VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

Robert Dzisevič

TRUMPO TEKSTO KLASIFIKACIJA NAUDOJANT NEURONINIUS TINKLUS

CLASSIFICATION OF SHORT TEXT USING NEURAL NETWORKS

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 6211BX016

Duomenų gavybos technologijos specializacija

Informatikos inžinerijos studijų kryptis

Vilnius, 2019

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių technologijų katedra

Antrosios pakopos studijų **Informacinių technologijų** programos magistro baigiamasis darbas 4

Pavadinimas **Trumpo teksto klasifikacija naudojant neuroninius tinklus**

Autorius **Robert Dzisevič**

Vadovas **Dmitrij Šešok**

Kalba: lietuvių

Anotacija

Šiame darbe yra nagrinėjamas dirbtinio neuroninio tinklo modelio pritaikymas klasifikuojant mažos apimties tekstinius duomenis. Trumpo teksto klasifikacija yra viena iš natūralios kalbos apdorojimo užduočių. Šios užduoties tikslas – tekstą, pagal jo turinį, priskirti vienai iš kelių iš anksto žinomų kategorijų. Užduotį sudaro dvi dalys: tinkamų tekstinių duomenų požymių parinkimas ir klasifikavimo modelio parinkimas. Pirmai problemai išspręsti buvo išnagrinėti trys skirtingi tekstinių duomenų išskyrimo metodai (TF-IDF, TF-IDF LSA, TF-IDF LDA), kurių veiksmingumas buvo patikrintas atliekant lyginąmąjį tyrimą naudojant tą patį neuroninio tinklo klasifikatorių. Tyrimo rezultatai parodė, kad TF-IDF metodu išgautos savybės geriausiai apibūdina tekstinius duomenis. Antrai problemai išspręsti buvo išnagrinėtas neuroninio tinklo modelio tikslumas, palyginus su kitais klasifikavimo modeliais. Visi klasifikavimo modeliai tarpusavyje buvo palyginti atliekant lyginąmąjį tyrimą esant įvairioms sąlygoms: kintant apdorojamų duomenų apimčiai, kintant galimų klasių skaičiui, kintant mokymo ir testavimo duomenų apimties santykiui, apdorojant skirtingų kalbų duomenis. Tyrimo rezultatai parodė, kad neuroninio tinklo klasifikatorius daugumoje atvejų pateikia geriausius rezultatus. Darbą sudaro 8 dalys: įvadas, klasifikacijos problema sistemų mokyme, klasifikavimo modeliai, požymių išskyrimas, požymių išskyrimo metodų palyginimas, klasifikavimo modelių palyginimas, rezultatai ir išvados, literatūros sąrašas.

Darbo apimtis – 50 p. teksto be priedų, 19 ilustr., 5 lentelės, 56 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: Teksto klasifikacija, natūralios kalbos apdorojimas, sistemų mokymas, neuroniniai tinklai, tekstinių požymių išskyrimas.

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Technologies

Master Degree Studies **Information Technologies** study programme Master Graduation Thesis

Title	**Classification of Short Text using Neural Networks**
Author	**Robert Dzisevič**
Academic supervisor	**Dmitrij Šešok**

Thesis language: Lithuanian

Annotation

This thesis analyses the application of an artificial neural network model in classification of short text. Classification of short text is one of many natural language processing tasks. The goal of this task is to assign the text to one of predefined categories based on its content. The task consists of two parts: text feature selection and classification model selection. For the first problem, three different text feature extraction methods were analysed (TF-IDF, TF-IDF LSA, TF-IDF LDA). The mentioned feature extraction methods were compared by measuring their performance using the same neural network classifier. The results show, that the TF-IDF feature extraction method holds the best results in capturing features which can best describe text data. For the second problem, the accuracy of a neural network model was analysed by comparing it with other classification models. During this research, different conditions were applied: when the amount of processing data changes, when the number of possible classes changes, when the amount of training and testing data changes, when text data of different languages is processed. The results show, that the neural network classifier in most cases outperforms other classifiers.

Structure: introduction, classification problem in machine learning, classification models, feature extraction, comparison of different feature extraction methods, comparison of different classification models, conclusions and suggestions, references.

Thesis consist of: 50 p. text without appendixes, 19 pictures, 5 tables, 56 bibliographic entries.

Appendixes included.

Keywords: Text classification, natural language processing, machine learning, neural networks, text feature extraction.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Priedų sąrašas

Santrumpų sąrašas

IT	Informacinės technologijos
NN	Neuroniniai tinklai (angl. *neural networks*)
AVK	Atraminių vektorių klasifikatorius
BK	Bajeso klasifikatorius
SMK	Sprendimų medžio klasifikatorius
K-NN	K-artimiausių kaimynų klasifikatorius (angl. *k-nearest neighbours*)
TF-IDF	TF-IDF metodas (angl. *term frequency inverse document frequency*)
LSA	Latentinė semantinė analizė (angl. *latent semantic analysis*)
LDA	Tiesinė diskriminantinė analizė (angl. *linear discriminant analysis*)
NLP	Natūralios kalbos apdorojimas (angl. *natural language processing*)
NLU	Naturalios kalbos supratimas (angl. *natural language understanding*)

ĮVADAS

Teksto klasifikavimo proceso metu naudojami sistemų mokymosi (angl. *machine learning*) algoritmai. Šie algoritmai mokymosi tikslu naudoja informaciją, gautą iš esamų duomenų ir jų efektyvumas priklauso nuo tų duomenų kiekio. Didesnė duomenų imtis leidžia klasifikavimo algoritmui pavaizduoti atvejį, artimesnį realybei. Anksčiau mokslininkai šiems algoritmams neskyrė daug dėmesio, nes reikalinga informacija nebuvo lengvai pasiekiama arba jos buvo mažai. Dabar visa informacija skelbiama internete, todėl net dideli kiekiai duomenų yra lengvai pasiekiami. Susidarė prielaidos plėstis mokslo šakai tiriančiai dirbtinį intelektą (angl. *artificial inteligence*) ir susikurti naujoms disciplinoms, įskaitant ir natūralios kalbos apdorojimo mokslą (angl. *natural language processing*) (Witten et al., 2017).

Natūralios kalbos apdorojimas – mokslas, tiriantis informacinių technologijų (IT) pritaikymo galimybes visoms natūralios kalbos naudojimo sritims, pavyzdžiui, automatinis teksto taisymas, vertimas iš rašytinio teksto į garsą, vertimas iš garso į rašytinį tekstą, automatinis vertimas, tekstinės informacijos paieška, teksto klasifikavimas ir t. t. Teksto klasifikavimo uždavinio esmė – tekstą pagal jo turinį priskirti vienai iš kelių iš anksto žinomų kategorijų. Teksto klasifikavimas, kaip ir daugelis kitų natūralios kalbos apdorojimo užduočių, reikalauja daug informacijos ir išsamių žinių apie išorinį pasaulį, kad suformuotų suvokimą apie jį. Suvokimo apibrėžimas yra viena iš pagrindinių natūralios kalbos apdorojimo problemų, o teksto klasifikavimas – vienas iš sudėtingiausių uždavinių (Jurafsky & Martin, 2014).

Teksto klasifikavimą sudaro dvi svarbios dalys: klasifikavimo modelio pasirinkimas ir tekstinių požymių, pagal kuriuos bus klasifikuojama, pasirinkimas. Nors klasifikavimo modeliai atlieka pagrindinį darbą – kategorizuoja duomenis, tačiau jie nemoka pasirinkti požymių, pagal kuriuos klasifikuojant, būtų pasiekti geriausi rezultatai. Tekstinių požymių pasirinkimas yra duomenų analitiko uždavinys, kurį analitikas atlieka tarpusavyje lygindamas skirtingus požymių išskyrimo metodus. Požymių išskyrimas šalina duomenų triukšmą, o šiuo būdu gauti požymiai yra artimi teksto konteksto suvokimui, todėl šis žingsnis daro didelę įtaką teksto klasifikavimo modelio tikslumui (Aggarwal & Zhai, 2012).

Egzistuoja nemažai sistemų mokymosi modelių, skirtų tekstui klasifikuoti. Vienas iš populiariausių yra neuroninio tinklo modelis, kurio struktūra ir veikimas primena žmogaus smegenų neuronų sandarą ir veikimo principą. Didžioji dalis neuroninių tinklų teorijos buvo sukurta jau seniai, tačiau anksčiau trūko duomenų ir priemonių geriems rezultatams pasiekti ir tirti šiuos modelius. Dabartinį susidomėjimą neuroninių tinklų modeliais nulėmė susiklosčiusios palankios sąlygos: didžioji dalis informacijos perkelta į pasiekiamą skaitmeninę erdvę, patobulinti kompiuterių procesoriai, kurie sugeba greitai apdoroti didelius kiekius informacijos (Russell & Norvig, 2016). Visa tai leido neuroniniams tinklams įgyti pranašumą palyginti su kitais modeliais sprendžiant

sudėtingus uždavinius: kalbos modeliavimą (Wu et al., 2016), kalbos dalių žymėjimą (Kolesau et al., 2018), antraščių generavimą (Xu et al., 2016), teksto apibendrinimą (Rush et al., 2015), atsakymų į klausimus radimą (Hermann et al., 2015).

Teksto klasifikavimas turi dideles pritaikymo galimybes įvairiose srityse. Interneto puslapių turinio klasifikavimas yra paieškos variklių (pvz.: „Google“, „Yandex“) pagrindas ir leidžia žmonėms greitai rasti tai, ko jie ieško internete. Teksto klasifikavimas gali būti panaudotas reklamuojant žmogui tam tikras prekes, atsižvelgiant į jo naršymo internete tekstinį turinį. Akademikai, teisininkai, socialiniai inžinieriai, valstybės tarnautojai bei kiti ne pelno siekiančių organizacijų atstovai dažnai dirba su nestruktūrizuotais duomenimis, kuriuos suklasifikavus būtų palengvinta informacijos valdymo veikla minėtose organizacijose. Teksto klasifikavimas gali būti pritaikytas sentimentų analizėje tekstą priskiriant žmonių emocijoms. Tai leistų pagal žmogaus pranešimus socialinėje medijoje nustatyti jo psichinę būklę ir greičiau atpažinti asmenis, kuriems reikia psichologinės pagalbos. Be to, tiriant pranešimus ir skelbimus socialinėje medijoje, teksto klasifikavimas gali padėti nustatyti įtartiną, nusikalstamą veiklą arba atpažinti visuomenei pavojingus asmenis ir t. t. (Maren et al., 2014).

Darbo tikslas – sukurti teksto klasifikatorių, naudojant neuroninių tinklų modelį, palyginti jo efektyvumą su kitais klasifikavimo modeliais.

Darbo uždaviniai

Tikslui pasiekti, buvo iškelti šie darbo uždaviniai:

1. Išnagrinėti teksto požymių išskyrimo metodus.
2. Išnagrinėti teksto klasifikavimo modelius.
3. Sukurti teksto klasifikatorių.
4. Atlikti skirtingų teksto požymių išskyrimo metodų lyginamąjį tyrimą.
5. Atlikti skirtingų teksto klasifikavimo modelių lyginamąjį tyrimą.

Tyrimo metodai

Šiame darbe naudojami tyrimo metodai, kurie padėjo atlikti iškeltus uždavinius:

1. Įvairių šaltinių analizė (mokslinė literatūra, straipsniai ir internetas).
2. Tekstui klasifikuoti naudojamų požymių išskyrimo ir klasifikavimo modelių analizė.
3. Įvairių atvirojo kodo projektų („Tensorflow“, „Snowball“) priemonių pritaikymas.

1. KLASIFIKACIJOS PROBLEMA

1.1. Sistemų mokymas

Sistemų mokymas – informatikos mokslo sritis, kuri apima statistinių tyrimų metodus, skaičiavimo mokymų teoriją ir neretai siejama su dirbtinio intelekto kūrimu. Sistemų mokymas nagrinėja algoritmus, kurie geba mokytis iš duomenų ir vėliau daryti prognozes, remiantis tuo, ką išmoko. Šių algoritmų veikimo principas pagrįstas modelio sukūrimu, kai naudojami pavyzdiniai įvesties duomenys, o ne iš anksto apibrėžtos instrukcijos (Alpaydin, 2009).

Sistemų mokymas yra glaudžiai susijęs su statistinių skaičiavimų disciplina, kuri specializuojasi prognozavime. Minėta disciplina apima matematinių skaičiavimų optimizavimą, kuris sistemų mokymosi sritį papildo įvairiais metodais, skirtais spręsti prognozavimo problemas. Sistemų mokymas pritaikomas sprendžiant problemas, kurioms išspręsti yra sunku arba neįmanoma sukurti, suprogramuoti algoritmą, pavyzdžiui, nereikalingų elektroninių laiškų (angl. *spam*) filtravimas (Tretyakov, 2004), optinis ženklų atpažinimas (angl. *optical character recognition*) (Kim et al., 2000), paieškos variklis (angl. *search engine*) (Joachims & Radlinski, 2007), kompiuterio regėjimas (angl. *computer vision*) (Rosten & Drummond, 2006), natūralios kalbos supratimas (angl. *natural language understanding*) (Hu et al., 2014) ir t. t.

Sistemų mokymą pirmą kartą apibrėžė Arthuras Samuelis 1959 m., kuris mokslą apibūdino kaip mokslo sritį, suteikiančią kompiuteriams galimybę mokytis, nereikalaujant nurodyti tikslių instrukcijų (Simon, 2013).

Labiau formalų apibrėžimą pateikė Tom M. Mitchellis: Kompiuterio programa išmoksta iš patirties *E* sprendžiant užduotis *T* ir matuojant sprendimo efektyvumą rodikliu *P*, jei didėjant patirčiai *E* gerėja ir sprendimo efektyvumo rodiklis *P*, sprendžiant užduotis *T* (Mitchell, 1997).

Sistemų mokymo užduotys dažniausiai yra suskirstytos į tris pagrindines kategorijas, atsižvelgiant į mokymo sistemoje esančių mokymo signalų ir grįžtamųjų ryšių (angl. *feedback*) prigimtį:

1. Prižiūrimas mokymas (angl. *supervised learning*) – kompiuteriui pateikiami pavyzdiniai įvesties duomenys ir atitinkami jų išvestys. Tikslas – išmokti bendrą taisyklę, kaip susieti įvestis su išvestimis.
2. Neprižiūrimas mokymas (angl. *unsupervised learning*) – išvestys nepateikiamos mokymo algoritmui. Algoritmas pats nustato galimas išvestis tirdamas įvesties duomenų struktūrą.
3. Sustiprintas mokymas (angl. *reinforced learning*) – kompiuterio programa mokosi sąveikaudama su dinamine aplinka. Algoritmas turi pasiekti kažkokį tikslą nereikalaujant papildomos pagalbos.

1.2. Natūralios kalbos apdorojimas

Natūralios kalbos apdorojimas (angl. *natural language processing*, NLP) – informatikos mokslo sritis, kuri tiria lingvistiką ir nagrinėja kompiuterių gebėjimus suprasti žmonių kalba parašytus tekstus. Šio mokslo siekis – palengvinti žmonių sąveiką su kompiuteriais, suteikiant galimybę žmonėms bendrauti su sistemomis naudojant žmonių kalbą. Natūralios kalbos apdorojimas nagrinėja žmonių kalbą naudojant įvairius skaičiavimo algoritmus, kurie išverčia tekstą žmonių kalba į kompiuteriui suprantamus duomenis (Khurana et al., 2017).

Kadangi dabar daug tekstinių duomenų galima rasti internete, NLP sulaukė nemažai dėmesio dėl savo pritaikomumo įvairiose srityse: teksto vertimas į skirtingas kalbas, informacijos išgavimas, teksto apibendrinimas, atsakymų į klausimus radimas, teksto klasifikavimas ir t. t.

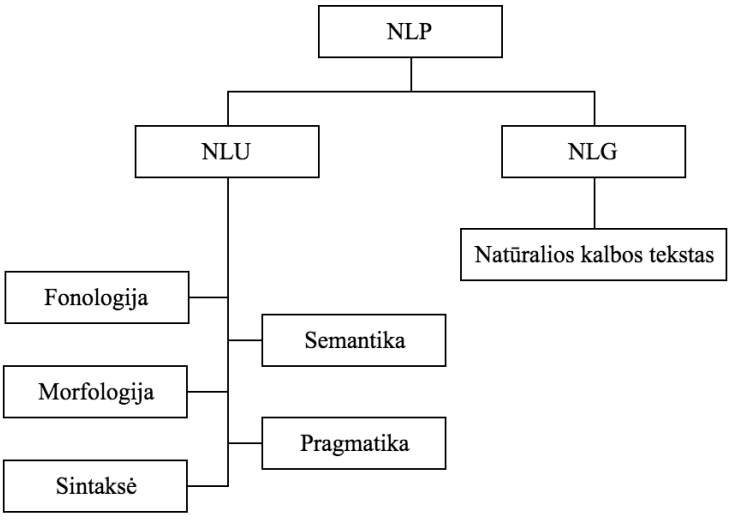
NLP gali būti suskirstytas į dvi dalis:

1. Natūralios kalbos supratimas (angl. *natural language understanding*, NLU).
2. Natūralios kalbos generavimas (angl. *natural language generation*, NLG).

NLU – mokslas, tiriantis žmonių kalbą, kuris susideda iš kelių dalių:

1. Fonologija – nagrinėja kalbos garsus.
2. Morfologija – nagrinėja žodžių sudarymą.
3. Sintaksė – nagrinėja sakinių struktūrą.
4. Semantika – nagrinėja teksto reikšmę.
5. Pragmatika – kaip ir semantika, nagrinėja teksto reikšmę (žr. 1 pav.).

Natūralios kalbos generavimas apima teksto suvokimą ir naujo teksto generavimą.



1 pav. Natūralios kalbos apdorojimo bendras klasifikavimas [22]

1.3. Teksto klasifikacija

Teksto klasifikacija yra viena iš natūralios kalbos apdorojimo problemų, kuri siekia suskirstyti tekstą į atitinkamas kategorijas. Teksto klasifikatoriaus efektyvumą nulemia gebėjimas įsisavinti anksčiau pateiktą informaciją. Viena pagrindinių problemos rūpesčių – mokymo duomenys. Neretai mokymui netinkamai parinkti duomenys iškreipia klasifikatoriaus rezultatus, todėl svarbu parinkti atitinkamus mėginių ėmimo metodus, kad užtikrinti duomenų įvairumą. Dažnai teksto klasifikatorius prastai klasifikuoja dėl klaidingo teksto suvokimo (Singh, 2018). Teksto klasifikacijai taikant sistemų mokymo modelius, susiduriama su šiais iššūkiais:

- Iš teksto išgaunami požymių vektoriai turi įgyti sudėtingą teksto semantiką (Banerjee et al., 2007).
- Išgauti požymių vektoriai turi būti normalizuoti.
- Tekstiniai duomenys dažnai būna įvairaus turinio, kokybės ir apimties (Corey et al., 2010).
- Klasifikatoriaus tikslumas priklauso nuo parinktų mokymo duomenų kokybės (Jiang et al., 2010).
- Tekstinių duomenų požymių kiekis priklauso nuo mokymo duomenų apimties. Svarbu parinkti tinkamą tekstinių požymių išgavimo algoritmą (Sammut & Webb, 2017).

Trumpo teksto klasifikacija – vienas iš teksto klasifikacijos problemos atvejų. Trumpų tekstų gausu internete: produktų atsiliepimai, pokalbių žinutės, naujienų komentarai, pranešimai socialiniuose tinklapiuose, pavyzdžiui, „Facebook“, „Twitter“ ir t. t. (Li et al., 2016).

Trumpo teksto klasifikacija turi papildomų iššūkių:

- Trumpi tekstai yra išsklaidyti duomenys ir labai priklausomi nuo konteksto, kuriame žodžiai vartojami.
- Trumpų tekstų pavyzdžių yra daug, tačiau ne visada jie yra priskirti kuriai nors klasei. Rankinis pavyzdžių priskyrimas klasei užima nemažai laiko, o dalinis priskyrimas netinka klasifikatoriaus modelio mokymui.
- Nestandartinių terminų egzistavimas kalboje: klaidinga rašyba, gramatinės klaidos, santrumpos, žargonai.

2. KLASIFIKAVIMO MODELIAI

Sistemų mokymo mokslas apima nemažai mokymo algoritmų, todėl šiame darbe bus plačiau nagrinėjami penki mokymo modeliai, kurie yra dažnai taikomi klasifikavimo uždaviniams spręsti:

1. Neuroninio tinklo modelis (daugiasluoksnis perceptronas).
2. Atraminių vektorių klasifikatorius.
3. Bajeso klasifikatorius.
4. Sprendimų medžio klasifikatorius.
5. K-artimiausių kaimynų klasifikatorius.

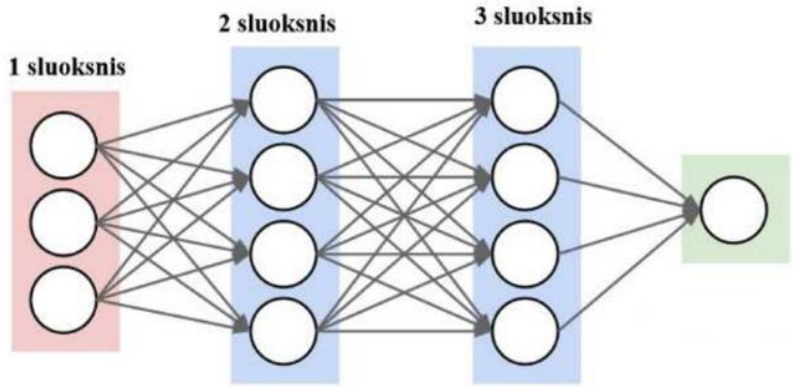
2.1. Neuroninio tinklo modelis

Dirbtiniai neuroniniai tinklai (angl. *artificial neural networks*) – informacijos apdorojimo sistema, kurios veikimo principas siejamas su žmogaus smegenyse esančių biologinių neuronų veikimo principu. Neuroninių tinklų teorijos užuomazgos prasidėjo 1943 m. kai neurofiziologas W. McCulloch ir matematikas W. Pitts paruošė straipsnį, kuris apibūdino, kaip galima sumodeliuoti žmogaus smegenų neurono veikimo principą naudojant elektros grandinę (McCulloch & Pitts, 1943). Dirbtiniai neuroniniai tinklai yra ypatingi tuo, kad jie, kaip ir žmogaus smegenys, turi galimybę mokytis. Tai reiškia, kad turint duomenų ir taikant mokymo algoritmus, neuroninis tinklas priderinamas prie duomenų struktūros taip, kad tinklas išmoksta atpažinti naujus duomenis, kurių anksčiau nematė.

Nors dirbtinių neuroninių tinklų teorija atrodė daug žadanti, ši skaičiavimo sistema iš pradžių susilaukė mažai dėmesio. Pirmi neuroniniai tinklai demonstravo prastus rezultatus, nes jie buvo paprasti, sudaryti iš kelių neuronų. Gerus rezultatus galima buvo pasiekti modeliuojant sudėtingus tinklus, susidedančius iš daugybės neuronų, tačiau tokie tinklai reikalauja daug skaičiavimo resursų, kurių buvo sunku išgauti naudojantis tuometinėmis technologijomis. Praėjus keliems dešimtmečiams skaičiavimo technologijos patobulėjo ir mokslininkai vėl ėmė domėtis neuroniniais tinklais. Susidarė palankios sąlygos toliau vystyti technologiją ir galiausiai neuroninių tinklų modeliai pradėjo rodyti itin gerus rezultatus sprendžiant įvairaus pobūdžio uždavinius: teksto klasifikavimas, vaizdų klasifikavimas, kalbų vertimas, balso vertimas į tekstą (ir atvirkščiai) ir t. t. (Beam, 2017). Galiausiai vienas iš įspūdingiausių neuroninių tinklų pasiekimų – profesionalaus žaidėjo (žmogaus) įveikimas žaidžiant stalo žaidimą „Go“ (Silver, et al., 2017).

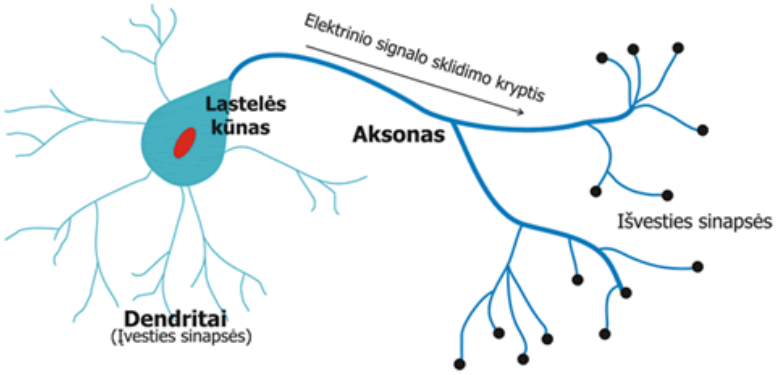
Neuroninio tinklo sandara

Dirbtinį neuroninį tinklą sudaro sluoksniai neuronų, kur vieno sluoksnio neuronai yra sujunti su prieš arba po jo esančio sluoksnio neuronais (žr. 2 pav.). Sluoksnių ir neuronų, esančių kiekviename sluoksnyje, kiekis priklauso nuo sprendžiamo uždavinio tipo ir sudėtingumo.



2 pav. Dirbtinis neuroninis tinklas [10]

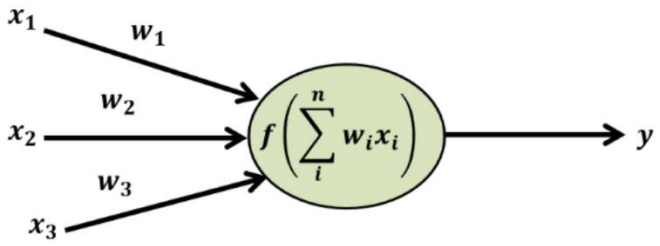
Tinklo neuronas – dirbtinė žmogaus biologinio neurono abstrakcija ir pagrindinis neuroninio tinklo komponentas. Dirbtinis neuronas (kaip ir biologinis neuronas) turi keletą įėjimų (dendritų), branduolį, kuriame nustatyta aktyvacijos funkcija ir vieną arba daugiau išvesčių (išvesties sinapsių) (žr. 3 pav.) (Berniukevičius & Kurilovas, 2017).



3 pav. Nervinė ląstelė – neuronas [30]

Dirbtinio neuroninio tinklo neuronas turi kelias ar daugiau įvesčių x_i ir kiekviena įvestis turi atitinkamą svorį w_i, kur $i = 1, 2, \ldots, n$ (žr. 4 pav.). Dažniausiai įvesčių ir svorių reikšmės – realieji skaičiai. Tinklo neuronas skaičiuoja pasvertąją sumą, neurono sužadinimo signalą a:

$$a = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \sum_{i=1}^{n} w_ix_i.$$



4 pav. Dirbtinis neuronas [6]

Radus neurono sužadinimo signalą, panaudojama aktyvacijos funkcija:

$$y = f(a) = f(\sum_{i=1}^{n} x_i x_i).$$

Aktyvacijos funkcijų yra kelios:

1. Slenkstinė:

$$f(a) = \begin{cases} 1, & jei\ a \geq w_0 \\ 0, & jei\ a < w_0 \end{cases}.$$

2. Loginis sigmoidas:

$$f(a) = \frac{1}{1 + e^{-a}}.$$

3. Tangento sigmoidas:

$$f(a) = \frac{2}{1 + e^{-2a}} - 1.$$

4. Gauso:

$$f(a) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}.$$

5. „Softmax“ funkcija:

$$f(a)_j = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}.$$

6. „ReLu“ funkcija:

$$f(a) = \max(0, a).$$

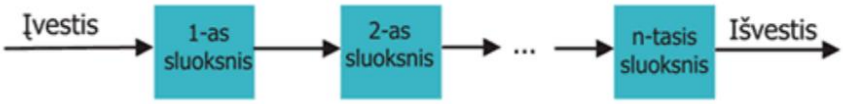
Tinklo modelis

Neuroninis tinklas gaunamas, kai sujungiamas vienas dirbtinis neuronas su kitu. Kiekvienas tinklas turi įėjimus, kurie perduoda kintamųjų reikšmes ir išėjimus, kurie formuoja tinklo išvesties reikšmę. Įėjimai atitinka sensorinius nervus (akių, ausų), o išėjimai – motorinius nervus (rankų, kojų). Tarp įėjimo ir išėjimo neuronų dažnai pasitaiko tarpinių, paslėptųjų (angl. *hidden*) neuronų, kurie taip pat padeda skaičiuoti funkcijų vertes. Įėjimo, tarpiniai ir išėjimo neuronai apjungiami tarpusavyje taip, kad vieno neurono išėjimo reikšmės tampa kitų neuronų įėjimo reikšmėmis (žr. 2 pav.) (Logan et al., 2017).

Pagal neuronų apjungimo būdą, neuroniniai tinklai skirstomi į dvi grupes:

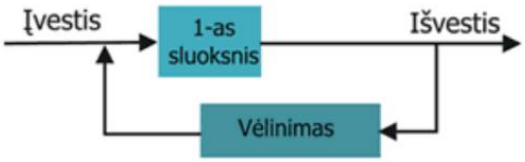
1. Tiesioginio sklidimo tinklai.
2. Grįžtamojo ryšio (rekurentiniai) tinklai.

Tiesioginio sklidimo (dar vadinami vienkrypčio ryšio) neuroniniuose tinkluose vieno sluoksnio išėjimai gali jungtis tik su kito sluoksnio įėjimais. Jokie kiti ryšiai su to paties sluoksnio išvestimis ar prieš tai einančio sluoksnio įvestimis neegzistuoja. Paskutinio neuronų sluoksnio išvestis yra tuo pačiu ir tinklo išvestis (žr. 5 pav.) (Glorot & Bengio, 2010).



5 pav. Vienkrypčio ryšio neuroninis tinklas [30]

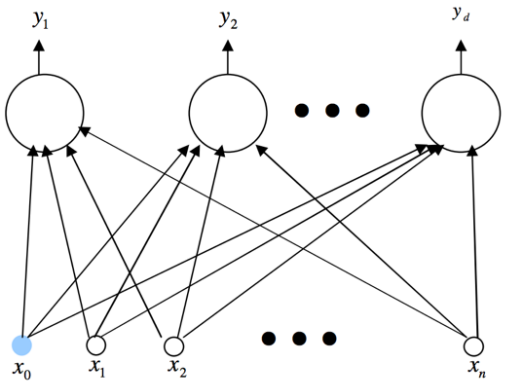
Grįžtamojo ryšio neuroninio tinklo įvestį sudaro išorinės įvestys ir to paties tinklo išvestis, kuriam būdingas vėlinimas (žr. 6 pav.) (Pascanu et al., 2013).



6 pav. Grįžtamojo ryšio neuroninis tinklas [30]

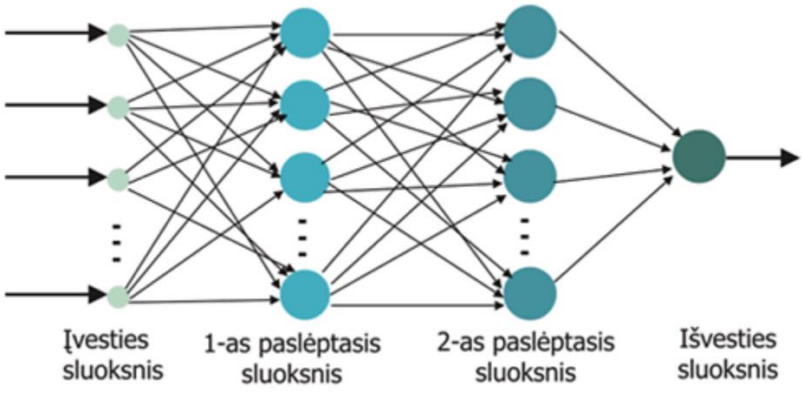
Perceptronas

Viena iš paprasčiausių ir dažnai naudojamų neuroninių tinklų variacijų yra perceptronas (angl. *perceptron*). Perceptronas – tiesioginio sklidimo (angl. *feedforward*) neuroninis tinklas, kur galimos tik vienos krypties jungtys iš įėjimų į išėjimus ir pats tinklas sudarytas iš vieno sluoksnio *K* neuronų su *n* įėjimais (žr. 7 pav.) (Raudys, 1998).



7 pav. Vienasluoksnis perceptronas [50]

Vienasluoksniai neuroniniai tinklai yra paprasti, tačiau netinkami sudėtingoms užduotims spręsti. Tokiu atveju yra naudojami daugiasluoksniai perceptronai, kurie yra sudaryti iš įėjimų aibės, išėjimų neurono sluoksnio ir paslėptųjų neuronų sluoksnių.



8 pav. Daugiasluoksnis perceptronas [30]

Daugiasluoksnio perceptrono struktūra – vienas po kito einantys neuronų sluoksniai, kur kiekvienas neuronas sujungtas su visais kitais kito sluoksnio neuronais (žr. 8 pav.) (Ramchoun et al., 2016).

Neuroninio tinklo mokymas

Egzistuoja trys neuroninių tinklų mokymo tipai:

1. **Mokymas su mokytoju**. Reikalingas išorinis mokytojas, kuris prižiūrėtų mokymosi procesą. Mokytoju gali būti tinklo mokymui skirtas duomenų rinkinys arba stebėtojas, kuris vertina tinklo našumą. Prižiūrimam mokymui (angl. *supervised learning*)

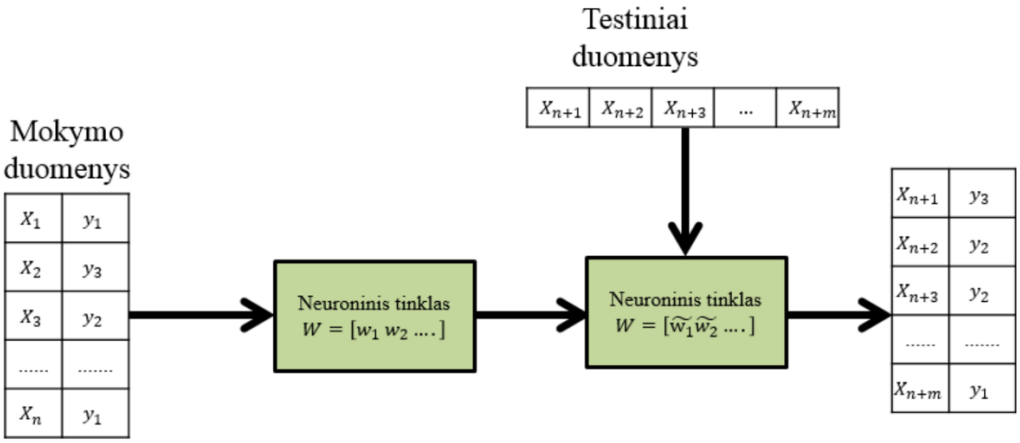
priskiriami šie algoritmai: atgalinio skleidimo, mažiausio kvadratinio vidurkio, radialinės bazės funkcijos algoritmas. Šio mokymo tikslas – priversti dirbtinį neuroninį tinklą pakeisti neuronų jungčių svorius atsižvelgiant į pavyzdines įvestis ir išvestis. Mokymas baigiamas, kai tinklas išmoksta su minimalia paklaida sieti įvestis su išvestimis (Hackeling, 2017).

2. **Mokymas be mokytojo**. Atsižvelgiant į kriterijus ir tinklo informaciją, neuroninis tinklas pats save suderina. Gaunami tik įvesčių pavyzdžiai, kuriuos tinklas turi pats pagal požymius jas suklasifikuoti.
3. **Hibridinis mokymas**. Apjungia mokymą su mokytoju ir be mokytojo būdus. Dalis tinklo svorių nustatoma su mokytojo pagalba, kita dalis – be mokytojo.

Dirbtinio neuroninio tinklo mokymas – tai neurono įvesties svorių w_i keitimo uždavinys. Sprendžiant klasifikavimo uždavinį, neuroninis tinklas veikia tokiu principu. Turime elementą, kurį norime priskirti kuriai nors kategorijai. Elementas yra aprašomas vektoriumi:

$$X = [x_1, x_2, \dots, x_n].$$

Aibė $Y = [y_1, y_2, \dots, y_m]$ aprašo visas galimas klases y_j, kurioms gali priklausyti elementai X. Neuroninio tinklo mokymo metu yra panaudojami mokymo duomenų rinkinio elementai su jiems priskirtomis kategorijomis – (X_i, y_j). Tada, pasinaudojant optimizavimo algoritmais, parenkami neuroninio tinklo neuronų svoriai (koeficientai) w_i, taip, kad neuroninis tinklas atvaizduotų mokymo elementus X į klasę y, kurioms jie ir priklauso.



9 pav. Neuroninio tinklo mokymo ir testavimo fazės [6]

Neuroninio tinklo klasifikatoriaus testavimo metu, naudojama kita aibė duomenų porų (X_i, y_j), kurių tinklas dar niekada nematė ir skaičiuojama, kiek kartų tinklas teisingai klasifikuoja elementus. Gauta metrika leidžia įvertinti klasifikatoriaus tikslumą. Jei klasifikatoriaus tikslumas mus netenkina – koreguojami neuroninio tinklo koeficientų w_i parinkimo būdai ir tinklas mokamas iš naujo tol, kol bus pasiektas pageidaujamas klasifikatoriaus tikslumas (žr. 9 pav.) (Berniukevičius et al., 2017).

„Klaidos sklidimo atgal“ algoritmas

Neuroninio tinklo mokymas grindžiamas klaidos funkcijos parinkimu ir jos minimizavimu atsižvelgiant į tinklo svorius. Problemos, su kuriomis susiduriama naudojant daugiasluoksnį neuroninį tinklą:

1. Nėra akivaizdaus būdo nustatyti paslėptų neuronų išvesčių pageidaujamas reikšmes.
2. Jei neuroninio tinklo išvestis pateikia neteisingą rezultatą, nėra būdų nustatyti, kuris paslėptas neuronas atsakingas už šį rezultatą, t. y., nežinome, kurį svorį ir kiek jį reikia keisti.

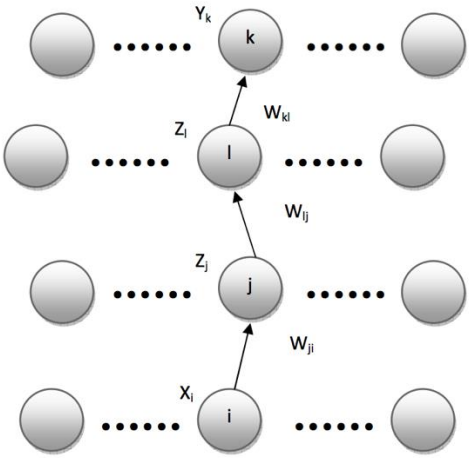
Kas žinoma apie neuroninį tinklą:

1. Jei tinklo perdavimo funkcijos diferencijuojamos, tuomet tinklo išėjimo sluoksnio neuronų perdavimo funkcijos diferencijuojamos įėjimo kintamųjų x_i ir svorių w_i atžvilgiu.
2. Jei parenkama klaidos funkcija E diferencijuoja neuroninio tinklo išėjimo atžvilgiu, tuomet ši funkcija bus diferencijuojama ir įėjimo kintamųjų, svorių atžvilgiu.
3. Tuomet klaidos išvestines galima skaičiuoti svorių atžvilgiu ir naudoti išvestines, minimizuojančias klaidos funkciją, rasti reikalingus svorius gradientiniais ar kitais metodais.

„Klaidos skleidimo atgal“ algoritmas yra ypatingas tuo, kad jis paskirsto klaidą tarp tinklo neuronų nuo pirmo iki paskutinio sluoksnio. Minėtame metode naudojama mokymo su mokytoju strategija. Šį algoritmą sudaro du žingsniai:

1. Įėjimų reikšmių „skleidimas pirmyn“ iš įvesties į išvesties sluoksnį.
2. Paklaidos „skleidimas atgal“ iš išvesties į įvesties sluoksnį.

„Klaidos skleidimo atgal“ algoritmui paaiškinti kaip pavyzdys bus naudojamas daugiasluoksnio neuroninio tinklo fragmentas pateiktas 10 paveiksle.



10 pav. Daugiasluoksnio neuroninio tinklo fragmentas [45]

Tiesioginio sklidimo metu kiekvienas neuronas skaičiuoja svorinę įėjimo elementų sumą. Gauta suma transformuojama į netiesinę perdavimo funkciją:

1. Įėjimo kintamieji. Suma: $a_j = \sum_i w_{ji} x_i$. Funkcija: $z_j = f(a_j)$.
2. Paslėpti sluoksniai: Suma: $a_l = \sum_j w_{lj} z_j$. Funkcija: $z_l = f(a_l)$.
3. Tinklo išvestis: Suma: $a_k = \sum_l w_{kl} z_l$. Funkcija: $y_k = f(a_k)$.

Čia: a_j, a_l, a_k – visų neuronų išėjimų svorinė suma; x_i, z_j, z_l – įėjimo kintamieji; w_{ji}, w_{lj}, w_{kl} – svoriai tarp nagrinėjamo neuronų sluoksnio ir žemesnio sluoksnio neuronų elementų; z_j, z_l, y_k – neuronų išvesties reikšmės.

Dirbtinio neuroninio tinklo mokymo metu stengiamasi minimizuoti klaidą, kuri apskaičiuojama pagal suminę kvadratinę klaidą. Klaida E^Spriklauso nuo svorio w_{ji} per svorinę įėjimų į j-ąjį elementą sumą a_j. Dalinė klaidos išvestinė pagal svorį w_{ji} apibrėžiama taip:

$$\frac{\partial E^S}{\partial w_{ji}} = \frac{\partial E^S}{\partial a_j} \times \frac{\partial a_j}{\partial w_{ji}}.$$

Pažymėjus:

$$\delta_j = \frac{\partial E^S}{\partial a_j}, \text{ iš } a_j \text{ formulės turime: } \frac{\partial a_j}{\partial w_{ji}} = \frac{\partial (\sum_i w_{ji} z_i)}{\partial w_{ji}} = z_i,$$

todėl:

$$\frac{\partial E^S}{\partial w_{ji}} = \delta_j z_i.$$

Jei nagrinėjamas pirmas tinklo sluoksnis, į kurio įėjimą paduodamas įėjimo reikšmių vektorius, tuomet:

$$\frac{\partial E^S}{\partial w_{ji}} = \delta_j x_i,$$

kur δ – tam tikros rūšies išėjimo klaida.

Išėjimo klaidos išėjimo mazgams δ_k ir paslėptų sluoksnių neuronams δ_j apskaičiuojami sekančiu principu. Žinodami, kad $y_k = f(a_k)$ pagal formulę apskaičiuojama klaida išėjimo mazgams:

$$\delta_k = \frac{\partial E^S}{\partial a_k} = \sum_k \frac{\partial E^S}{\partial y_k} \times \frac{\partial y_k}{\partial a_k} = f'(a_k) \frac{\partial E^S}{\partial y_k},$$

o paslėptiems mazgams:

$$\delta_j = \frac{\partial E^S}{\partial a_j} = \sum_k \frac{\partial E^S}{\partial a_k} \times \frac{\partial a_k}{\partial a_j}.$$

Įsistačius $a_k = \sum_j f(a_j) w_{ik}$ ir $\frac{\partial a_k}{\partial a_j} = f'(a_j) w_{kj}$ į prieš tai pateiktą formulę (δ_j) gauname:

$$\delta_j = \sum_k \delta_k f'(a_j) w_{kj} = f'(a_j) \sum_k \delta_k w_{kj},$$

kur s – sekančio sluoksnio neuronas.

Turint visas išvestines, bendra klaida apskaičiuojama pagal formulę:

$$\frac{\partial E}{\partial w_{ji}} = \sum_s \frac{\partial E^S}{\partial w_{ji}}.$$

Galiausiai belieka atnaujinti neuroninio tinklo svorius. Svorio koeficientai gali būti keičiami dviem būdais:

1. Po vieno duomenų vektoriaus pateikimo tinklui:

$$w_{ji}^{naujas} = w_{ji}^{senas} - \eta \frac{\partial E^S}{\partial w_{ji}} = w_{ji}^{senas} - \eta \delta_j z_i.$$

2. Po visos duomenų imties pateikimo tinklui:

$$w_{ji}^{naujas} = w_{ji}^{senas} - \eta \frac{\partial E}{\partial w_{ji}} = w_{ji}^{senas} - \eta \delta_j^s z_i^s.$$

Čia η – koeficientas priklausantis intervalui [0;1] (Nielsen, 2018).

2.2. Atraminių vektorių klasifikatorius

Atraminių vektorių klasifikatorius (angl. *Support Vector Classifier,* SVC) – sistemų mokymosi algoritmas, kurį išrado V. Vapnikas ir A. Červonenkis praeito amžiaus 7-ajame dešimtmetyje ir ištobulino iki galutinio varianto 1995 m. (Cortes & Vapnik, 1995). Šis prižiūrimo mokymosi metodas išpopuliarėjo dėl galimybės spręsti sudėtingus uždavinius (klasifikavimo, objektų atpažinimo, regresijos uždavinių sprendimo) įvairiose srityse (klasifikuojant dokumentus, atpažinant vaizdus ir kt.). Kitas šio algoritmo privalumas yra jo greitaveika sprendžiant sudėtingus uždavinius lyginant su dauguma kitų sistemų mokymo modeliais.

Atraminių vektorių klasifikatoriaus veikimo principas – požymių erdvėje rasti hiperplokštumą, kuri skirtų skirtingoms klasėms priklausančius taškus. Priklausomai nuo sprendžiamo uždavinio sudėtingumo, klasių skaičius gali būti neribotas. Įvertinus klases apibrėžiančių duomenų pasiskirstymą erdvėje pasirenkama lygtis aprašanti hiperplokštumą. Paprasčiausia hiperplokštumos lygties forma yra tiesinė funkcija, kuri yra taikoma, kai duomenys priklauso tik dviems skirtingoms klasėms, o patys duomenys nėra tarpusavyje persikeitę (žr. 11 pav.).

Šis modelis gali būti naudojamas klasifikuoti duomenis ne tik į dvi klases, bet gali būti išplėstas iki *m* klasių. Tuomet atitinkamai reikia *m* hiperplokštumų norint padalinti ir paruošti vektorinę erdvę elementų klasifikavimui. Atraminių vektorių klasifikatoriaus matematinis modelis: mokymo duomenys apibrėžiami kaip $(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)$, kur $x_i \in \Re^n$, $y_i \in \{-1, +1\}, i \in [1, \ldots, l]$. Klasifikavimo funkcijos apibrėžimas:

$$g(x) = sign\big(f(x)\big)$$

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b,$$

kur *K* – branduolio funkcija; $b \in \Re$ – slenkstis; α_i – svoriai, kurie tenkina sąlygas:

$$\forall i: 0 \leq \alpha_i \leq E,$$

$$\sum_{i=1}^{l} \alpha_i \, y_i = 0,$$

kur *E* – neteisingo klasifikavimo kaina. Vektoriai x_i su nenuliniu α_i – pagalbiniai vektoriai. Tiesinės atraminių vektorių klasifikatoriaus branduolio funkcijos *K* apibrėžimas:

$$K(x_i, x) = x_i \times x,$$

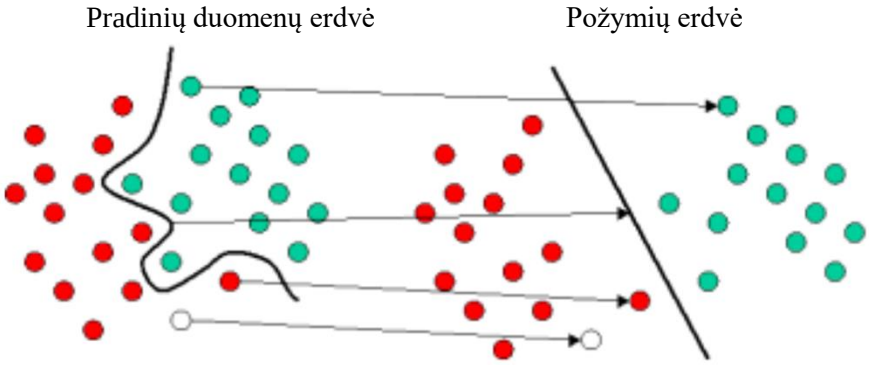
kas leidžia funkciją $f(x)$ perrašyti tokiu pavidalu:

$$f(x) = w \times x + b, \text{ kur } w = \textstyle\sum_{i=1}^{l} y_i \alpha_i x_i.$$

Modelio mokymas reiškia α_i ir b radimą, tuo pačiu išsprendžiant šią optimizacijos problemą:

$$\begin{cases} \max(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \, \alpha_j y_i y_j K(x_i, y_j)); \\ \forall i: 0 \leq \alpha_i \leq E; \\ \sum_{i=1}^{l} \alpha_i \, y_i = 0. \end{cases}$$

Šios sistemos sprendimas pateikia optimalią hiperplokštumą, kuri yra klasifikavimo sprendimo riba tarp galimų klasių (Fletcher, 2009).



11 pav. Pradinių duomenų erdvė ir požymių erdvė [46]

Modelio privalumai:

- Modelis efektyvus, kai duomenys pasižymi dideliu požymių skaičiumi.
- Modelis tinka dirbti su mažos apimties duomenų aibėmis.

Modelio trūkumai:

- Modelis netinka dirbti su triukšmingais, neapdorotais duomenimis.
- Modelis netinka dirbti su didelės apimties duomenų aibėmis kadangi modelio mokymo procesas užima daug laiko.
- Modelis apribotas priimti tik tam tikros struktūros duomenis („Bag of Words“ metriką arba TF-IDF statistiką).

2.3. Bajeso klasifikatorius

Bajeso (angl. *Naive Bayes*) klasifikatorius yra vienas paprasčiausių klasifikavimo algoritmų, sukurtas ir nagrinėtas jau nuo 1960 m. Šis algoritmas veikia su prielaida, kad visi duomenų parametrai yra laikomi nepriklausomais vieno nuo kito ir kad kiekvienas parametras vienodai daro įtaką galutiniam klasifikavimo rezultatui.

Tarkime turime duomenis suskirstytus į klases $C_j, j = 1, 2, \dots, v$, kur v yra galimų klasių skaičius. Tikslas – suklasifikuoti naujai pateiktus objektus remiantis išvesties objektais. Spėjame, kai duomuo d_i turi k nepriklausomų parametrų reikšmių $\{x_{i1}, x_{i2}, \dots, x_{ik}\}$. Turime suskaičiuoti vėlesnę (angl. *posterior*) tikimybę:

$$P(C_j|X_i) = \frac{P(X_i|C_j)P(C_j)}{P(X_i)},$$

tai yra tikimybė, kad X_i priklauso klasei C_j. Čia $P(C_j)$ – priorinės klasės C_j tikimybė, (X_i) – tikimybė, kad X_i yra kiekvienoje klasėje, o $P(X_i|C_j)$ – sąlyginė tikimybė, kuri gali būti išreikšta tokia išraiška:

$$P(X_i|C_j) = \prod_{k=1}^{n} P(x_{ij}|C_j).$$

Tikimybė $P(C_j|X_i)$ apskaičiuojama kiekvienai klasei. Naują duomenį priskiriame klasei C_j, kuris įgyja aukščiausią vėlesnę tikimybę (Rish, 2001).

Modelio privalumai:

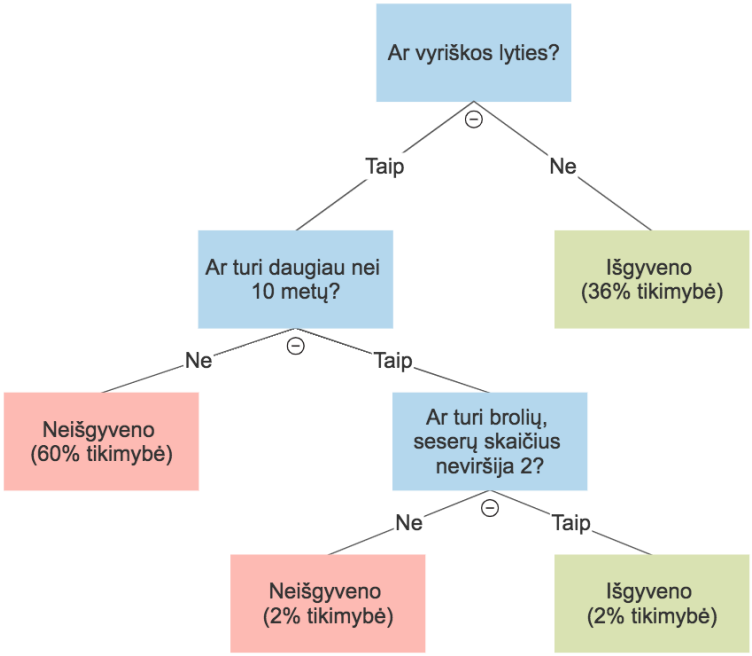
- Modelį nesunku suprasti ir sukurti.
- Modelis tinka dirbti su mažos apimties duomenų rinkiniais.

Modelio trūkumai:

- Modelis veikia su prielaida, kad duomenų požymiai yra nepriklausomi tarpusavyje. Jei ši sąlyga duomenyse negalioja – modelis pateikia prastus rezultatus.
- Modelis netinka dirbti su didelės apimties duomenų rinkiniais.

2.4. Sprendimų medžio klasifikatorius

Sprendimų medžio modelis taip pat yra vienas iš algoritmų, kurie yra dažnai naudojami klasifikavimo uždaviniams spręsti. Duomenų klasifikavimo metu šis algoritmas sukuria medžio modelį iš jau sužymėtų, suklasifikuotų duomenų. Šio medžio mazgai žymi duomenų požymius, kiekviena medžio šaka (sujungimai tarp mazgų ir lapų) žymi sprendimą, taisyklę, o medžio lapai – galutinę išeigą (kategoriją) (žr. 12 pav.) (Maimon & Rokach, 2005).



12 pav. Paprasto sprendimų medžio pavyzdys

Sprendžiant klasifikavimo užduotį, sprendimų medžio algoritmas stengiasi pagal duomenų požymius padalinti, duomenis į smulkesnes aibes tol, kol išgaunami pakankamai maži duomenų rinkiniai, kurių elementai patenka į tą pačią kategoriją. Sprendimų medžio modelio apibrėžimas: duota duomenų aibė $D = \{X_1, X_2, \dots, X_n\}$, kur duomenis X_i apibūdina parametrai $\{x_1, x_2, \dots, x_m\}$, o n – aibėje esančių elementų skaičius, m – duomenų požymių skaičius. Taip pat duota galimų klasių aibė $C = \{C_1, C_2, \dots, C_k\}$, kur k – galimų klasių skaičius. Sprendimų medis yra susietas su D ir turi tokias ypatybes:

- Kiekvienas medžio mazgas yra susiejamas su parametru x_j.
- Kiekviena medžio šaka yra siejama su predikatu, kuri yra taikoma medžio mazgui iš kurio ji eina.
- Kiekvienas medžios lapas yra pažymėtas klase C_l.

Klasifikavimo metu, sprendimų medžio metodas atlieka du žingsnius:

1. Sprendimų medžio modelio indukcija (angl. *induction*) – naudojant mokymo duomenis, sudaromas sprendimų medis.
2. Kiekvienas naujas duomuo $X_i \in D$, pasinaudojus sukurtu medžiu, priskiriamas vienai iš galimų klasių.

Kuriant medžio modelį, algoritmas kiekviename žingsnyje pasirenka tokį kintamąjį (duomenų požymį), kuris padalintų duomenų aibę geriausiu būdu. Matuojant geriausius duomenų padalinimus, algoritmas gali naudoti skirtingas skaičiavimo metrikas, kurias apibendrinus (radus metrikų reikšmių vidurkį) yra nustatomas duomenų padalijimo kokybės matas (Tan et al., 2006). Dažniausiai yra taikomos šios metrikos:

- „Gini“ priemaiša.
- Informacijos išlošis.
- Variacijos sumažinimas.

Modelio privalumai:

- Modelio dalijimo taisykles paprasta suprasti ir interpretuoti.
- Apdoroja tiek skaitinius, tiek kategorinius duomenis.
- Tinka dirbti su didelėmis duomenų aibėmis kadangi medžio dydis nepriklauso nuo duomenų aibės dydžio.

Modelio trūkumai:

- Sprendimų medžio mokymas gali sukurti sudėtingus medžius, kurie neapibendrins mokymo duomenų.
- Modelis sunkiai apdoroja tolydžius duomenis. Siekiant tokio tipo požymius apdoroti, modelis dalina juos į kategorijas.

2.5. K-artimiausių kaimynų metodas

K-artimiausių kaimynų metodas (angl. *k-nearest neighbor,* K-NN) yra dar vienas modelis, kuris dažnai naudojamas klasifikavimo uždaviniams spręsti. Modelio esmė – duomenų pavaizdavimas vektorinėje erdvėje, kurios dimensijos atitinka klasifikacijai naudojamų duomenų požymių aibę.

Prieš atliekant klasifikavimą, modeliui pateikiami mokymo duomenys, kurie turi raktinių požymių. Šie mokymo duomenys yra iš anksto priskirti kategorijoms. Kiekvienam naujam klasifikuojamam elementui yra vykdomi šie žingsniai:

1. Elementui randami *k* artimiausių elementų iš mokymo duomenų aibės. Elementų artumas nustatomas pagal vieną iš pasirinktinų atstumo matų: euklidinis, kosinusinis ir t. t. (Bandyopadhyay & Saha, 2012).
2. Išnagrinėjus *k* artimiausius kaimynus nustatoma, kuriai kategorijai atrinktų elementų priklauso daugiausiai ir galiausiai tai kategorijai priskiriamas klasifikuojamas elementas.

Modelio matematinis apibrėžimas: naujo elemento *e* priskyrimas kategorijai C_i, jei C_i turi didžiausią panašumo rodiklį (angl. *similarity score*) tarp visų galimų kategorijų. Panašumo rodiklis elementui *e* kategorijai C_i apskaičiuojamas pagal formulę:

$$s(e, C_i) = \sum_{e_j \in k-NN} sim(e, e_j) y(e_j, C_i),$$

kur $sim(e, e_j)$ – panašumas tarp elemento *e* ir mokomojo elemento e_j (dažniausiai apskaičiuojamas kaip euklidinis atstumas tarp vektorių atitinkančių *e* ir e_j; $e_j \in k - NN$ reiškia, kad e_j yra iš k artimiausių *e* kaimynų funkcijos $sim()$ atžvilgiu; $y(e_j, C_i)$ – funkcija, kuri įgyja reikšmę 1, kai elementas e_j priklauso kategorijai C_i ir 0 kitu atveju. Apskaičiavus visus minėtus rodiklius, elemento priskyrimas kategorijai yra vykdomas pagal formulę:

$$\arg \max_{j=1,\dots,m} \left(s(e, C_i)\right),$$

kur $C_1, C_2, \dots, C_m$ – iš anksto apibrėžtos galimos kategorijos (Kramer, 2013).

Modelio privalumai:

- Modelis yra intuityvus, paprastas ir lengva jį sukurti.
- Modelio mokymas neužima daug laiko.

Modelio trūkumai:

- Modelis užtrunka daug laiko klasifikuojant didelės apimties duomenų aibes.
- Modelis prastai veikia, kai duomenys turi daug požymių.
- Modelis reikalauja pasirinkti optimalų kaimyninių elementų kiekį.

3. POŽYMIŲ IŠSKYRIMAS

3.1. Tekstinių požymių išskyrimo problematikos

Didžioji dalis sistemų mokymosi algoritmų geba operuoti vektoriais, kurie yra sudaryti iš realiųjų skaičių. Šių vektorių elementai sistemų mokymosi kontekste dažnai vadinami požymiais (angl. *features*). Norint klasifikuoti tekstą sistemų mokymosi algoritmais, reikalingi požymių išskyrimo (angl. *feature extraction*) algoritmai, kurie sugeba tekstą transformuoti į skaitinį pavidalą, tinkamą klasifikuojant (Aggarwal & Zhai, 2012).

Išskiriant teksto požymius dažniausiai susiduriama su šiomis problemomis:

1. Tekstas sudarytas iš simbolių. Simboliai – ne skaičiai, o kategoriniai duomenys.
2. Dauguma sistemų mokymosi algoritmų įvesties vektoriai yra fiksuoto ilgio. Teksto ilgis yra nepastovus.
3. Kai išskirtų požymių skaičius yra didelis, sistemų mokymosi algoritmai sunaudoja daug kompiuterių resursų skaičiavimams atlikti, todėl pageidaujama, kad požymių būtų ne per daug. Didelį kiekį požymių sukuria žodžių ypatybė natūralioje kalboje turėti daug skirtingų gramatinių formų.

3.2. Teksto paruošimas požymių išskyrimui

Prieš išskiriant teksto požymius, dažnai atliekamos transformacijos, kurios supaprastina tekstinių duomenų struktūrą ir pašalina dalį duomenų triukšmo:

1. Skaidymas į žodžius.
2. Žodžių kanonizavimas.
3. Nereikšmingų žodžių šalinimas.

Skaidymas į žodžius

Skaidymas į žodžius – tekstinių duomenų transformacija (angl. *tokenization*), kurios metu tekstas yra išskaidomas į atskirus elementus – žodžius. Transformacijos metu žodžiai yra atskiriami nuo skyrybos ženklų esančių tekste. Skyrybos ženklai gali būti pašalinami iš teksto arba laikomi savarankiškais elementais. Verta paminėti, kad nors ši transformacija atrodo triviali daugumai kalbų, yra ir išimčių, pavyzdžiui, japonų, kinų kalbos, kur žodžiai nėra skiriami tarpais (Trim, 2019).

Žodžių kanonizavimas

Žodžių kanonizavimas (angl. *canonicalization*) – transformacija, kurios metu žodžiai yra normalizuojami, paverčiami į standartinę formą. Egzistuoja keli žodžių kanonizavimo būdai:

1. Rašybos taisymas.
2. Žodžio raidžių suvienodinimas.
3. Kamieno nustatymas (angl. *stemming*), kurio metu pašalinama žodžio galūnė.

4. Lemavimas (angl. *lemmatization*), kurio metu žodis yra paverčiamas į pagrindinę jo formą. Pavyzdžiui, bendratį (jei žodis – veiksmažodis), vienaskaitos vardininko linksnį (jei žodis – daiktavardis).

Pagrindinė kanonizavimo paskirtis – sumažinti teksto požymių skaičių (Kantrowitz et al., 2000).

Nereikšmingų žodžių šalinimas

Nereikšmingas žodžių šalinimas – transformacija, kurios metu iš teksto pašalinami dažnai sutinkami arba savarankiškos prasmės neturintys žodžiai (angl. *stop words*) (Ganesan, 2019). Transformacijai atlikti yra sudaromas specialus nereikšmingų žodžių sąrašas, į kurį paprastai įtraukiami:

1. Prielinksniai.
2. Artikeliai.
3. Jungtukai.
4. Įvardžiai.

3.3. Požymių išskyrimo metodai

3.3.1. TF-IDF statistika

TF-IDF statistika (angl. *Term Frequency Inverse Document Frequency*) yra „Bag of Words“ modelio variacija. „Bag of words“ modelis yra supaprastinta tekstinių duomenų reprezentacija naudojama apdorojant tekstą ir iš jo išgaunant informaciją. Taikant šį modelį tekstas išreiškiamas kaip multiaibė, kurios elementai yra žodžiai (Brownlee, 2019). Pagal modelio apibrėžimą, tekstas *T* paverčiamas vektoriumi *V* tokiu principu:

1. Sudaromas žodynas *W* iš *n* žodžių (dažniausiai sudaromas iš mokymo duomenų):
$$W = \{w_1, w_2, \dots, w_n\}.$$
2. Suskaičiuojama, kiek kartų tekste *T* pasikartoja kiekvienas žodis esantis žodyne *W*. Žodžių pasikartojimų skaičiai pažymimi $c(T, w_1), c(T, w_2), \dots, c(T, w_n)$.
3. Galiausiai sudaromas vektorius *V*:
$$V = \big(c(T, w_1), c(T, w_2), \dots, c(T, w_n) \big).$$

Nors šis modelis yra vienas iš paprasčiausių ir dažniausiai taikomų metodų dirbant su tekstiniais duomenimis, jis pasižymi keliais trūkumais:

- Modelis neatsižvelgia į žodžių tvarką tekste.
- Jei žodynas yra didelis, sudarytas vektorius *V* turės daug dimensijų, kurios gali neigiamai paveikti teksto klasifikatoriaus tikslumą.

Taikant TF-IDF metodą atsižvelgiama ne tik į žodžio pasikartojimų tekste skaičių, bet ir į žodžio svarbą duotai tekstų aibei. TF-IDF statistika remiasi intuicija, kad dažnai pasitaikantys žodžiai nėra

geras rodiklis tekstams atskirti. Svarbiais žodžiais laikomi tie, kurie yra aptinkami mažesniame kiekyje tekstų. TF-IDF statistikai apskaičiuoti yra apibrėžti du dydžiai:

1. Sąvokos dažnumas (angl. *term frequency*).

   Dažniausiai skaičiuojamas žodžio pasikartojimų skaičius tekste:

   $$tf(t,w) = c(t,w),$$

   čia *t* – tekstas, *w* – žodis, kuriam skaičiuojama statistika.

2. Sąvokos atvirkštinis dažnumas dokumentuose (angl. *inverse document frequency*).

   Šis dydis apskaičiuojamas pagal šią formulę:

   $$idf(T,w) = ln\frac{|T|}{|p(T,w)|},\ \ p(T,w) = \{t: t \in T \wedge w \in t\},$$

   čia *T* – tekstų aibė, *w* – žodis, kuriam statistika skaičiuojama, *p*(*T*, *w*) – aibės *T* poaibis, į kurį įeina tik tie tekstai, kuriuose yra žodis *w*.

Žodžio *w* TF-IDF statistika yra lygi dviejų minėtų dydžių sandaugai:

$$tfidf(T,t,w) = tf(t,w) \times\ idf(T,w),$$

čia *T* – tekstų aibė, *t* – aibei priklausantis tekstas, *w* – teksto žodis.

Remiantis TF-IDF statistikos apibrėžimu:

- Kuo rečiau žodis sutinkamas tekste, tuo didesnė jo TF-IDF statistika.
- Žodžio TF-IDF statistika yra lygi nuliui, jei žodis yra aptinkamas visuose tekstuose.

Statistika pasižymi tokiais pat trūkumais kaip ir „Bag of Words“ modelis (Góralewicz, 2019).

3.3.2. LSA metodas

LSA metodas (angl. *Latent Semantic Analysis*) yra skirtas paslėptiems tekstinių duomenų kontekstams rasti. Taikant šį metodą, stengiamasi žodžius tarpusavyje susieti užslėptu kontekstu, geriausiai apibūdinančiu nagrinėjamus žodžius (Landaeur et al., 2019). LSA metodas apskaičiuoja dokumentų arba dokumento ištraukų reprezentacijas tokiu principu:

1. Sudaroma $m \times n$ dydžio žodžio-ištraukos matrica *A*, kur kiekviena eilutė atitinka žodį *w*, o stulpelis – ištrauką *d*. Matricos elemento a_{ij} reikšmė gali būti lygi žodžio w_i pasikartojimo skaičiui teksto ištraukoje d_j („Bag of Words“ modelis) arba lygi TF-IDF statistikai.

	$\boldsymbol{d_1}$	$\boldsymbol{d_2}$	...	$\boldsymbol{d_n}$
$\boldsymbol{w_1}$	a_{11}	a_{12}	...	a_{1n}
$\boldsymbol{w_2}$	a_{21}	a_{22}	...	a_{2n}
...	...	...	...	...
$\boldsymbol{w_m}$	a_{m1}	a_{m2}	...	a_{mn}

2. Apskaičiuojamos $m \times m$ dydžio ištrauka-ištraukos matrica B ir $n \times n$ dydžio žodžio-žodžio matrica *C*:

$$B = A^T A, \qquad C = AA^T$$

3. Atliekamas matricos *A* išskaidymas singuliariomis reikšmėmis:

$$A = S\Sigma U^T,$$

čia *S* yra matricos *B* tikrinių vektorių matrica, *U* – matricos *C* tikrinių vektorių matrica ir Σ yra singuliarių reikšmių diagonalinė matrica, kurios įstrižainės reikšmės apskaičiuojamos traukiant kvadratinę šaknį iš matricos *B* tikrinių reikšmių.

4. Singuliarios reikšmės, kurios yra mažos – laikomos nereikšmingomis, todėl šiame žingsnyje parenkama *k* didžiausių (reikšmingiausių) singuliarių reikšmių. Likusios diagonalinės matricos Σ reikšmės yra ignoruojamos (pakeičiamos nuliais) ir matrica Σ yra sumažinama iki $k \times k$ dydžio matricos.
5. Sumažinus Σ matricą, tuo pačiu sumažėja matricų *S, U* ir *A* dydžiai:

$$A_k = S_k \Sigma_k U_k^T$$

LSA metodu apskaičiuotos reikšmės teksto reprezentacijos matricoje leidžia išreikšti žodžių ir ištraukų panašumus į pagalbą pasitelkiant skaliarinę sandaugą, kosinusą bei kitas panašias metrikas. Pagrindinis šio metodo privalumas – nagrinėjamų duomenų matricos dydžio sumažinimas.

LSA metodo trūkumai:

- Prastai išreiškia žodžius, kurie turi daug galimų reikšmių.
- Metodas priklausomas nuo SVD algoritmo (angl. *Singular Value Decomposition*), kuris dažniausiai reikalauja nemažai resursų skaičiavimams atlikti.

3.3.3. LDA metodas

Tiesinės diskriminantinės analizės metodas (angl. *Linear Discriminant Analysis,* LDA) – projekcijos metodas, kuris veikia pagal mokymo su mokytoju strategiją. Šis metodas išnagrinėja duomenis, jų savybes (priklausymą kuriai nors klasei) ir transformuoja daugiamatės erdvės duomenis į mažesnės erdvės duomenis taip, kad klasių atskiriamumo kriterijaus reikšmė būtų optimali (Raschka, 2019). LDA metodas transformuoja duomenis tokiu principu:

1. Turime duomenų matricą *V*, kurios kiekvienoje eilutėje yra vektorius V_i, o matricos stulpeliai – duomenis apibūdinantys atributai. Matrica *V* gali būti apskaičiuota taikant „Bag of Words“ metodą arba TF-IDF statistiką.

$$V = \{V_1, V_2, \dots, V_m\}, \quad V_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}.$$

Čia *m* – duomenų vektorių (nagrinėjamų atvejų) kiekis, *n* – duomenis apibūdinančių atributų kiekis.

2. Tegul duomenų matrica *V* sudaryta iš kelių matricų $V^{(1)}$, $V^{(2)}$, ..., $V^{(k)}$, kurių eilutės yra vektoriai $V_i^{(j)}, i = 1, \dots, m_j, j \in \{1, \dots, k\}$, atitinkantys objektus, priklausančius vienai iš k galimų klasių, o m_j yra objektų, priklausančių *j*-ajai klasei, skaičius.
3. Apskaičiuojamas objektus $V_i^{(j)}$ apibūdinančių atributų visos aibės *V* reikšmių vidurkių vektorius $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ ir kiekvienos klasės aibės $V^{(j)}$ reikšmių vidurkių vektoriai $\mu^{(j)}$.
4. Apskaičiuojami kovariacijos koeficientai, sudaroma bendra kovariacinė matrica *C* ir atskirų klasių kovariacinės matricos $C^{(j)}, j = 1, \dots, k$.
5. Apskaičiuojamas išsibarstymas klasių viduje (angl. *within-class scatter*):

$$S_w = \sum_{j=1}^{k} p_j C^{(j)},$$

čia p_j – klasių tikimybė ($p_j = \frac{m_j}{m}$, kur m_j – objektų, priklausančių *j*-ajai klasei, skaičius, o m – visų nagrinėjamų objektų skaičius).

6. Apskaičiuojamas tarpklasinis išsibarstymas (angl. *between-class scatter*):

$$S_b = C - S_w.$$

7. Randami matricos $S_w^{-1} S_b$ tikriniai vektoriai ir tikrinės reikšmės. Vektoriai surūšiuojami tikrinių reikšmių mažėjimo tvarka. Atrenkami d ($d < k$) tikriniai vektoriai, kurie turi atitinkamas didžiausias tikrines reikšmes. Iš atrinktų vektorių sudaroma matrica A_d.
8. Apskaičiuojama duomenų transformacija:

$$Y_i = (V_i - \mu) \times A_d, \;\; i = 1, \dots, m,$$

čia A_d yra matrica sudaryta iš tikrinių vektorių,

Pagrindinis LDA metodo privalumas – gebėjimas gerokai sumažinti duomenų požymių erdvę iki galimų klasių skaičiaus, kas pagreitina klasifikatoriaus mokymo laiką.

Didžiausias metodo trūkumas – transformuojant didesnę požymių erdvę į mažesnę, atsiranda rizika prarasti nemažai informacijos nagrinėjamuose duomenyse.

4. POŽYMIŲ IŠSKYRIMO METODŲ PALYGINIMAS

Siekiant palyginti ankstesniuose skyriuose aprašytus požymių išskyrimo metodus, buvo atliktas eksperimentas. Šiame skyriuje aprašyta eksperimento atlikimo metodika ir pateikti eksperimento metu gauti rezultatai.

4.1. Tyrimo aprašymas

Požymių išskyrimo metodai buvo lyginami pagal vieną kriterijų – klasifikatoriaus tikslumą. Klasifikavimo tikslumas yra dalis testavimo duomenų aibės elementų, kuriems klasifikatorius teisingai priskyrė klasę. Tikslumas skaičiuojamas tokiu principu:

$$T = \frac{|D_T|}{|D_V|},$$

čia T – klasifikatoriaus tikslumas, D_T – teisingai suklasifikuoti testavimo duomenys, D_V – visi testavimo duomenys.

Eksperimento metu tarpusavyje buvo lyginami šie požymių išskyrimo algoritmai:

1. TF-IDF statistika;
2. TF-IDF LSA metodas;
3. TF-IDF LDA metodas.

Požymių išskyrimo algoritmai buvo suprogramuoti „Python“ programavimo kalba pasinaudojant „scikit-learn“ biblioteka.

Žodynai (duomenys) visiems šiems požymių išskyrimo metodams buvo sudaryti iš apdorotų mokymo duomenų (kanonizuoti žodžiai, pašalinti nereikšmingi žodžiai). LSA metodas buvo nustatytas visada sudaryti 500 požymių turinčius duomenų vektorius.

Klasifikavimo tikslumas gali priklausyti ne tik nuo taikomo požymių išskyrimo metodo algoritmo, bet ir nuo to, kaip tekstiniai duomenys yra apdorojami prieš išskiriant požymius iš jų. Su visais minėtais požymių išskyrimo metodais matavimai buvo atlikti tris kartus:

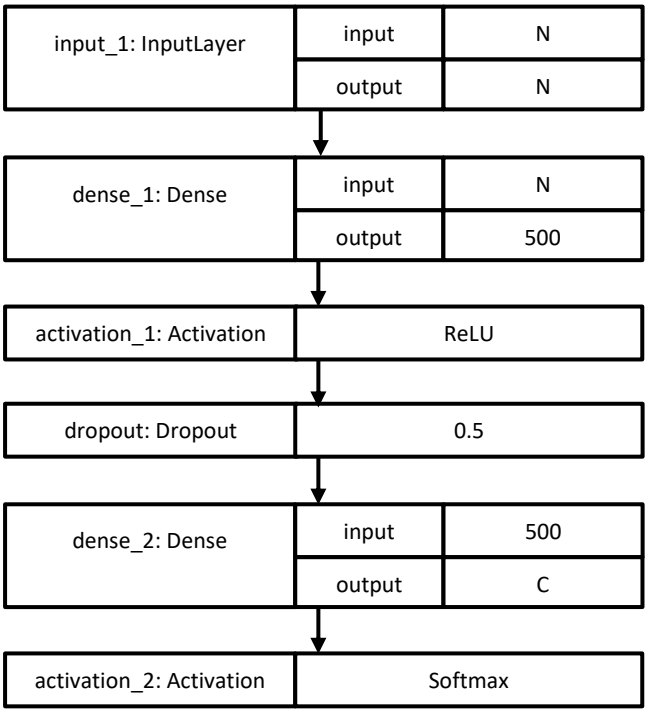
1. Su nepakeistais duomenimis.
2. Su duomenimis, kuriuose žodžiai kanonizuoti kamieno nustatymo būdu.
3. Su duomenimis, kuriuose žodžiai kanonizuoti ir pašalinti nereikšmingi žodžiai.

Visų minėtų požymių išskyrimo algoritmų išvestis buvo klasifikuojama naudojant vieną ir tą patį neuroninio tinklo modelį. Neuroninio tinklo modelis buvo suprogramuotas „Python“ programavimo kalba naudojantis „Keras“ ir „Tensorflow“ bibliotekomis.

Modelis pasižymėjo tokia architektūra:

1. Vienas paslėptas sluoksnis turintis 500 neuronų. Jų aktyvacijos funkcija – „ReLU“ funkcija.
2. Išvesties sluoksnis turintis tiek neuronų, kiek yra klasių duomenų mokymo aibėje. Jų aktyvacijos funkcija – „Softmax“ funkcija.

Mokant neuroninio tinklo modelį, klasifikatoriaus veiksmingumo normalizavimui buvo naudojamas išmetimo (angl. *dropout*) metodas.



13 pav. Neuroninio tinklo modelio architektūra

Pirmas sluoksnis „input_1“ yra įvesties sluoksnis, kurį sudaro mokymo duomenų vektoriai, kur kiekvienas vektorius turi *N* požymių. Sluoksniai „Dense“ yra sudaryti iš neuronų, kur kiekvienas neuronas yra sujungtas su ankstesnio ir paskesnio tinklo neuronu. Kiekvienas šis sluoksnis turi po aktyvacijos funkciją, kuri pašalina duomenų triukšmą iš sluoksnio išvesties prieš paduodant ją sekančiam neuronų sluoksniui. Sluoksnis „dense_1“ naudoja „ReLU“ aktyvacijos funkciją, o sluoksnis „dense_2“ – „Softmax“ funkciją. Sluoksnis „Dropout“ yra metodas neleisti klasifikavimo modeliui persimokyti. Šis metodas mokymo metu atsitiktinai pasirenka dalį sluoksnio įvesties reikšmių ir priskiria joms nulinę reikšmę. Pateiktame modelyje pasirinkta sluoksnio įvesties reikšmių dalis yra lygi 0,5. Paskutinis sluoksnis „dense_2“ kaip išvestį pateikia vektorius, turinčius *C* požymių. Skaičius *C* yra lygus kiekiui galimų kategorijų, į kuriuos gali būti suklasifikuotas tekstas. Kiekvienas vektorius atitinka teksto ištrauką, o vektoriaus reikšmės – teksto ištraukos tikimybes priklausyti

vienai ar kitai kategorijai. Modeliui taikomas „categorical cross entropy“ kainos funkcija, kuri įvertina, kaip gerai modelis klasifikuoja (žr. 13 pav.).

Mokant tą patį tinklą kelis kartus, kiekvieną kartą gaunamas nevienodas klasifikavimo tikslumas. Tai nutinka dėl dviejų priežasčių: neuroninio tinklo modelio pradiniai svoriai parenkami atsitiktinai ir modelio kainos funkcija dažnai turi lokalių minimumų. Šios problemos įtaka eksperimento rezultatams buvo sumažinta, mokant po 4 neuroninio tinklo modelius su kiekvienu požymių išskyrimo metodu. Palyginimo tikslu naudojamas jų klasifikavimo tikslumų vidurkis.

4.2. Duomenų aibė

Eksperimento metu buvo naudojama duomenų aibė, kuri buvo surinkta naudojant savadarbį scenarijų (angl. *script*), parašytą „Python“ programavimo kalba. Šis scenarijus renka tam tikrų kategorijų skelbimų antraštes iš skelbimų tinklapių „alio.lt“ ir „skelbiu.lt“ (žr. A priedas). Duomenų aibę iš viso sudarė 10 000 skelbimų antraščių, suskirstytų į 20 kategorijų.

Duomenų aibė buvo sudaryta iš lietuviškų tekstų (antraščių), todėl tekstinių duomenų apdorojimo algoritmuose naudojami lietuvių kalbos nereikšmingų žodžių sąrašas ir lietuvių kalbai skirta kamieno nustatymo programa „Snowball“.

Eksperimento metu, buvo naudojamos skirtingų dydžių duomenų imtys (500, 1000, 2000, 4000, 8000, 10 000). Kiekvieno bandymo metu, duomenų imtis buvo padalinta į dvi dalis: mokymo ir testavimo duomenis santykiu 80:20.

4.3. Tyrimo eiga

Tyrimas buvo atliktas tokia tvarka:

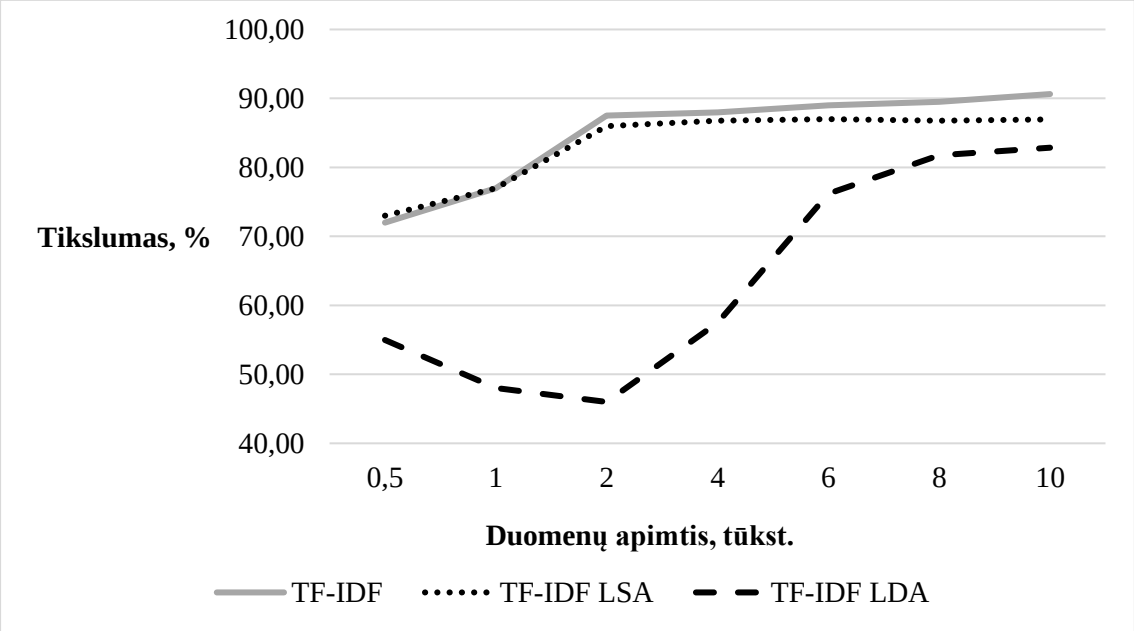
1. Buvo surinkti tekstiniai duomenys iš įvairių lietuviškų skelbimų tinklapių.
2. Surinkti duomenys buvo apdoroti:
    a. Pašalintas triukšmas duomenyse.
    b. Žodžiai buvo kanonizuoti (paversti į kamieninę formą).
3. Apdoroti tekstiniai duomenys buvo padalinti į du rinkinius: mokymo ir testavimo duomenis santykiu 80:20.
4. Iš mokymo duomenų buvo išgauti duomenis apibūdinantys požymiai.
5. Išgauti tekstinių duomenų požymiai buvo pateikti klasifikavimo modelio mokymui.
6. Baigus mokyti klasifikavimo modelį, kad patikrinti jo tikslumą, modeliui buvo pateikiami tekstiniai duomenys iš testavimo duomenų rinkinio.
7. Suklasifikavus testavimo duomenis, gauti rezultatai buvo palyginti su realiais rezultatais (žr. B priedas).

4.4. Tyrimo rezultatai

Duomenų aibės klasifikavimo tikslumo įverčiai pateikti 1 lentelėje ir 14 paveiksle, o klasifikavimo modelio kainos funkcijos reikšmės pateiktos 15 paveiksle. 1 lentelėje didesni įverčiai paryškinti kita spalva.

1 lentelė. Klasifikavimo modelio tikslumo įverčiai taikant skirtingus požymių išskyrimo metodus

Duomenų dydis, tūkst.	TF-IDF	TF-IDF LSA	TF-IDF LDA
0,5	72,00 %	73,00 %	55,00 %
1	77,00 %	77,00 %	48,00 %
2	87,50 %	86,00 %	46,00 %
4	88,00 %	86,75 %	57,50 %
6	89,00 %	87,00 %	76,18 %
8	89,51 %	86,77 %	81,77 %
10	90,63 %	86,94 %	82,86 %

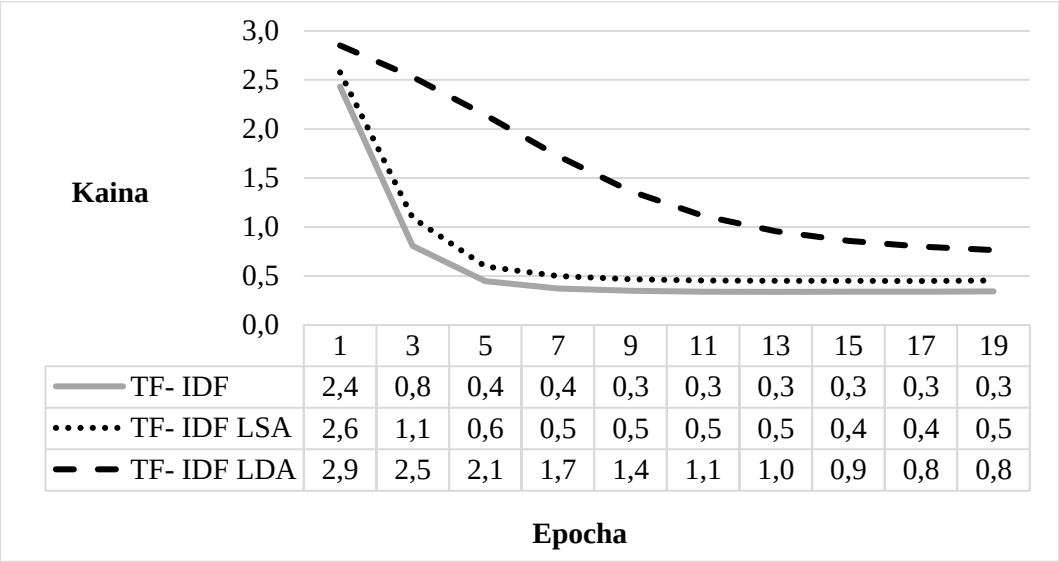


14 pav. Klasifikavimo tikslumas taikant skirtingus požymių išskyrimo metodus

Išnagrinėjus bandymų metu gautus tikslumo įverčius, matoma, kad:

1. Didžiausias tikslumas buvo 90,63 %. Jis buvo pasiektas modeliui klasifikuojant didžiausią duomenų imtį ir duomenų požymius išskiriant TF-IDF metodu.
2. Didžiąją dalį bandymų klasifikavimo modelis demonstravo geresnius rezultatus taikant TF-IDF požymių išskyrimo metodą.

3. Klasifikuojant mažiausią duomenų imtį, didžiausias tikslumas buvo 73,00 %. Jis buvo pasiektas duomenų požymius išskiriant TF-IDF LSA metodu.
4. Taikant TF-IDF LDA metodą, klasifikavimo modelis demonstravo prastus rezultatus dirbant su mažesnėmis duomenų imtimis, tačiau rezultatai pradėjo itin gerėti dirbant su didesniais duomenų kiekiais.



	1	3	5	7	9	11	13	15	17	19
TF- IDF	2,4	0,8	0,4	0,4	0,3	0,3	0,3	0,3	0,3	0,3
TF- IDF LSA	2,6	1,1	0,6	0,5	0,5	0,5	0,5	0,4	0,4	0,5
TF- IDF LDA	2,9	2,5	2,1	1,7	1,4	1,1	1,0	0,9	0,8	0,8

15 pav. Klasifikavimo kainos funkcijos reikšmės taikant skirtingus požymių išskyrimo metodus

5. KLASIFIKAVIMO MODELIŲ PALYGINIMAS

Siekiant palyginti ankstesniuose skyriuose aprašytus klasifikavimo modelius, buvo atliktas tyrimas, kurio metu buvo tarpusavyje palyginti penkių skirtingų klasifikatorių tikslumai, esant keturioms sąlygoms:

1. kai kinta apdorojamų duomenų apimtis;
2. kai kinta galimų klasių skaičius;
3. kai kinta mokymo ir testavimo duomenų apimties santykis;
4. kai klasifikuojami tekstiniai duomenys priklauso skirtingoms kalboms.

5.1. Bendras tyrimų aprašymas

Klasifikavimo algoritmai buvo lyginami pagal vieną kriterijų – klasifikavimo tikslumą, kuris buvo skaičiuojamas pagal formulę:

$$A = \frac{|T|}{|V|},$$

čia A – klasifikatoriaus tikslumas, T – teisingai suklasifikuoti testavimo duomenys, V – visi testavimo duomenys.

Eksperimento metu tarpusavyje buvo lyginami šie klasifikavimo algoritmai:

1. Atraminių vektorių klasifikatorius.
2. Bajeso klasifikatorius.
3. Sprendimų medžio klasifikatorius.
4. K-artimiausių kaimynų metodas.
5. Neuroninis tinklas (daugiasluoksnis perceptronas).

Duomenų požymiams išgauti buvo taikoma TF-IDF statistika, kadangi pagal 4 skyriuje aprašytą tyrimą, šis požymių išskyrimo metodas pateikė geriausius rezultatus.

Teksto klasifikavimo modeliai buvo suprogramuoti „Python“ programavimo kalba. Neuroninis tinklas sukurtas naudojant „Tensorflow“ ir „Keras“ bibliotekas, o visiems likusiems algoritmams sukurti buvo panaudota „scikit-learn“ biblioteka.

Eksperimentuose naudotas tokios pat architektūros su tais pačiais parametrais neuroninio tinklo klasifikatorius kaip ir neuroninis tinklo modelis aprašytas tyrime 4 skyriuje.

Klasifikatorių konfigūracijos:

- Atraminių vektorių klasifikatoriaus konfigūracija:
    - implementacija: „scikit-learn“ bibliotekos metodas „SGDClassifier“;
    - kainos funkcija: „squared_hinge“;
    - iteracijų skaičius: 10.

- Bajeso klasifikatoriaus konfigūracija:
    - implementacija: „scikit-learn“ bibliotekos metodas „GaussianNB“.
- Sprendimų medžio klasifikatoriaus konfigūracija:
    - implementacija: „scikit-learn“ bibliotekos metodas „DecisionTreeClassifier“.
- K-artimiausių kaimynų klasifikatoriaus konfigūracija:
    - implementacija: „scikit-learn“ bibliotekos metodas „KNeighborsClassifier“.

Mokant tą patį klasifikatoriaus modelį kelis kartus, kiekvieną kartą gaunamas nevienodas klasifikavimo tikslumas. Tai nutinka dėl dviejų priežasčių: modelio kainos funkcija dažnai turi lokalių minimumų arba rezultatams įtaką padaro mokymui paimta duomenų aibė. Šios problemos įtaka eksperimento rezultatams buvo sumažinta, mokant kiekvieną klasifikavimo modelį po 4 kartus, o klasifikavimo tikslumui palyginti buvo naudojamas kiekvieno modelio klasifikavimo tikslumų vidurkis.

Eksperimentuose naudojamos tekstinių duomenų aibės buvo apdorotos. Siekiant pašalinti triukšmą duomenyse, iš duomenų buvo pašalinti nereikšmingi žodžiai. Likę žodžiai buvo standartizuoti panaudojus kamieno nustatymo programą „Snowball“.

Klasifikatorių žymėjimai:

- AVK – atraminių vektorių klasifikatorius.
- BK – Bajeso klasifikatorius.
- SMK – sprendimų medžio klasifikatorius.
- K-NN – artimiausių kaimynų klasifikatorius.
- NN – neuroninis tinklas.

5.2. Klasifikavimas kintant apdorojamų duomenų apimčiai

Šiame skyrelyje aprašomas eksperimentas, kurio metu yra įvertinamas kiekvieno klasifikavimo modelio klasifikavimo tikslumas, kai kinta apdorojamų duomenų apimtis.

5.2.1. Duomenų aibė

Eksperimento metu buvo naudojama duomenų aibė, kuri buvo surinkta pasinaudojant savadarbiu scenarijumi parašytu „Python“ programavimo kalba. Parašytas scenarijus renka tam tikrų kategorijų skelbimų antraštes iš penkių skelbimų tinklapių „plius.lt“, skelbiu.lt“, skelbimai.lt“, rinka.lt“ ir „alio.lt“. Duomenų aibę iš viso sudarė 100 000 skelbimų antraščių suskirstytų į 8-ias kategorijas (žr. A priedas).

Eksperimento metu, buvo naudojamos skirtingų dydžių duomenų imtys (20 000, 40 000, 60 000, 80 000, 100 000). Kiekvieno bandymo metu, duomenų imtis buvo padalinta į dvi dalis: mokymo ir testavimo duomenis santykiu 80:20.

5.2.2. Tyrimo rezultatai

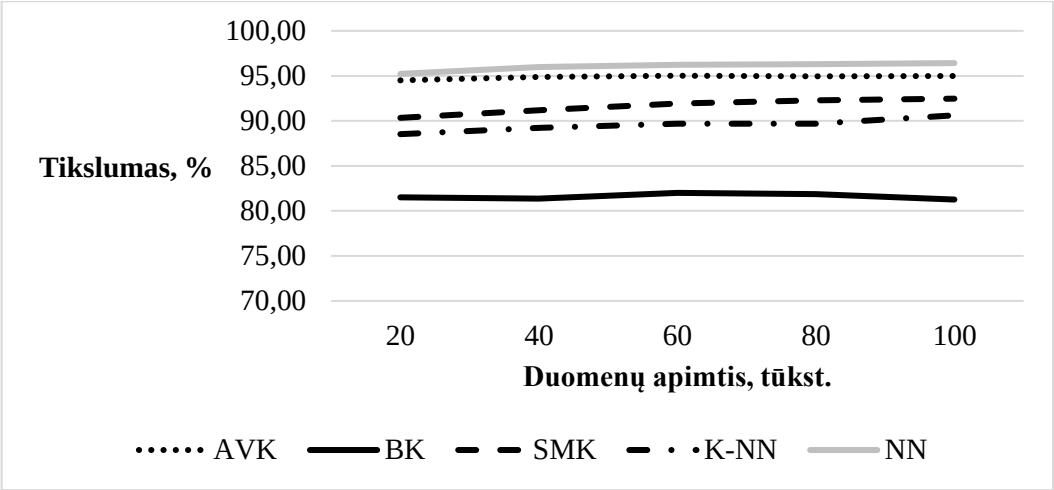
Duomenų aibės klasifikavimo tikslumo įverčiai pateikti 2 lentelėje ir 16 paveiksle. 2 lentelėje didesni įverčiai paryškinti kita spalva.

2 lentelė. Klasifikavimo tikslumas kintant apdorojamų duomenų apimčiai

Duomenų apimtis, tūkst.	**AVK**	**BK**	**SMK**	**K-NN**	**NN**
20	94,50 %	81,52 %	90,34 %	88,52 %	95,22 %
40	94,88 %	81,35 %	91,19 %	89,23 %	95,96 %
60	95,02 %	82,00 %	91,92 %	89,70 %	96,23 %
80	94,93 %	81,87 %	92,29 %	89,70 %	96,30 %
100	94,97 %	81,26 %	92,47 %	90,62 %	96,42 %

Išnagrinėjus eksperimento metu gautus tikslumo įverčius, matome, kad:

1. Apdorojant skirtingos apimties duomenis, visais atvejais, neuroninio tinklo klasifikatorius parodė geriausius rezultatus, pasiekdamas didžiausią tikslumą.
2. Didžiausias klasifikavimo tikslumas (96,42 %) buvo pasiektas apdorojant didžiausią duomenų apimtį (100 tūkst. tekstinių pavyzdžių) ir naudojant neuroninio tinklo klasifikatorių.
3. Pagal klasifikavimo tikslumą, antras geriausias klasifikatorius – atraminių vektorių klasifikatorius, kuris visais atvejais mažai atsiliko nuo neuroninio tinklo klasifikatoriaus rezultatų.
4. Prasčiausius rezultatus parodė Bajeso klasifikatorius, kuris visais atvejais pasiekdavo mažiausią klasifikavimo tikslumą palyginus su kitais klasifikatoriais.
5. Mažiausias klasifikavimo tikslumas – 81,26 %. Jis buvo pasiektas apdorojant didžiausią duomenų apimtį (100 tūkst. tekstinių antraščių) ir naudojant Bajeso klasifikatorių.



16 pav. Klasifikavimo tikslumas kintant apdorojamų duomenų apimčiai

5.3. Klasifikavimas kintant galimų klasių skaičiui

Šiame skyrelyje aprašomas eksperimentas, kurio metu yra įvertinamas kiekvieno klasifikavimo modelio klasifikavimo tikslumas, kai kinta galimų klasių skaičius.

5.3.1. Duomenų aibė

Eksperimento metu buvo naudojama duomenų aibė, kuri buvo surinkta pasinaudojant savadarbiu scenarijumi parašytu „Python“ programavimo kalba. Parašytas scenarijus renka tam tikrų kategorijų skelbimų antraštes iš penkių skelbimų tinklapių „plius.lt“, skelbiu.lt“, skelbimai.lt“, rinka.lt“ ir „alio.lt“. Duomenų aibę iš viso sudarė 100 000 skelbimų antraščių suskirstytų į 8-ias kategorijas.

Eksperimento metu, tekstiniai duomenys buvo klasifikuojami kintant galimų kategorijų skaičiui (2, 4, 6, 8). Kiekvieno bandymo metu, duomenų imtis buvo padalinta į dvi dalis: mokymo ir testavimo duomenis santykiu 80:20.

5.3.2. Tyrimo rezultatai

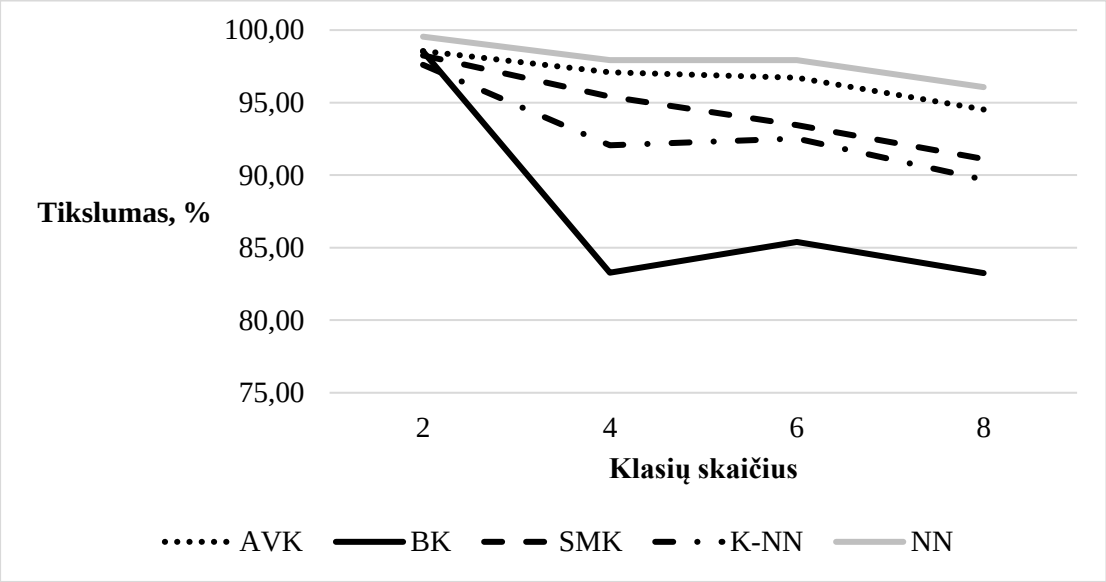
Duomenų aibės klasifikavimo tikslumo įverčiai pateikti 3 lentelėje ir 17 paveiksle. 3 lentelėje didesni įverčiai paryškinti kita spalva.

3 lentelė. Klasifikavimo tikslumas kintant galimų klasių skaičiui

Klasių skaičius	AVK	BK	SMK	K-NN	NN
2	98,56 %	98,53 %	98,25 %	97,61 %	99,55 %
4	97,10 %	83,27 %	95,42 %	92,07 %	97,92 %
6	96,72 %	85,39 %	93,45 %	92,53 %	97,92 %
8	94,53 %	83,25 %	91,12 %	89,69 %	96,07 %

Išnagrinėjus eksperimento metu gautus tikslumo įverčius, matome, kad:

1. Apdorojant skirtingos apimties duomenis, visais atvejais, neuroninio tinklo klasifikatorius parodė geriausius rezultatus, pasiekdamas didžiausią tikslumą.
2. Didžiausias klasifikavimo tikslumas (99,55 %), buvo pasiektas klasifikuojant duomenis į dvi klases ir naudojant neuroninio tinklo klasifikatorių.
3. Pagal klasifikavimo tikslumą, antras geriausias klasifikatorius – atraminių vektorių klasifikatorius, kuris visais atvejais mažai atsiliko nuo neuroninio tinklo klasifikatoriaus rezultatų.
4. Prasčiausius rezultatus parodė Bajeso klasifikatorius, kuris dauguma atvejų pasiekdavo mažiausią klasifikavimo tikslumą palyginus su kitais klasifikatoriais.
5. Mažiausias klasifikavimo tikslumas – 83,25 %. Jis buvo pasiektas klasifikuojant duomenis tik į 8-ias klases ir naudojant Bajeso klasifikatorių.
6. Bajeso klasifikatoriaus tikslumas, palyginus su kitais klasifikatoriais, staigiai mažėja, kai galimų klasių skaičius didėja.



17 pav. Klasifikavimo tikslumas kintant galimų klasių skaičiui

5.4. Klasifikavimas kintant mokymo ir testavimo duomenų apimties santykiui

Šiame skyrelyje aprašomas eksperimentas, kurio metu įvertinamas kiekvieno klasifikavimo modelio klasifikavimo tikslumas, kai kinta mokymo ir testavimo duomenų apimties santykiai.

5.4.1. Duomenų aibė

Eksperimento metu buvo naudojama duomenų aibė, kuri buvo surinkta pasinaudojant savadarbiu scenarijumi parašytu „Python“ programavimo kalba. Parašytas scenarijus renka tam tikrų kategorijų skelbimų antraštes iš penkių skelbimų tinklapių „plius.lt“, skelbiu.lt“, skelbimai.lt“, rinka.lt“ ir „alio.lt“. Duomenų aibę iš viso sudarė 8 000 skelbimų antraščių suskirstytų į 8-ias kategorijas.

Eksperimento metu, tekstiniai duomenys buvo klasifikuojami kintant mokymo ir testavimo duomenų apimčių santykiams (50:50, 40:60, 70:30, 20:80, 10:90, 5:95).

5.4.2. Tyrimo rezultatai

Duomenų aibės klasifikavimo tikslumo įverčiai pateikti 4 lentelėje ir 18 paveiksle. 4 lentelėje didesni įverčiai paryškinti kita spalva.

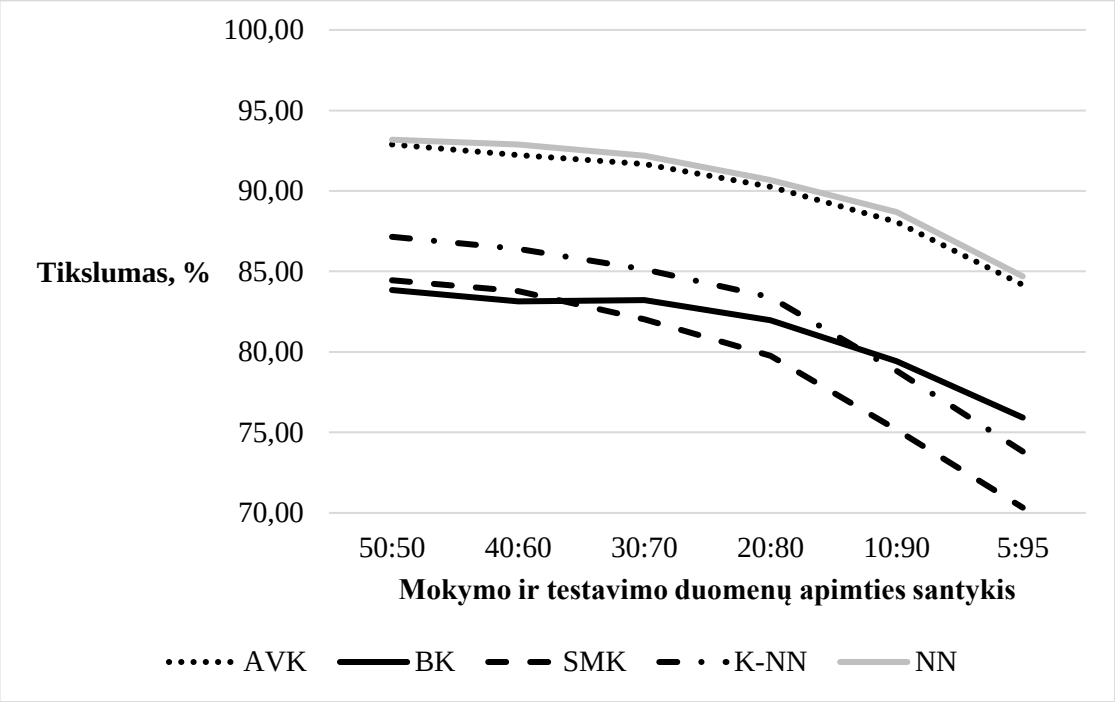
4 lentelė. Klasifikavimo tikslumas kintant mokymo ir testavimo duomenų apimties santykiui

Santykis	AVK	BK	SMK	K-NN	NN
50:50	92,89 %	83,84 %	84,45 %	87,15 %	93,18 %
40:60	92,22 %	83,15 %	83,77 %	86,40 %	92,87 %
30:70	91,68 %	83,22 %	82,04 %	85,12 %	92,18 %
20:80	90,26 %	81,96 %	79,77 %	83,39 %	90,68 %
10:90	88,06 %	79,42 %	75,16 %	78,82 %	88,69 %
5:95	84,16 %	75,92 %	70,33 %	73,82 %	84,69 %

Išnagrinėjus eksperimento metu gautus tikslumo įverčius, matome, kad:

1. Apdorojant skirtingos apimties mokymo ir testavimo duomenis, visais atvejais, neuroninio tinklo klasifikatorius parodė geriausius rezultatus, pasiekdamas didžiausią tikslumą.
2. Didžiausias klasifikavimo tikslumas (93,18 %) buvo pasiektas klasifikuojant duomenis, naudojant neuroninio tinklo klasifikatorių, kai mokymo ir testavimo duomenų santykis buvo lygus – 50:50.
3. Pagal klasifikavimo tikslumą, antras geriausias klasifikatorius – atraminių vektorių klasifikatorius, kuris visais atvejais mažai atsiliko nuo neuroninio tinklo klasifikatoriaus rezultatų.

4. Prasčiausius rezultatus parodė sprendimų medžio klasifikatorius, kuris dauguma atvejų pasiekdavo mažiausią klasifikavimo tikslumą palyginus su kitais klasifikatoriais.
5. Mažiausias klasifikavimo tikslumas (83,25 %) buvo pasiektas klasifikuojant duomenis, naudojant sprendimų medžio klasifikatorių, kai mokymo ir testavimo duomenų santykis buvo lygus – 5:95.



18 pav. Klasifikavimo tikslumas kintant mokymo ir testavimo duomenų apimties santykiui

5.5. Klasifikavimas apdorojant skirtingų kalbų duomenis

Šiame skyrelyje aprašomas eksperimentas, kurio metu yra įvertinamas kiekvieno klasifikavimo modelio klasifikavimo tikslumas, kai apdorojami skirtingų kalbų tekstiniai duomenys.

5.5.1. Duomenų aibė

Eksperimento metu buvo naudojamos dvi duomenų aibės:

- Tekstiniai duomenys lietuvių kalba.
- Tekstiniai duomenys anglų kalba.

Tekstiniai duomenys lietuvių kalba – duomenys buvo surinkti pasinaudojant savadarbiu scenarijumi parašytu „Python“ programavimo kalba. Parašytas scenarijus renka tam tikrų kategorijų skelbimų antraštes iš penkių skelbimų tinklapių „plius.lt“, skelbiu.lt“, skelbimai.lt“, rinka.lt“ ir „alio.lt“. Duomenų aibę iš viso sudarė 8 000 skelbimų antraščių suskirstytų į 8-ias kategorijas.

Tekstiniai duomenys anglų kalba – duomenys, kuriuos sudaro įvairių temų antraštės iš tinklapio „Stackoverflow“. Duomenų aibę iš viso sudarė 8 000 antraščių suskirstytų į 8-ias kategorijas.

Eksperimento metu, tekstiniai duomenys buvo klasifikuojami apdorojant skirtingų kalbų tekstinius duomenis (LT – lietuvių k., EN – anglų k.). Kiekvieno bandymo metu, duomenų imtis buvo padalinta į dvi dalis: mokymo ir testavimo duomenis santykiu 80:20.

5.5.2. Tyrimo rezultatai

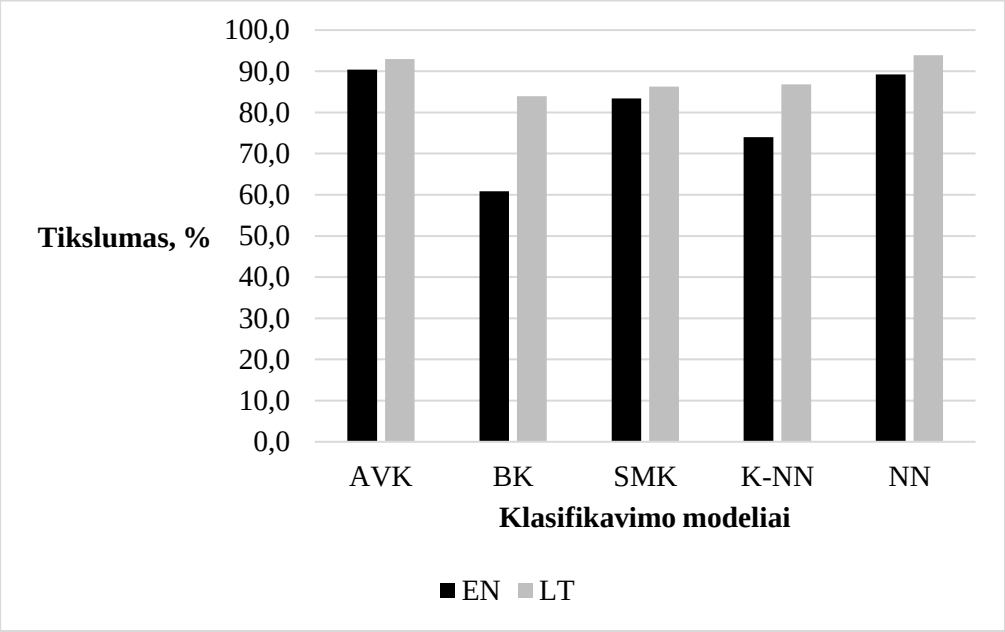
Duomenų aibės klasifikavimo tikslumo įverčiai pateikti 4 lentelėje ir 19 paveiksle. 4 lentelėje didesni įverčiai paryškinti kita spalva.

5 lentelė. Klasifikavimo tikslumas apdorojant skirtingų kalbų duomenis

Kalbos	AVK	BK	SMK	K-NN	NN
EN	90,36 %	60,82 %	83,39 %	73,97 %	89,20 %
LT	92,98 %	83,92 %	86,25 %	86,84 %	93,92 %

Išnagrinėjus eksperimento metu gautus tikslumo įverčius, matome, kad:

1. Apdorojant tekstinius duomenis lietuvių kalba, didžiausią klasifikavimo tikslumą (93,92 %) pasiekė neuroninio tinklo klasifikatorius.
2. Apdorojant tekstinius duomenis anglų kalba, didžiausią klasifikavimo tikslumą (90,36 %) pasiekė atraminių vektorių klasifikatorius.
3. Prasčiausius rezultatus pademonstravo Bajeso klasifikatorius. Apdorojant dviejų skirtingų kalbų tekstinius duomenis šis klasifikatorius pasiekė mažiausią tikslumą palyginus su kitų klasifikatorių rezultatais. Klasifikuojant duomenis lietuvių kalba – 83,92 %, anglų kalba – 60,82 %.



19 pav. Klasifikavimo tikslumas apdorojant skirtingų kalbų duomenis

REZULTATAI IR IŠVADOS

Atliekant darbą, buvo pasiekti šie rezultatai:

1. Išnagrinėjus skirtingus duomenų požymių išskyrimo metodus, buvo realizuoti trys teksto požymių išskyrimo metodai: TF-IDF, TF-IDF LSA ir TF-IDF LDA.
2. Išnagrinėjus skirtingus duomenų klasifikavimo modelius, buvo realizuoti penki teksto klasifikavimo modeliai: atraminių vektorių klasifikatorius, bajeso klasifikatorius, sprendimų medžio klasifikatorius, k-artimiausių kaimynų klasifikatorius ir neuroninių tinklų modelis.
3. Sukurtas teksto klasifikatorius buvo panaudotas tyrimams atlikti.
4. Atlikus tyrimą, palyginus skirtingus duomenų požymių išskyrimo metodus, buvo nustatytas metodas, leidžiantis klasifikatoriui pasiekti didžiausią tikslumą.
5. Atlikus tyrimą, palyginus skirtingus teksto klasifikavimo modelius, buvo nustatyti modeliai, kurie demonstruoja geriausius rezultatus klasifikuojant didelės apimties duomenis.

Iš gautų tyrimo rezultatų galima daryti tokias išvadas:

1. TF-IDF statistikos metodu išskirti teksto požymiai geriausiai apibrėžia tekstą ir leidžia dažniausiai klasifikavimo modeliui pasiekti geriausius rezultatus palyginus su kitais požymių išskyrimo metodais (TF-IDF LSA, TF-IDF LDA).
2. Klasifikuojant mažos apimties duomenis, TF-IDF ir TF-IDF LSA požymių išskyrimo metodai veikia panašiai, nepriklausomai nuo duomenų įvairovės.
3. TF-IDF LSA ir TF-IDF LDA požymių išskyrimo metodai demonstruoja panašius gerus rezultatus kaip ir TF-IDF metodas, kai klasifikatoriui pateikiami didesni duomenų kiekiai mokymui.
4. Klasifikuojant didelės apimties duomenis, neuroninių tinklų modelis leidžia pasiekti didžiausią tikslumą (96,42 %).
5. Klasifikuojant trumpus tekstus į dvi klases, neuroninių tinklų modelis demonstruoja geriausius rezultatus (99,55 %), kai antrą geriausią rezultatą pateikia atraminių vektorių klasifikatorius (98,56 %).
6. Klasifikuojant duomenis į keletą ir daugiau klasių, geriausius klasifikavimo rezultatus pateikia neuroninių tinklų modelis (96,07 %).
7. Klasifikuojant trumpus tekstus anglų kalba, geriausus rezultatus pateikia atraminių vektorių klasifikatorius (90,36 %).
8. Klasifikuojant trumpus tekstus lietuvių kalba, didžiausią tikslumą pasiekia neuroninio tinklo klasifikatorius (93,92 %).

9. Daugelyje tyrimo atvejų neuroninio tinklo klasifikatorius demonstravo geriausius rezultatus pasiekdamas didžiausią klasifikavimo tikslumą konkuruodamas su atraminių vektorių klasifikatoriumi, kuris dažnai pagal tyrimo rezultatus užimdavo antrą vietą.

Šiame darbe išnagrinėti ne visi teksto požymių išskyrimo metodai ir teksto klasifikavimo būdai. Darbe buvo sutelktas dėmesys į požymių išskyrimo algoritmus, kurių išvestis – fiksuoto ilgio skaičių vektoriai. Kaip viena iš alternatyvų, teksto požymiams išskirti, gali būti taikomas „Word2Vec“ algoritmas. Šis požymių išskyrimo metodas išgauna tekste esančių žodžių savybes, atsižvelgiant į šalio žodžio esančius terminus. Tokiu būdu išgautos teksto žodžių reprezentacijos labiau artimos žodžio semantikos apibūdinimui.

Taikant „Word2Vec“ algoritmą, išgaunamos tekstinių duomenų savybės, kurios reikalauja kitokios, sudėtingesnės struktūros klasifikavimo modelių. Vienas iš tokių modelių yra konvoliuciniai neuroniniai tinklai, kurie savo struktūra yra sudėtingesni nei daugiasluoksniai perceptronai, tačiau leidžia išmokti sudėtingesnės struktūros duomenų savybes.

LITERATŪROS SĄRAŠAS

1. Aggarwal, C. C., & Zhai, C. X. (2012). *Mining Text Data.* New York: Springer.

2. Alpaydin, E. (2009). *Introduction to Machine Learning.* The MIT Press.

3. Bandyopadhyay, S., & Saha, S. (2012). *Unsupervised Classification: Similarity Measures, Classical and Metaheuristic Approaches, and Applications.* Springer.

4. Banerjee, S. (2007). Boosting Inductive Transfer for Text Classification using Wikipedia.

5. Beam, A. L. (2017). *Deep Learning 101 - Part 1: History and Background*. Prieiga per internetą: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

6. Berniukevičius, A., & Kurilovas, E. (2017). *Dirbtiniai neuroniniai tinklai personalizuotam mokymuisi*. Prieiga per internetą: https://www.mii.lt/LMR/B/2017/58B04.pdf

7. Brownlee, J. (2019). *A Gentle Introduction to the Bag-of-Words Model*. Prieiga per internetą: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

8. Corey, A. H., Claudio, R., & Alfaro, R. (2010). Automated Text Binary Classification Using Machine Learning Approach.

9. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks.

10. Dabbura, I. (2018). *Coding Neural Network—Forward Propagation and Backpropagtion*. Prieiga per internetą: https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76

11. Fletcher, T. (2009). Support Vector Machines Explained.

12. Ganesan, K. (2019). *All About Stop Words for Text Mining and Information Retrieval.* Prieiga per internetą: http://kavita-ganesan.com/feed-items/all-about-stop-words-for-text-mining-and-information-retrieval/#.XKJXjeszYWo

13. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks.

14. Góralewicz, B. (2019). *The TF*IDF Algorithm Explained*. Prieiga per internetą: https://www.onely.com/blog/what-is-tf-idf/

15. Hackeling, G. (2017). Mastering Machine Learning with scikit-learn.

16. Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., & Suleyman, M. (2015). Teaching Machines to Read and Comprehend.

17. Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences.

18. Jiang, E. P. (2010). Content-Based Spam Email Classification using Machine-Learning Algorithms. Esantis *Text Mining: Applications and Theory* (p. 37-56).

19. Joachims, T., & Radlinski, F. (2007). Search Engines that Learn from Implicit Feedback.

20. Jurafsky, D., & Martin, J. H. (2014). *Speech and Language Processing.* New Jersey: Pearson Prentice Hall.

21. Kantrowitz, M., Mohit, B., & Mittal, V. (2000). Stemming and its effects on TFIDF Ranking.

22. Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges.

23. Kim, K. K., Kim, K. I., Kim, J. B., & Kim, H. J. (2000). Learning-based approach for license plate recognition.

24. Kolesau, A., Šešok, D., & Rybokas, M. (2018). A Character-Based Part-of-Speech Tagger with Feedforward Neural Networks.

25. Kramer, O. (2013). *Dimensionality Reduction with Unsupervised Nearest Neighbors.* Springer.

26. Landauer, T. K., Foltz, P. W., & Laham, D. (2019). *An Introduction to Latent Semantic Analysis.* Prieiga per internetą: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf

27. Li, Y., Tripathi, A., & Srinivasan, A. (2016). Challenges in Short Text Classification: The Case of Online Auction Disclosure.

28. Logan, S. (2017). *Understanding the Structure of Neural Networks*. Prieiga per internetą: https://becominghuman.ai/understanding-the-structure-of-neural-networks-1fa5bd17fef0

29. Maimon, O., & Rokach, L. (2005). Data Mining and Knowledge Discovery Handbook.

30. Makūnaitė, R. (2006). *Neuroniniai tinklai*. Prieiga per internetą: http://www.elektronika.lt/teorija/kompiuterija/4342/neuroniniai-tinklai/

31. Maren, A. J., Harston, C. T., & Pap, R. M. (2014). *Handbook of Neural Computing Applications.* San Diego: Academic Press Inc.

32. McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity.

33. Mikheev, A. (2018). *Text Segmentation.* Longman.

34. Mitchell, T. M. (1997). Machine Learning.

35. Nielsen, M. (2018). Neural Networks and Deep Learning.

36. Pascanu, R., Mikolov, T., & Bengio , Y. (2013). On the difficulty of training recurrent neural networks.

37. Ramchoun, H., Idrissi, M. A., Ghanou, Y., & Ettaouil, M. (2016). Multilayer Perceptron: Architecture Optimization and Training.

38. Raschka, S. (2019). *Linear Discriminant Analysis – Bit by Bit*. Prieiga per internetą: https://sebastianraschka.com/Articles/2014_python_lda.html

39. Raudys, Š. (1998). Evolution and generalization of a single neurone: I. Single-layer perceptron as seven statistical classifiers.

40. Rish, I. (2001). An empirical study of the naive Bayes classifier.

41. Rosten, E., & Drummond, T. (2006). Machine Learning for High-Speed Corner Detection.

42. Rush, A. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization.

43. Russell, S. J., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach.* New Jersey: Pearson Prentice Hall.

44. Sammut, C., & Webb, G. I. (2017). Classification Learning. Esantis *Encyclopedia of Machine Learning and Data Mining* (p. 209).

45. Sargelis, K. (2009). *Klaidos Skleidimo Atgal Algoritmo Tyrimai*. Prieiga per internetą: gs.elaba.lt/object/elaba:1742648/1742648.pdf

46. Sehgal, R. (2013). *Low Cost Point of Care Testing for Cognitive Motor Aspects Using Electroencephalography.*

47. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., & Hubert, T. (2017). Mastering the game of Go without human knowledge.

48. Simon, P. (2013). Too Big to Ignore: The Business Case for Big Data. 89.

49. Singh, J. (2018). Challenges in Text Classification using Machine Learning Techniques.

50. Stabingienė, L. (2014). *Dirbtiniai neuroniniai tinklai (DNT)*. Prieiga per internetą: http://www.ilab.lt/stabingiene/sk2_2.html

51. Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining.* Pearson Education, Inc.

52. Tretyakov, K. (2004). Machine Learning Techniques in Spam Filtering.

53. Trim, C. (2019). *The Art of Tokenization*. Prieiga per internetą: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en

54. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). *Data Mining – Practical Machine Learning Tools and Techniques.* Cambridge: Elsevier.

55. Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., & Macherey, W. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

56. Xu, K., Ba, J., Kiros, R., Cho, K., & Courville, A. (2016). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.