P. Christensen, K. Gillingham ir N. W., „Uncertainty in forecasts of long-run economic growth,“ *Proceedings of the National Academy of Sciences,* t. 115, nr. 21, pp. 5409-5414, 2018.

[21] R. Ghazali, A. J. Hussain ir N. M. Nawi, „Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network,“ *Neurocomputing,* t. 72, nr. 10-12, pp. 2359-2367, 2009.

[22] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang ir J. K. Zhenyu, „Time series forecasting for nonlinear and non-stationary processes: a review and comparative study,“ *IIE Transactions,* t. 47, nr. 10, pp. 1053-1071, 2015.

[23] J. Brownlee, Deep Learning for Time Series Forecasting. Predict the Future with MLPs, CNNs and LSTMs in Python, 1.4 mont., eBook, 2018.

[24] L. Stabingienė, Ekonometrika, Klaipėda: Klaipėdos universitetas, 2014.

[25] J. Liu ir S. Chien, „Non-stationary Multivariate Time Series Prediction with Selective Recurrent Neural Networks,“ *PRICAI 2019: Trends in Artificial Intelligence,* t. 11672, nr. 1, pp. 636-649, 2019.

[26] Z. Che, S. Purushotham, K. Cho, D. Sontag ir Y. Liu, „Recurrent Neural Networks for Multivariate Time Series with Missing Values,“ *Scientific Reports,* t. 8, nr. 1, 2018.

[27] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke ir J. Schmidhuber, „A Novel Connectionist System for Unconstrained Handwriting Recognition,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* t. 31, nr. 5, pp. 855-868, 2009.

[28] L. Yang, Z. Wenming, C. Zhen ir Z. Tong, „Face recognition based on recurrent regression neural network,“ *Neurocomputing,* t. 297, pp. 50-58, 2018.

[29] F. A. Gers, J. Schmidhuber ir F. Cummins, „Learning to Forget: Continual Prediction with LSTM,“ 1999. [Tinkle]. Available: https://pdfs.semanticscholar.org/e10f/98b86797ebf6c8caea6f54cacbc5a50e8b34.pdf. [Kreiptasi 03 11 2019].

[30] J. Yim, „A Comparison of Neural Networks with Time Series Models for Forecasting Returns on a Stock Market Index,“ įtraukta *IEA/AIE '02: Proceedings of the 15th international conference on Industrial and engineering applications of artificial intelligence and expert systems: developments in applied artificial intelligence*, 2002.

[31] P. G. Zhang ir M. Qi, „Neural Network Forecasting for Seasonal and Trend Time Series,“ *European Journal of Operational Research,* t. 160, nr. 2, pp. 501-514, 2005.

[32] J. Gamboa, „Deep Learning for Time-Series Analysis,“ *arXiv,* 2017.

[33] M. Roondiwala, H. Patel ir S. Varma, „Predicting Stock Prices Using LSTM,“ *International Journal of Science and Research,* t. 6, nr. 4, pp. 1754-1756, 2017.

[34] A. Karpathy, J. Johnson ir L. Fei-Fei, „Visualizing and Understanding Recurrent Networks,“ 2015. [Tinkle]. Available: https://arxiv.org/abs/1506.02078. [Kreiptasi 22 10 2019].

[35] K. Althelaya, E. El-Alfy ir S. Mohammed, „Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU),“ 2018. [Tinkle]. Available: https://ieeexplore.ieee.org/abstract/document/8593076. [Kreiptasi 19 01 2020].

[36] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. J. S. Yang ir Z. Kong, „Time series forecasting for nonlinear and non-stationary processes: a review and comparative study,“ *IIE Transactions,* t. 47, nr. 10, pp. 1053-1071, 2015.

[37] O. Hammer, „Past 4 - the Past of the Future,“ Natural History Museum, University of Oslo, 2020. [Tinkle]. Available: https://folk.uio.no/ohammer/past/. [Kreiptasi 20 05 2020].

[38] A. Cottrell ir R. Lucchetti, „Gnu Regression, Econometrics and Time-series Library,“ Wake Forest University, Università Politecnica delle Marche, 2020. [Tinkle]. Available: http://gretl.sourceforge.net/. [Kreiptasi 20 05 2020].

[39] Alyuda Researche, „Alyuda,“ Alyuda Researche, 2020. [Tinkle]. Available: https://www.alyuda.com/product/forecasting-software. [Kreiptasi 20 05 2020].

[40] Attrasoft, „Attrasoft PredictorPro,“ Attrasoft, 2020. [Tinkle]. Available: http://attrasoft.com/predictorpro/. [Kreiptasi 20 05 2020].

[41] Neurosolutions, „The Premier Neural Network Software,“ Neurosolutions, 2020. [Tinkle]. Available: http://www.neurosolutions.com/. [Kreiptasi 20 05 2020].

[42] V. Čekanavičius ir G. Murauskas, Statistika ir jos taikymai, Vilnius: TEV, 2006.

[43] National Institute of Standards and Technology, „NIST/SEMATECH e-Handbook of Statistical Methods,“ National Institute of Standards and Technology, 2003. [Tinkle]. Available: https://www.itl.nist.gov/div898/handbook/. [Kreiptasi 06 03 2020].

[44] S. Palachy, „Stationarity in time series analysis,“ 2020. [Tinkle]. Available: https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322. [Kreiptasi 01 05 2020].

[45] S. Palachy, „Detecting stationarity in time series data,“ 30 04 2020. [Tinkle]. Available: https://www.kdnuggets.com/2019/08/stationarity-time-series-data.html. [Kreiptasi 01 05 2020].

[46] E. W. Weisstein, „Hypothesis Testing,“ https://mathworld.wolfram.com/, [Tinkle]. Available: https://mathworld.wolfram.com/HypothesisTesting.html. [Kreiptasi 28 12 2019].

[47] S. Bisgaard ir M. Kulahci, Time series analysis and forecasting by example, New Jersey: John Wiley & Sons, Inc., 2011.

[48] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt ir Y. Shin, „Testing the null hypothesis of stationarity against the alternative of a unit root,“ *Journal of Econometrics,* t. 54, p. 159–178, 1992.

[49] I. Nilsson, „Unit Root Tests And Structural Breaks in the Swedish Electricity Price,“ Lulca, 2009.

[50] K. Sheppard, „arch.unitroot.ZivotAndrews,“ Arch, 2019. [Tinkle]. Available: https://arch.readthedocs.io/en/latest/unitroot/generated/arch.unitroot.ZivotAndrews.html. [Kreiptasi 20 05 2020].

[51] M. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. Tyukov, T. A. Janovsky ir V. A. Kamaev, „A survey of forecast error measures,“ *World Applied Sciences Journal,* pp. 171-176, 2013.

[52] R. J. Hyndman ir A. B. Koehler, „Another look at measures of forecast accuracy,“ *International journal of forecasting,* t. 22, nr. 4, pp. 679-688, 2006.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel ir M. Blondel, „Scikit-learn: Machine Learning in Python,“ *Journal of Machine Learning Research,* t. 12, pp. 2825-2830, 2011.

[54] I. Kyriakidis, J. Kukkonen, K. Karatzas, G. Papadourakis ir A. Ware, „New Statistical Indices for Evaluating Model Forecasting,“ 2015. [Tinkle]. Available: http://iranarze.ir/wp-content/uploads/2017/12/53-English-IranArze.pdf. [Kreiptasi 29 12 2019].

[55] R. Nau, „What’s a good value for R-squared?,“ 2019. [Tinkle]. Available: https://people.duke.edu/~rnau/rsquared.htm. [Kreiptasi 17 03 2020].

[56] S. Glen, „What does it mean to Detrend Data?,“ 2020. [Tinkle]. Available: https://www.statisticshowto.com/detrend-data/. [Kreiptasi 01 05 2020].

[57] R. J. Hyndman ir G. Athanasopoulos, Forecasting: principles and practice, 2 mont., Melbourne, Australia: OTexts, 2018.

[58] Merriam-Webster, „Definition of neural network By Merriam-Webster,“ Merriam-Webster, Incorporated, 2020. [Tinkle]. Available: https://www.merriam-webster.com/dictionary/neural%20network. [Kreiptasi 12 05 2020].

[59] E. Fiesler, „Neural Network Classification and Formalization,“ *Computer Standards & Interfaces,* t. 16, nr. 3, pp. 231-239, 1994.

[60] X. S. Hu, S. Zagoruyko ir N. Komodakis, „Exploring Weight Symmetry in Deep Neural Networks,“ 2019. [Tinkle]. Available: https://arxiv.org/pdf/1812.11027.pdf. [Kreiptasi 12 01 2020].

[61] R. Rojas, Neural Networks - A Systematic Introduction, Berlin: Springer, 1996.

[62] J. Patterson ir A. Gibson, Deep Learning– A Practitioner's Approach, O'Reilly, 2017.

[63] I. Goodfellow, Y. Bengio ir C. Aaron, Deep Learning, The MIT Press, 2016.

[64] J. L. Elman, „Finding structure in time,“ *COGNITIVE SCIENCE,* t. 14, nr. 2, pp. 179-211, 1990.

[65] J. J. Hopfield, „Neural Networks and Physical Systems with Emergent Collective Computational Abilities,“ *Proceedings of the National Academy of Sciences,* 1982.

[66] S. Hochreiter ir J. Schmidhuber, „Long short-term memory,“ t. 9, nr. 8, p. 1735, 1997.

[67] H. Sak, A. Senior ir F. Beaufays, „Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling,“ [Tinkle]. Available: https://static.googleusercontent.com/media/research.google.com/lt//pubs/archive/43905.pdf. [Kreiptasi 19 02 2020].

[68] A. Graves, G. Wayne ir I. Danihelka, „Neural Turing Machines,“ *arXiv,* 2014.

[69] M. Schuster ir K. K. Paliwal, „Bidirectional recurrent neural networks,“ *IEEE Transactions on Signal Processing,* t. 45, nr. 11, pp. 2673-2681, 1997.

[70] H. Jaeger, „Echo state networks,“ *Sholarpedia,* 2007.

[71] G. E. Hinton, „Boltzmann machine,“ *Scholarpedia,* t. 2, nr. 5, p. 1668, 2007.

[72] G. E. Hinton ir T. J. Sejnowski, „Learning and releaming in Boltzmann machines,“ *Parallel distributed processing: explorations in the microstructure of cognition,* t. 1, pp. 282-317, 1986.

[73] A. Gulli ir S. Pal, Deep Learning with Keras, 1 mont., Packt Publishing, 2017.

[74] T. M. Mitchell, Machine Learning, New York: McGraw-Hill Science/Engineering/Math, 1997.

[75] Y. Bengio, P. Lamblin, D. Popovici ir H. Larochelle, „Greedy Layer-Wise Training of Deep Networks,“ įtraukta *Advances in Neural Information Processing Systems 19*, Montreal, Quebec, 2007.

[76] C. Olah, „Understanding LSTM Networks,“ 27 08 2017. [Tinkle]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Kreiptasi 15 11 2019].

[77] D. G. Torres ir H. Qiu, „Applying recurrent neural networks for multivariate time series forecasting of volatile financial data,“ 2018. [Tinkle]. Available: https://www.researchgate.net/publication/322027012_Applying_Recurrent_Neural_Networks_for_Multivariate_Time_Series_Forecasting_of_Volatile_Financial_Data. [Kreiptasi 16 11 2019].

[78] „SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral,“ Existek, 2020. [Tinkle]. Available: https://medium.com/existek/sdlc-models-explained-agile-waterfall-v-shaped-iterative-spiral-e3f012f390c5. [Kreiptasi 03 05 2020].

[79] „Software Development Life Cycle (SDLC),“ 2020, [Tinkle]. Available: https://www.w3schools.in/sdlc-tutorial/software-development-life-cycle-sdlc/. [Kreiptasi 04 05 2020].

[80] C. Gilley, „The Pros and Cons of Agile Product Development,“ Uservoice, [Tinkle]. Available: https://community.uservoice.com/blog/the-pros-and-cons-of-agile-product-development/. [Kreiptasi 04 05 2020].

[81] „SDLC - Iterative Model,“ Tutorialspoint, 2020. [Tinkle]. Available: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm. [Kreiptasi 04 05 2020].

[82] B. Goldy, „What is Spiral Model? A Simple Explanation of Spiral Software Development Life Cycle (SDLC),“ 01 10 2018. [Tinkle]. Available: https://www.techuz.com/blog/what-is-sdlc-spiral-model/. [Kreiptasi 04 05 2020].

[83] „What is V-model- advantages, disadvantages and when to use it?,“ TRY QA, 2020. [Tinkle]. Available: http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/. [Kreiptasi 05 05 2020].

[84] S. M. Rajkumar, „Waterfall Model in Software Development Life Cycle,“ [Tinkle]. Available: https://www.softwaretestingmaterial.com/waterfall-model-in-sdlc/. [Kreiptasi 05 05 2020].

[85] Atlantic Systems Guild, „Volere Requirements Specification Template,“ 2019. [Tinkle]. Available: https://www.volere.org/templates/volere-requirements-specification-template/. [Kreiptasi 20 05 2020].

[86] S. R. J. Robertson, Mastering the Requirements Process, sBook mont., Hoboken: Pearson ITP, 2012.

[87] Microsoft, „Access“ duomenų bazės bendrinimo būdai,“ Microsoft, 2020. [Tinkle]. Available: https://support.office.com/lt-lt/article/%E2%80%9Eaccess-duomen%C5%B3-baz%C4%97s-bendrinimo-b%C5%ABdai-03822632-da43-4d8f-ba2a-68da245a0446. [Kreiptasi 10 05 2020].

[88] Microsoft, „12 Ways To Troubleshoot Microsoft Access Crashing issue When Opening Report,“ MS Access, 2020. [Tinkle]. Available: http://www.accessrepairnrecovery.com/blog/fix-ms-access-keep-crashing-issue. [Kreiptasi 10 05 2020].

[89] Continuum Analytics, „Miniconda,“ Continuum Analytics, Inc. (dba Anaconda, Inc.)., 2017. [Tinkle]. Available: https://docs.conda.io/en/latest/miniconda.html. [Kreiptasi 05 10 2019].

[90] K. Sheppard, „bashtage/arch: Release 4.13 (Version 4.13). Zenodo.,“ 28 03 2019. [Tinkle]. Available: https://zenodo.org/record/593254. [Kreiptasi 12 04 2020].

[91] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard ir R. Jozefowicz, „TensorFlow: Large-scale machine learning on heterogeneous systems,“ 2015. [Tinkle]. Available: https://www.tensorflow.org/. [Kreiptasi 01 10 2019].

[92] NumFOCUS, Inc., „NumPy. The fundamental package for scientific computing with Python,“ 2020. [Tinkle]. Available: https://numpy.org/. [Kreiptasi 12 04 2020].

[93] W. McKinney, „Pandas,“ Community, [Tinkle]. Available: https://pandas.pydata.org/. [Kreiptasi 28 09 2019].

[94] Python community, „PYODBC,“ 2020. [Tinkle]. Available: https://pypi.org/project/pyodbc/. [Kreiptasi 02 10 2019].

[95] S. Seabold ir J. Perktold, „Statsmodels: Econometric and statistical modeling with python,“ 2010. [Tinkle]. Available: https://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf. [Kreiptasi 18 12 2019].

[96] P. Raybaut, „Spyder,“ Python community, [Tinkle]. Available: www.spyder-ide.org. [Kreiptasi 12 10 2019].

[97] J. Sacks, T. J. Mitchell, W. Welch ir H. P. Wynn, „Design and analysis of computer experiments,“ 01 1989. [Tinkle]. Available: https://www.researchgate.net/publication/265574083_Design_and_analysis_of_computer_experiments_With_comments_and_a_rejoinder_by_the_authors. [Kreiptasi 20 05 2020].

[98] R. Rao, „NIFTY-50 Stock Market Data (2000-2019),“ 2000-2019. [Tinkle]. Available: https://www.kaggle.com/rohanrao/nifty50-stock-market-data/version/2. [Kreiptasi 25 04 2020].

[99] W. Mendenhall, R. J. Beaver ir B. M. Beaver, Introduction to Probability and Statistics, Belmont: Cengage Learning, 2008.

[100] R. V. McCarthy, M. M. McCarthy, W. Ceccucci ir L. Halawi, „What Do Descriptive Statistics Tell Us,“ 2019. [Tinkle]. Available: https://link.springer.com/chapter/10.1007/978-3-030-14038-0_3. [Kreiptasi 30 11 2019].

[101] M. Vierumäki, „Hyperparameter Optimization with Keras,“ 15 05 2018. [Tinkle]. Available: https://towardsdatascience.com/hyperparameter-optimization-with-keras-b82e6364ca53. [Kreiptasi 24 23 2020].

[102] J. Brownlee, „How to Choose Loss Functions When Training Deep Learning Neural Networks,“ 20 04 2020. [Tinkle]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/. [Kreiptasi 02 05 2020].

[103] J. Brownlee, „Loss and Loss Functions for Training Deep Learning Neural Networks,“ 23 10 2019. [Tinkle]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. [Kreiptasi 02 05 2020].

[104] B. Basnet, „LSTM Optimizer Choice ?,“ 2016. [Tinkle]. Available: https://deepdatascience.wordpress.com/2016/11/18/which-lstm-optimizer-to-use/. [Kreiptasi 07 04 2020].

[105] J. Brownlee, „Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,“ 14 11 2019. [Tinkle]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Kreiptasi 12 04 2020].

[106] S. Sharma, „Epoch vs Batch Size vs Iterations,“ 23 10 2017. [Tinkle]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Kreiptasi 05 23 2020].

[107] Q. Xiao, „3 step to complete data cleaning and data preprocessing,“ 09 08 2019. [Tinkle]. Available: https://easyai.tech/en/blog/data-cleaning-and-preprocessing-for-beginners/. [Kreiptasi 15 05 2020].

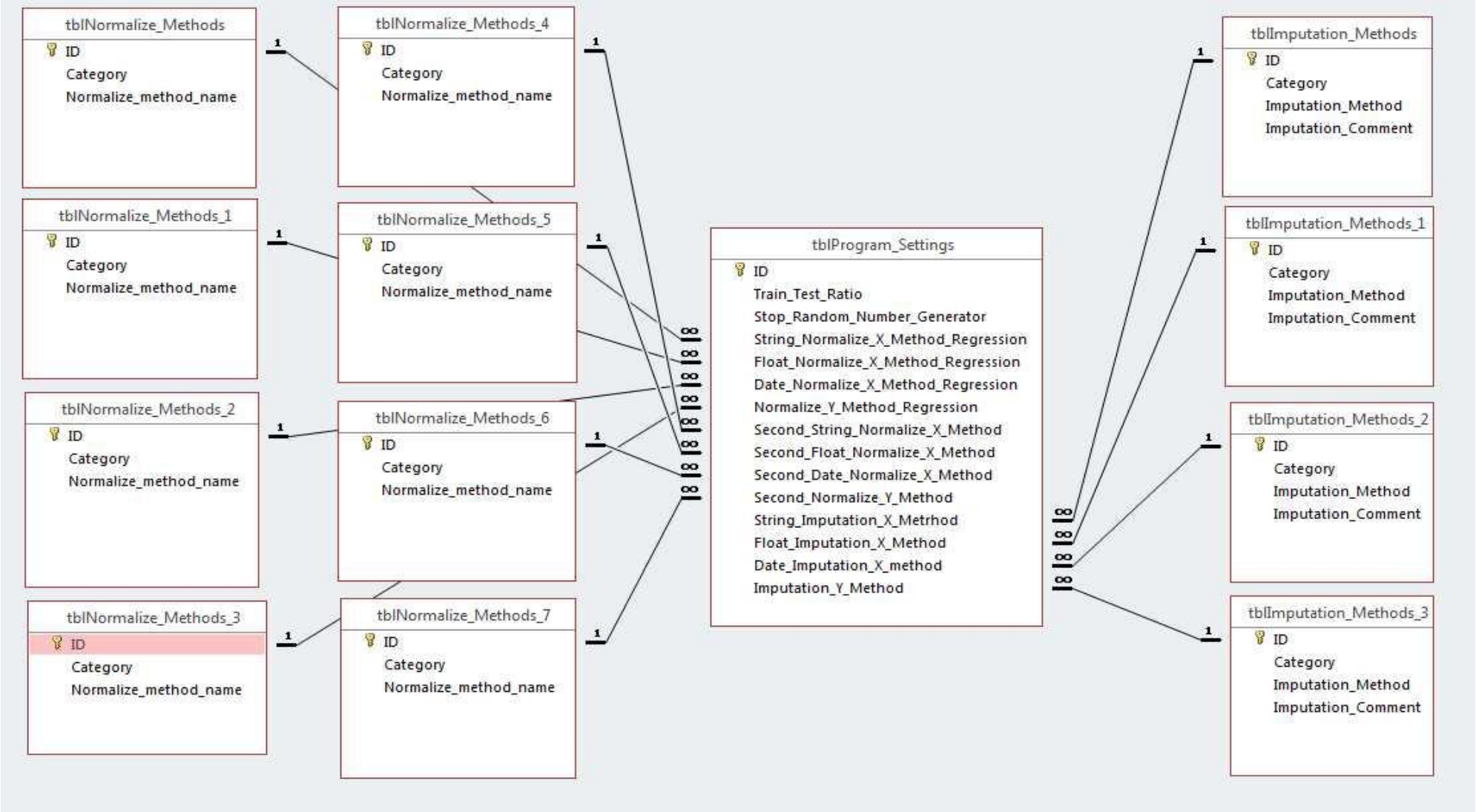
[108] S. Ray, „A Comprehensive Guide to Data Exploration,“ 10 01 2016. [Tinkle]. Available: https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/. [Kreiptasi 17 03 2020].

[109] J. Brownlee, „How To Prepare Your Data For Machine Learning in Python with Scikit-Learn,“ 11 12 2019. [Tinkle]. Available: https://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/. [Kreiptasi 12 04 2020].

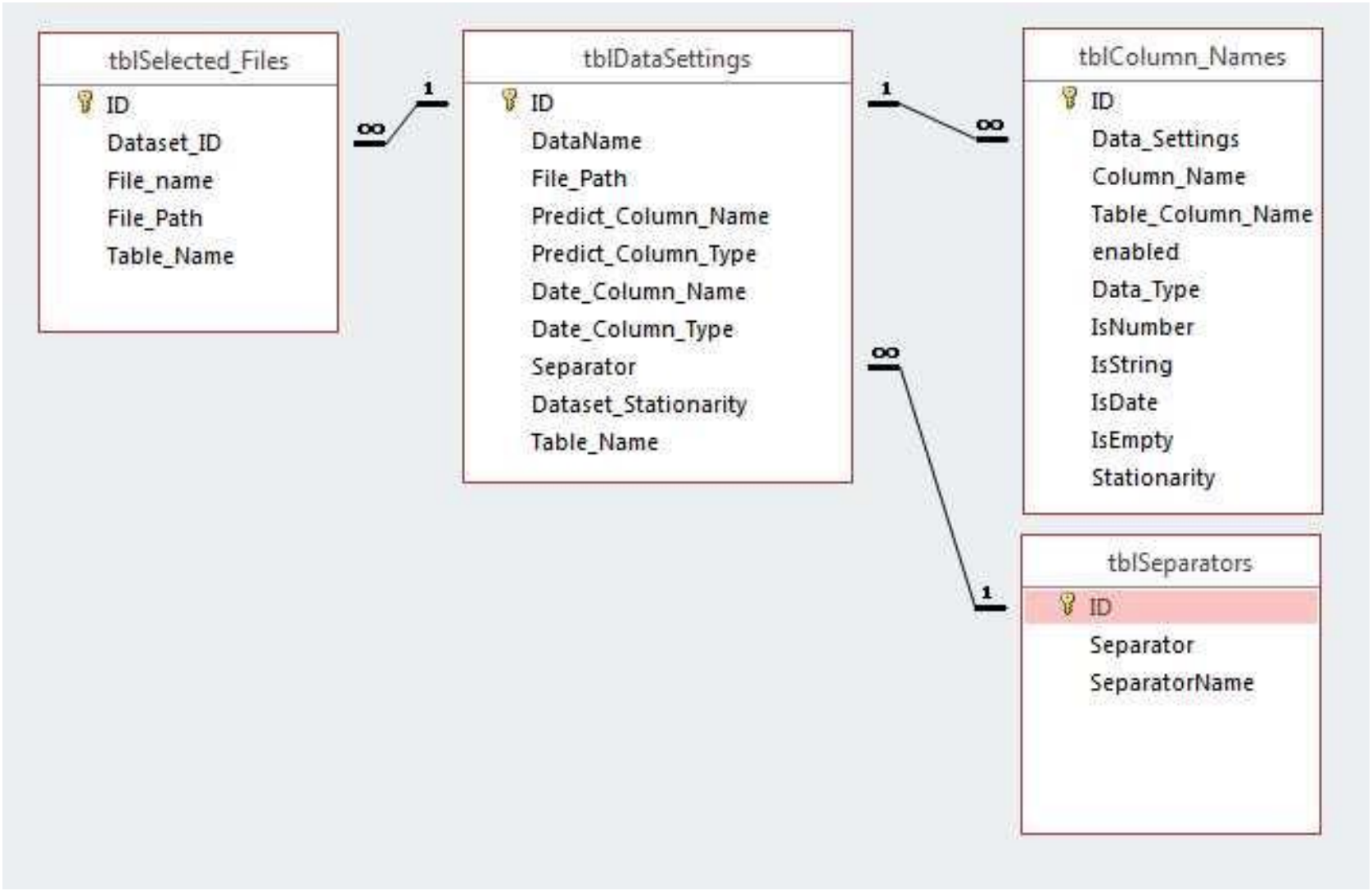
[110] R. Draelos, „Best Use of Train/Val/Test Splits, with Tips for Medical Data,“ 15 09 2019. [Tinkle]. Available: https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/. [Kreiptasi 07 03 2020].

PRIEDAI

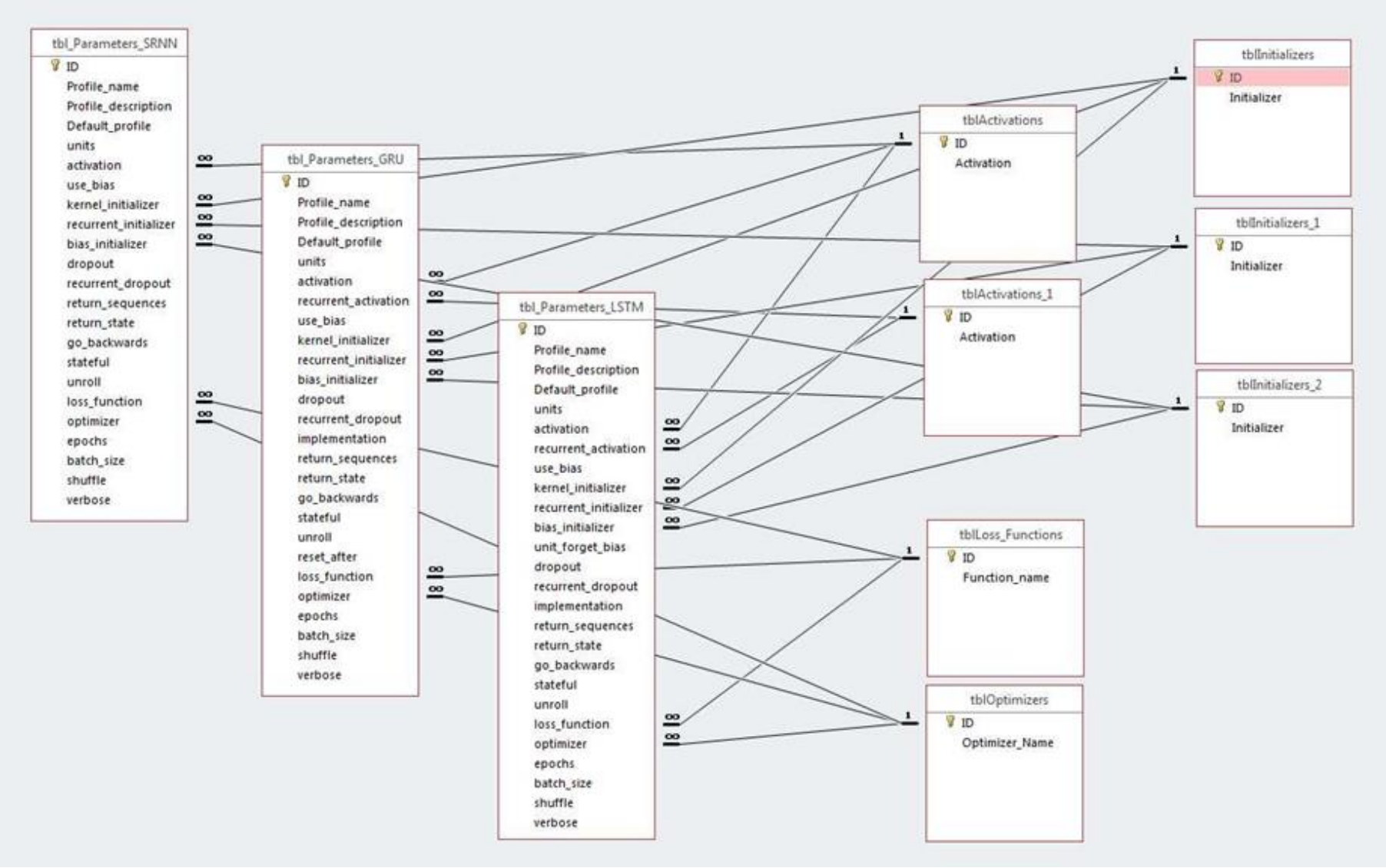
Su programos nuostatomis susijusiai informacijai saugoti skirtų lentelių tarpusavio ryšiai



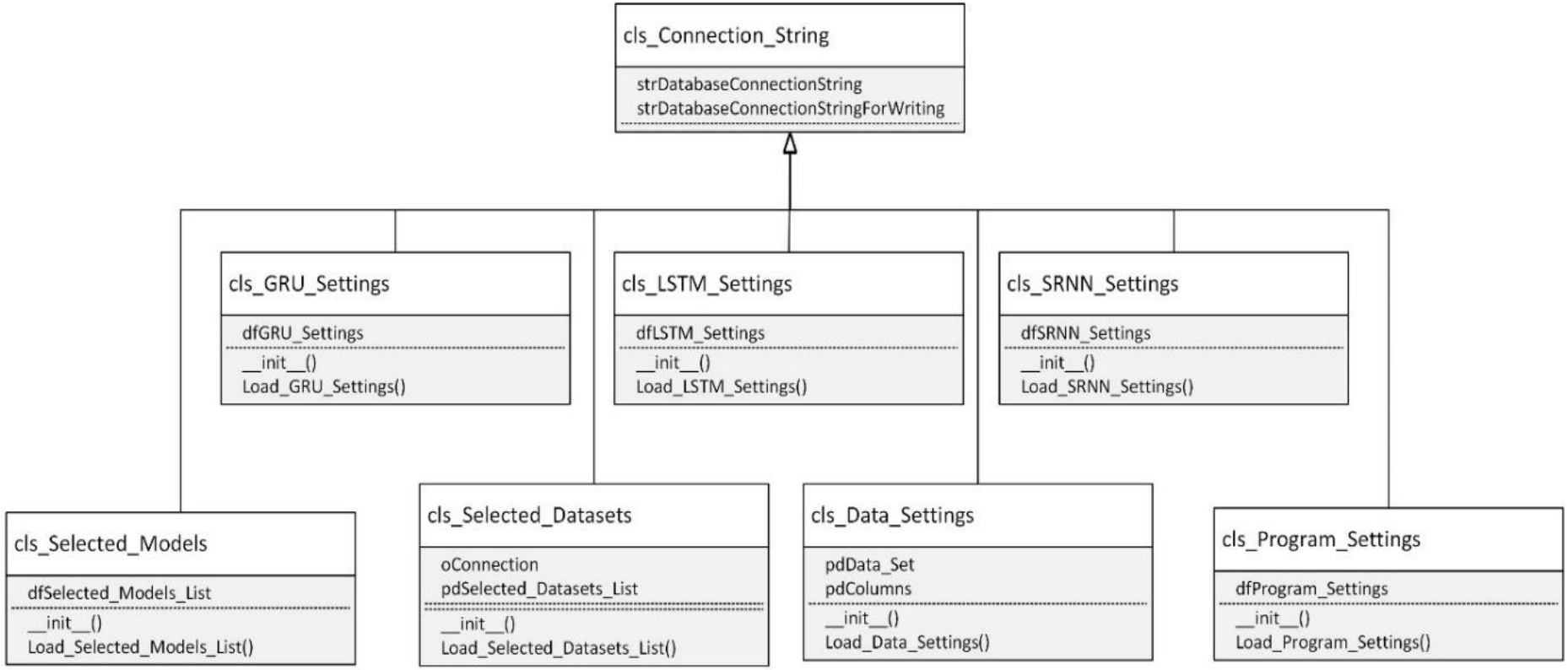
Su duomenų aibių nuostatomis susijusiai informacijai saugoti skirtų lentelių tarpusavio ryšiai



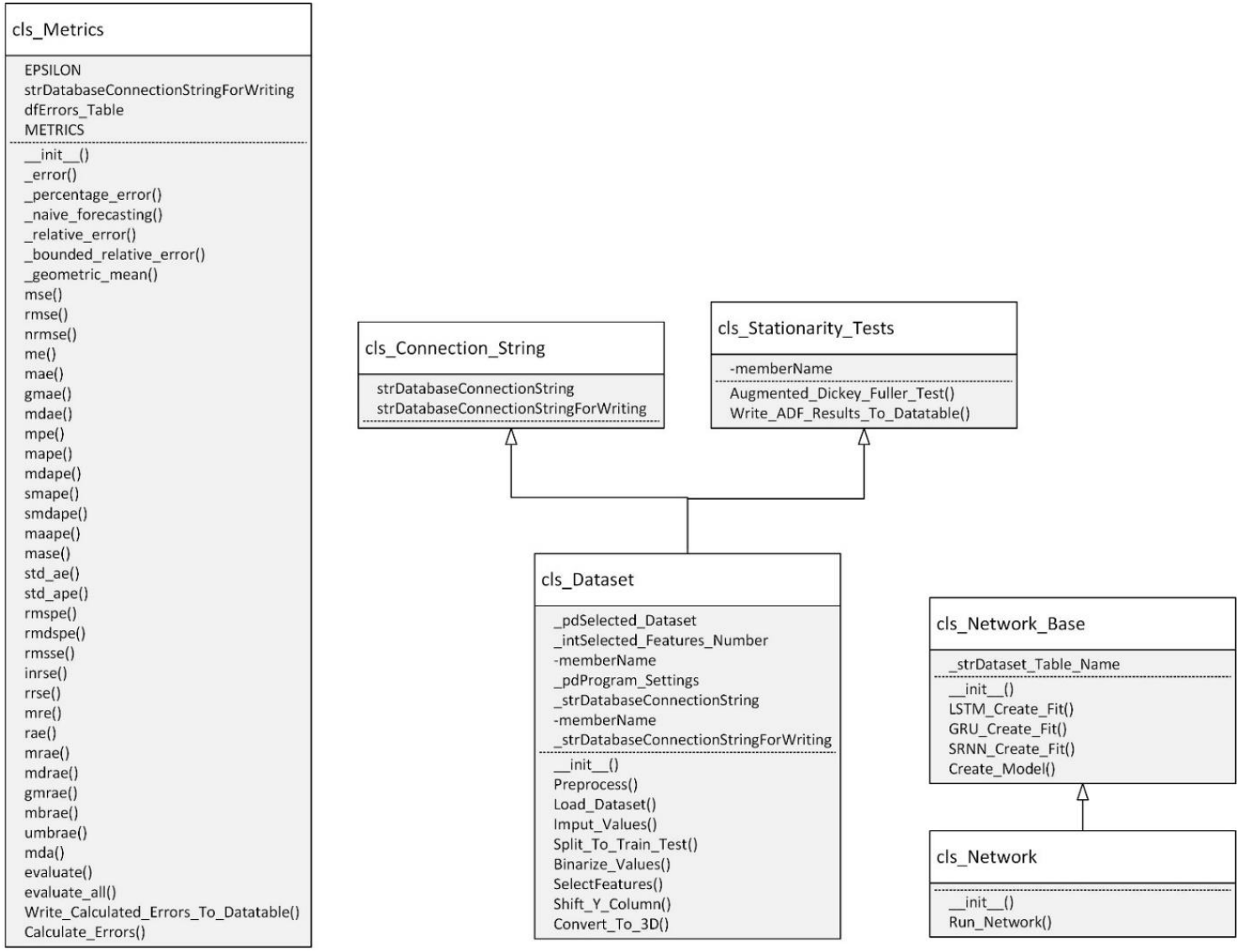
Su neurononinių nuostatomis susijusiai informacijai saugoti skirtų lentelių tarpusavio ryšiai



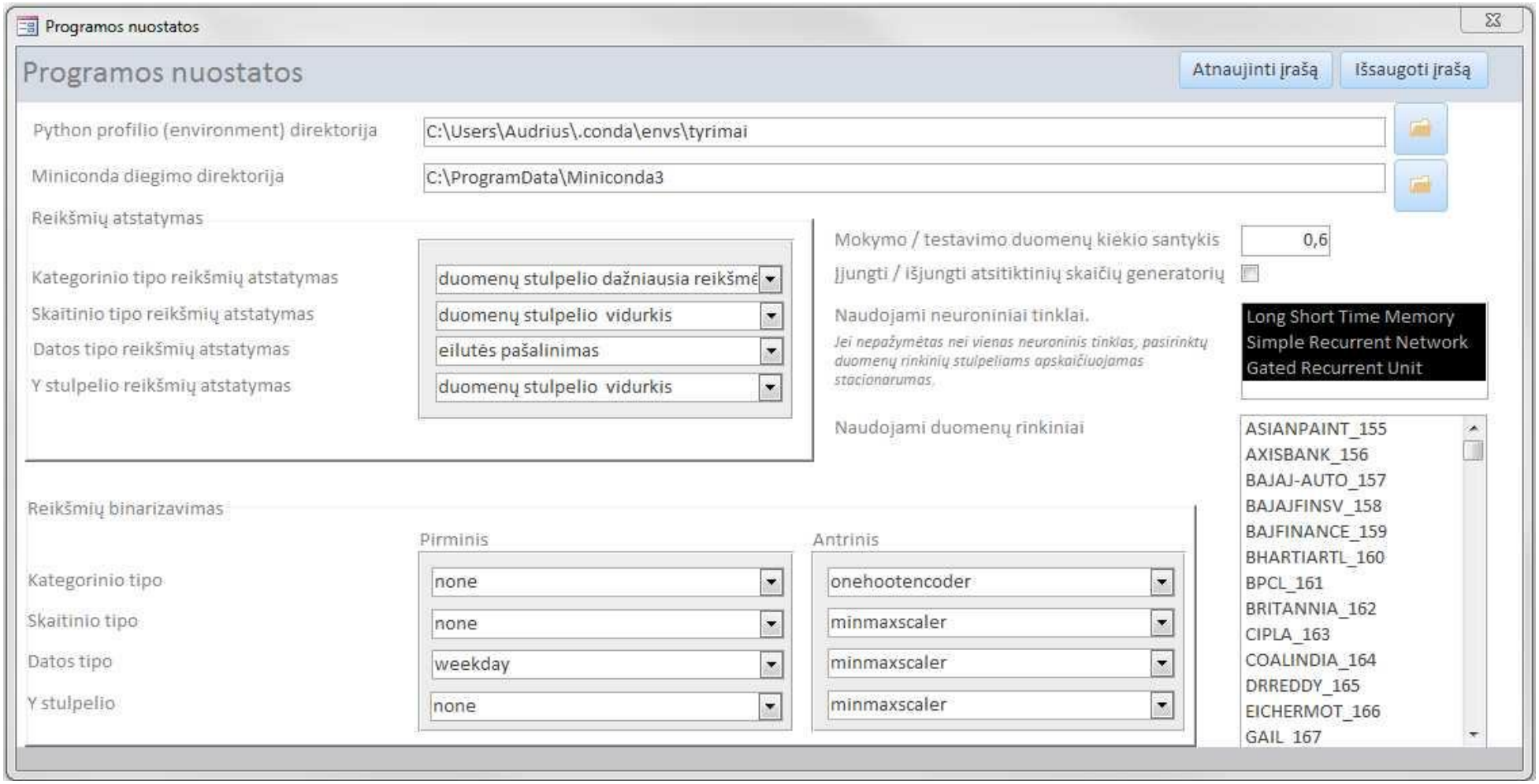
Su duomenų nuskaitymu iš disko susijusių klasių UML klasių diagrama



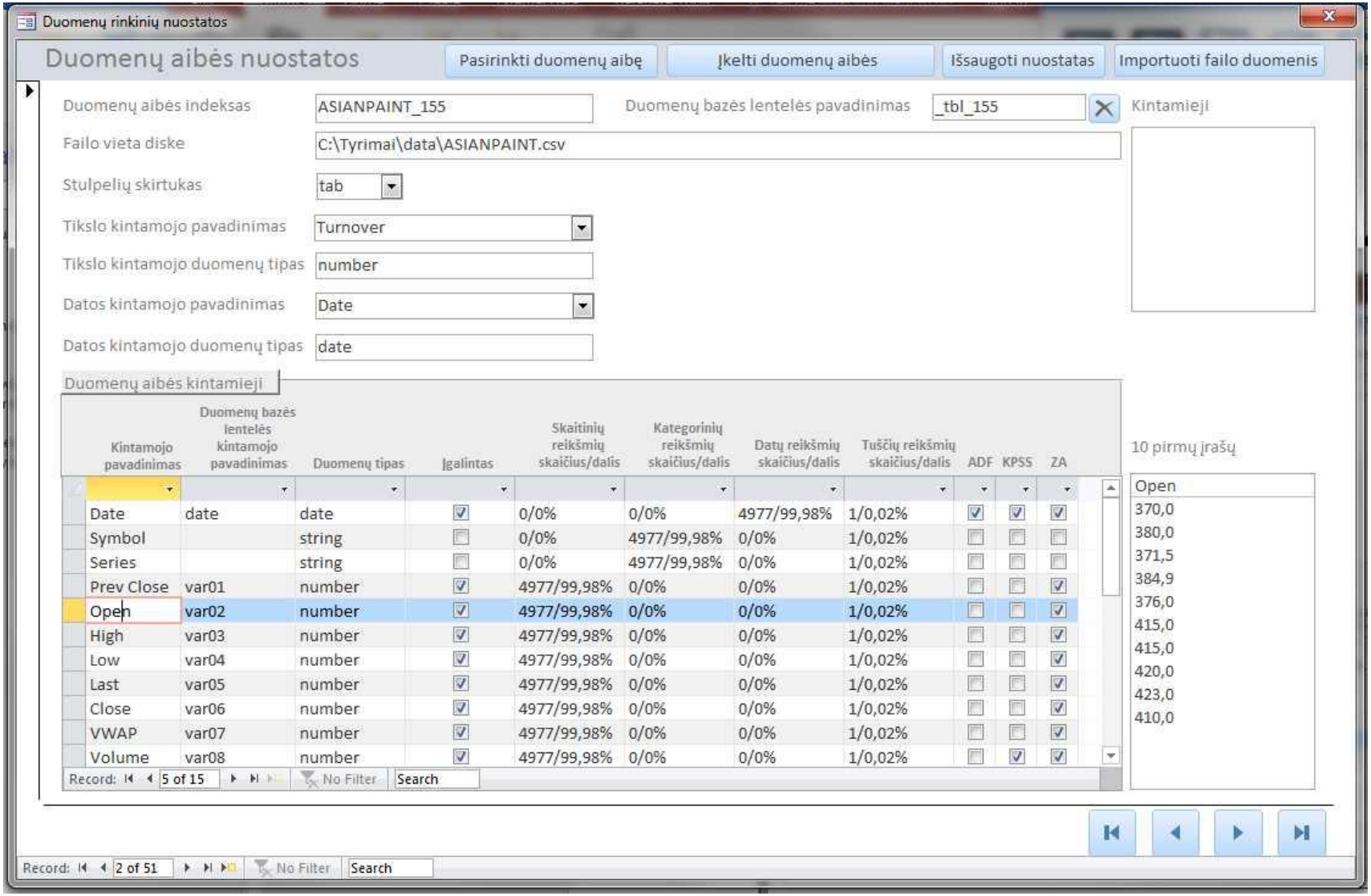
Su metrikomis, duomenų aibėmis ir neuroniniais tinklais susijusių klasių UML klasių diagrama



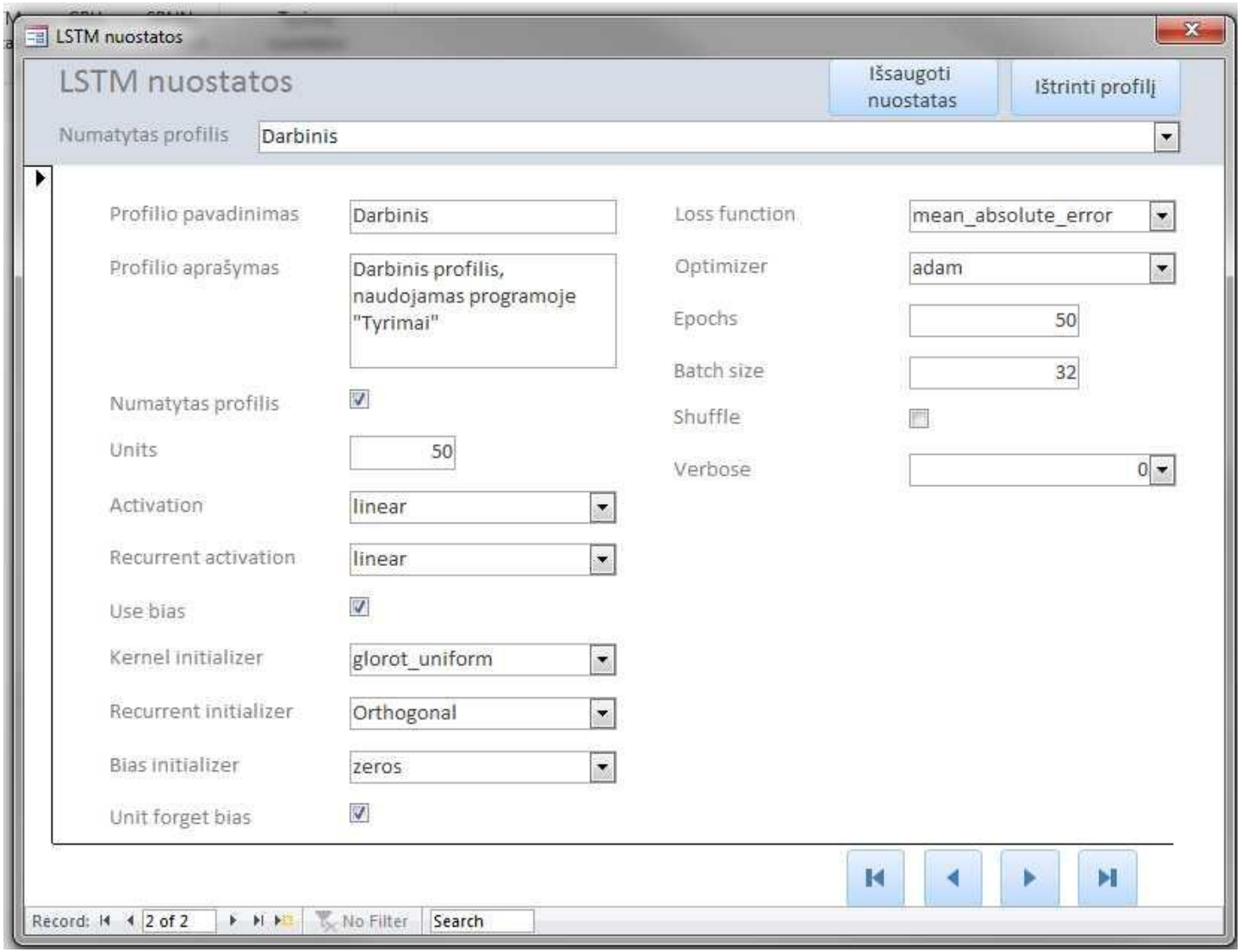
Formos „Programos nuostatos“ vaizdas



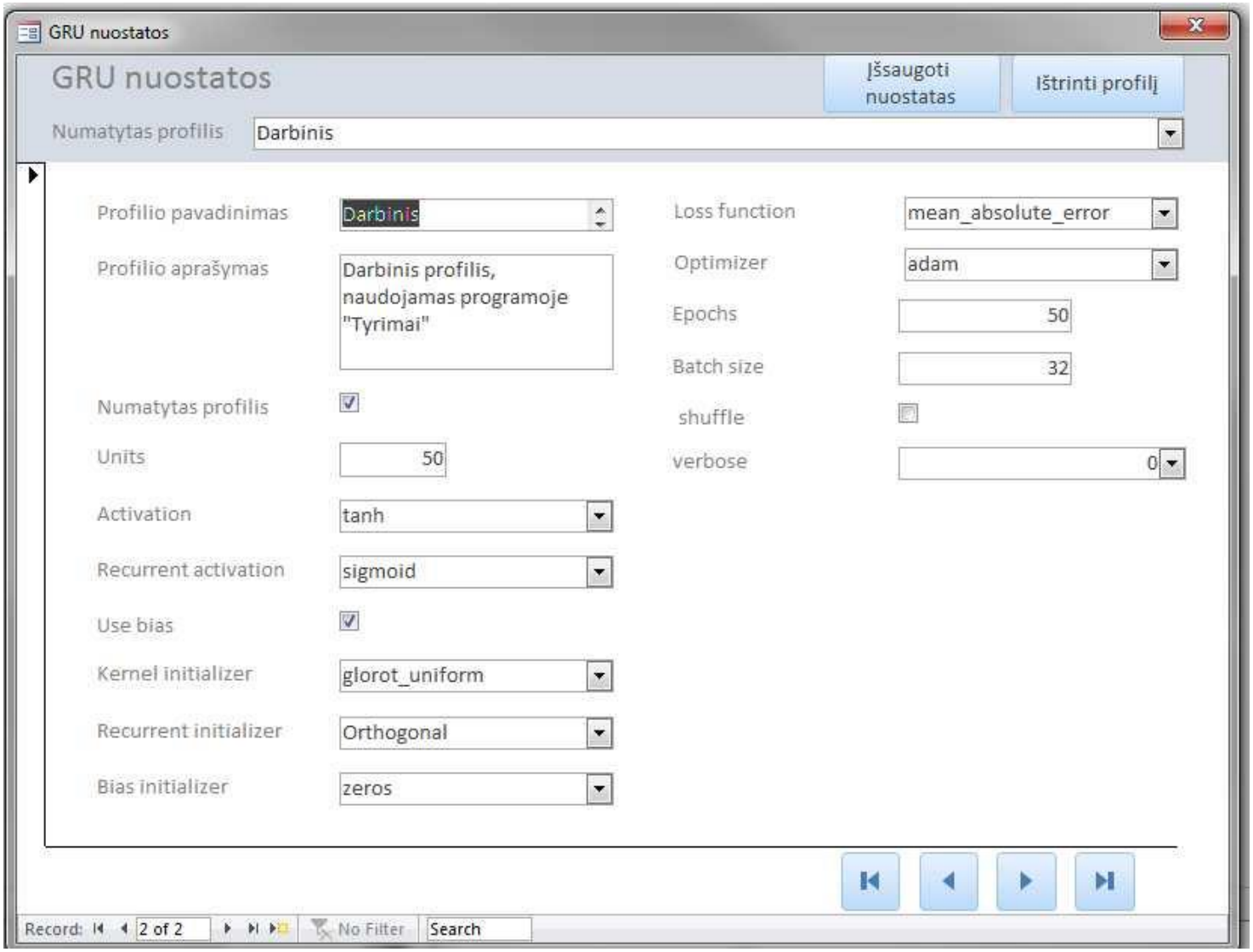
Formos „Duomenų aibių nuostatos“ vaizdas



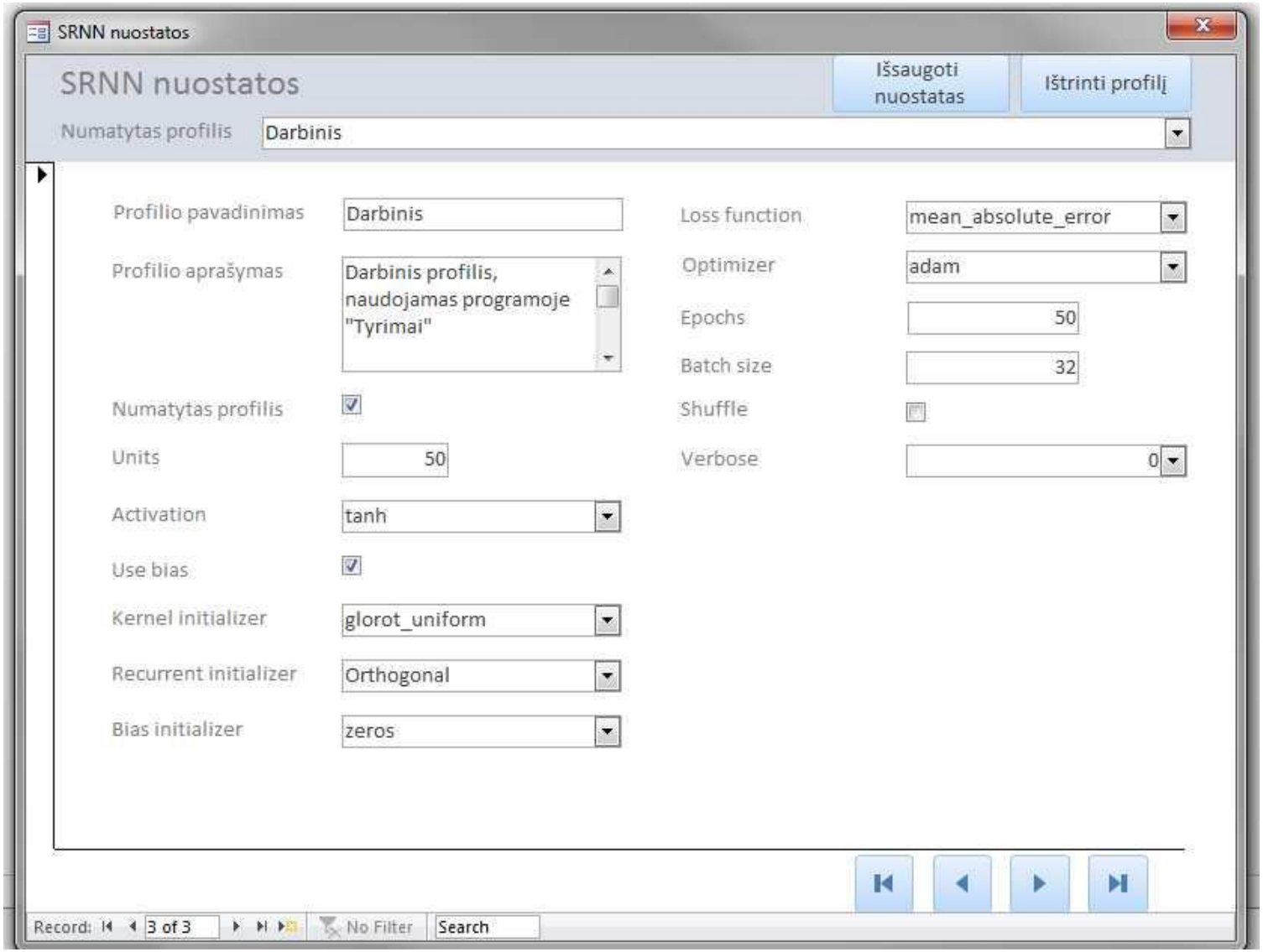
Formos „LTSM nuostatos“ vaizdas



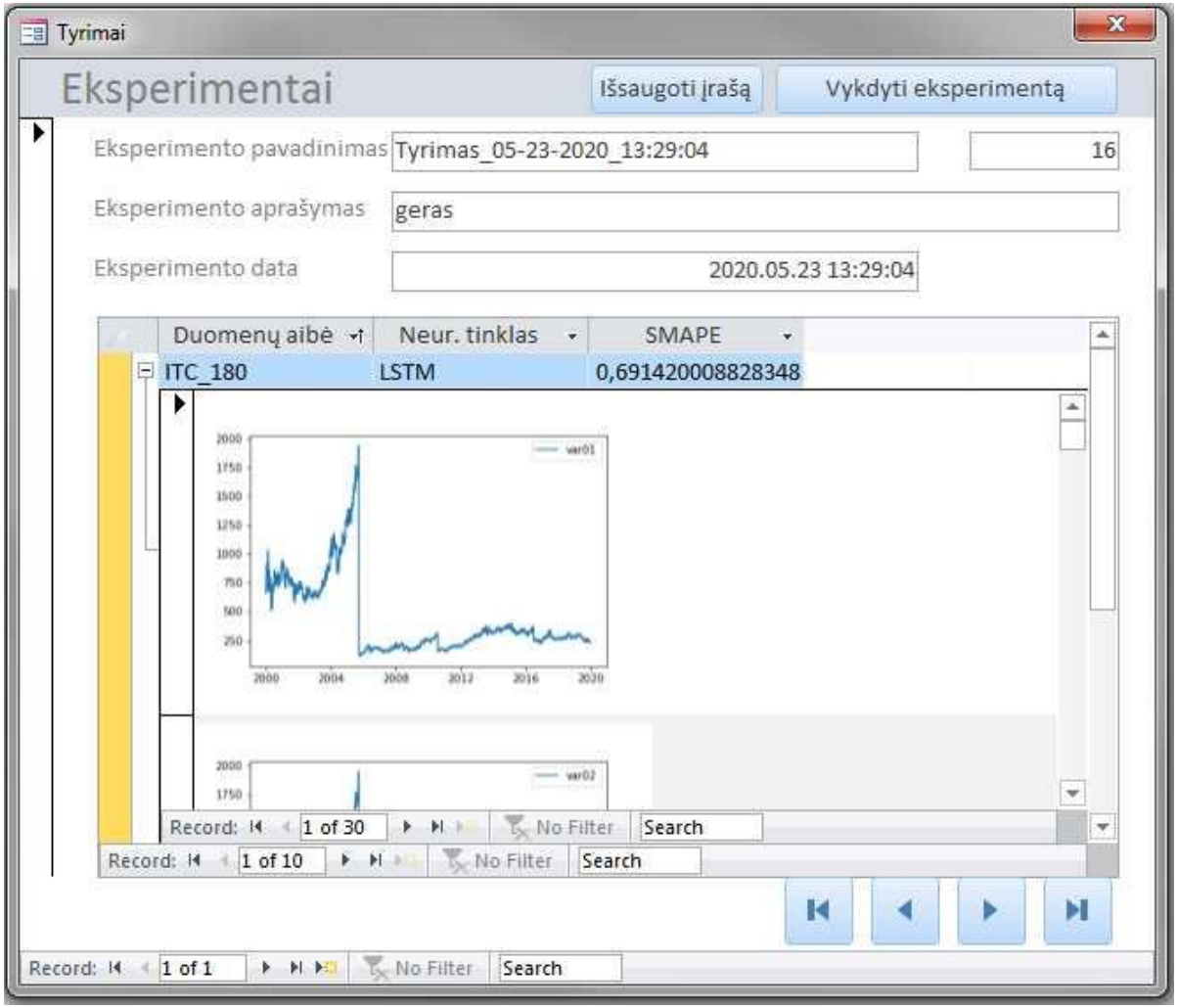
Formos „GRU nuostatos“ vaizdas



Formos „SRNN nuostatos“ vaizdas



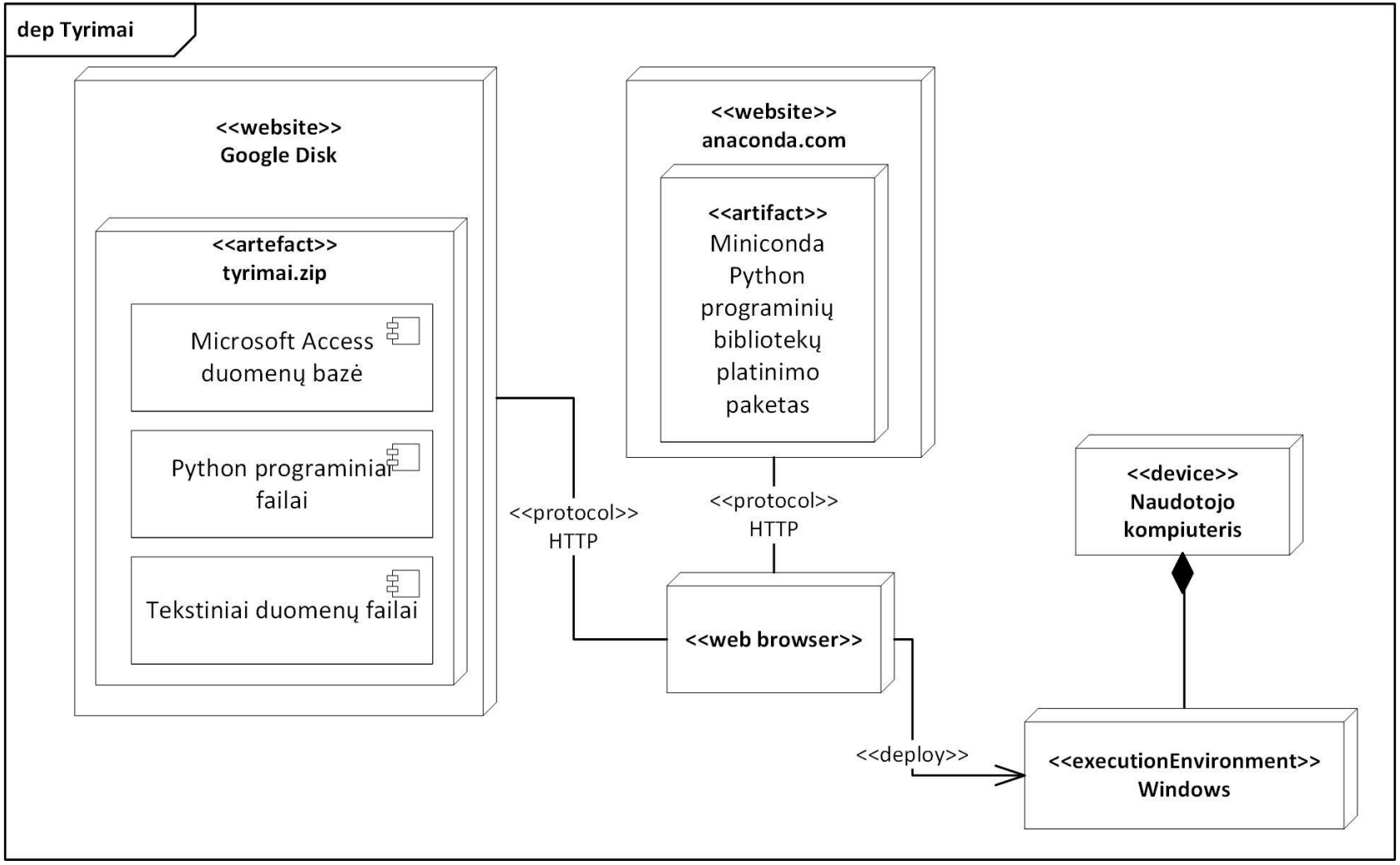
Formos „Eksperimentai“ vaizdas



Programinės įrangos kodas

https://bit.ly/magistrinisdarbas

Programinės įrangos diegimo diagrama



LSTM neuroninio tinklo hiperparametrų derinimas

LSTM hiperparametro pavadinimas	**Pradinė reikšmė**	BAJFINANCE_159	HDFCBANK_345	ICICIBANK_175	INFRATEL_178	IOC_179	ITC_180	SUNPHARMA_193	TITAN_198	ULTRACEMCO_199	ZEEL_203
Units	50	200									
Activation	tanh		relu							relu	
Recurrent activation	hard sigmoid										
Use bias	true	false			false				false		
Kernel initializer	glorot uniform									zeros	
Recurrent initializer	orthogonal	lecun normal		he normal				he normal		variance scaling	
Bias initializer	zeros										
Unit forget bias	true										
Loss function	mean absolute error										mean absolute percentage error
Optimizer	adam	adagrad	adagrad	adagrad	adagrad		adagrad	adagrad			
Epochs	50		200		200	100	200	100	100		100
Batch size	64		20	30	30	10		50	20	20	20
Shuffle	false	true	true	true	true	true	true	true	true	true	true

GRU neuroninio tinklo hiperparametrų derinimas

GRU hiperparametro pavadinimas	**Pradinė reikšmė**	BAJFINANCE_159	HDFCBANK_345	ICICIBANK_175	INFRATEL_178	IOC_179	ITC_180	SUNPHARMA_193	TITAN_198	ULTRACEMCO_199	ZEEL_203
Units	50						100	100			
Activation	tanh	sigmoid	relu		sigmoid						
Recurrent activation	hard sigmoid										
Use bias	true	false						false			false
Kernel initializer	glorot uniform	zeros								zeros	
Recurrent initializer	orthogonal	variance scaling					variance scaling				lecun normal
Bias initializer	zeros										
Unit forget bias	true										
Loss function	mean absolute error										
Optimizer	adam		adagrad	adagrad	adagrad	adagrad		adagrad	adagrad		adagrad
Epochs	50		200	200	200	100			200	200	
Batch size	64		20	10		20	10	10	30	20	20
Shuffle	false		true	true	true		true	true	true		true

SRNN neuroninio tinklo hiperparametrų derinimas

SRNN hiperparametro pavadinimas	**Pradinė reikšmė**	BAJFINANCE_159	HDFCBANK_345	ICICIBANK_175	INFRATEL_178	IOC_179	ITC_180	SUNPHARMA_193	TITAN_198	ULTRACEMCO_199	ZEEL_203
Units	50		200				200	100			
Activation	tanh	softmax				sigmoid		relu		sigmoid	
Recurrent activation	hard sigmoid										
Use bias	true	false			false			false			
Kernel initializer	glorot uniform						zeros		zeros	zeros	zeros
Recurrent initializer	orthogonal		lecun normal			lecun normal	zeros				zeros
Bias initializer	zeros							ones	zeros		
Unit forget bias	true										
Loss function	mean absolute error										mean absolute percentage error
Optimizer	adam	adagrad	adagrad	adagrad	adagrad						adagrad
Epochs	50	200		200	200	100		200			
Batch size	64	50	50	10	30	30	30		10	40	
Shuffle	false		true	true	true		true				true