TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Brenda Uga 143067IAPB

TTÜ TUDENGITE VÄLJALANGEMISE ENNUSTAMINE: TÕENÄOSUSE ARVUTAMINE MASINÕPPE MEETODITE ABIL NING TULEMUSTE KUVAMINE VEEBIRAKENDUSES

Bakalaureusetöö

Juhendaja: Ago Luberg
MSc

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Brenda Uga

22.05.2017

Annotatsioon

Käesoleva töö eesmärgiks on töötada välja süsteem, mis võimaldab kasutajal ennustada tudengite väljalangevust ning analüüsida kasutatavast andmestikust tulenevaid põhjuseid väljalangemise riskigruppi kuulumise kohta. Töö käigus luuakse masinõppe mudelid kasutades informaatika õppekava tudengite soorituste andmeid ning üldisikuandmeid. Mudelite ennustuse tulemuste kuvamiseks ja analüüsimise lihtsustamiseks luuakse ka veebirakendus.

Töö tulemusena realiseeritud masinõppe mudelid täidavad ennustamise täpsuse jaoks seotud eesmärki. Rakendus võimaldab kasutajal valida seitsme algoritmi ning erinevate andmehulkade kasutamise vahel. Kõigil katsetel erinevate andmehulkade ja algoritmidega on mudelite ennustustäpsus vahemikus 60 - 95%. Katsete tulemusi on võimalik vaadata veebirakenduses, milles kuvatakse tulemused kasutajasõbralikul ja mugaval viisil eri tüüpi tabelite ja graafikute abil.

Antud töö tulemuste põhjal võib järeldada, et mudelid on piisavalt täpsed, et leiaksid rakendust teaduskonnas tudengite väljalangemise ennetamisel. Kasutust leiaksid nii ennustuse tulemused iga tudengi kohta kui ka andmestikust leitud kõige mõjukamate tunnuste info.

Rakendus on realiseeritud kasutades Pythonit ning kogu koodibaas on avalikustatud autori Githubi lehel aadressil https://github.com/BrendaUga/BrendaUga.github.io.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 6 peatükki, 10 joonist, 2 tabelit.

Abstract

Predicting dropouts among TUT students: calculating probabilities using machine learning and displaying results in a web application

The purpose of this thesis is to develop a system, which allows the user to predict student dropout and analyse the data-dependent reasons that determine whether the student belongs to the dropout risk group or not. As the result, author creates machine learning models using grades and general person data of each Informatics program student. For the purpose of displaying results and facilitating analysing, a web application is also developed.

Machine learning models created as a result all fulfil the goal set regarding prediction accuracy. The application lets the user choose between seven different algorithms and different datasets. All experiments with different datasets and algorithms show that the models predict with accuracy between 60 to 95%. The prediction results can be viewed in the web application, which uses various types of tables and graphs to showcase the results in a user-friendly way.

Based on the results, it can be said that the models are precise enough to be applied at the faculty to prevent student dropouts. Both the results for each student and the importances of features from the data can be used for prevention of dropouts.

The application is realized using Python and the codebase is published here https://github.com/BrendaUga/BrendaUga.github.io.

The thesis is in Estonian and contains 39 pages of text, 6 chapters, 10 figures, 2 tables.

Lühendite ja mõistete sõnastik

Above the fold	Osa ekraanist, mis on ilma kerimiseta nähtaval
Back-end	Serveripoolne osa arhitektuurist, mis tegeleb äriloogika ja andmetega
Cross validation	Valideerimistehnika, mis analüüsib kui hästi mudel oskab üldistada tundmatul andmehulgal
CSV	Failitüüp, milles iga rida koosneb komadega eraldatud väljadest
Dashboard	Ülevaatlik vaade, mis näitab põhikasutusjuhu jaoks kõige olulisemat infot
Dataframe	Pandas teegi tabelilaadne kahedimensiooniline andmestruktuur, mis sisaldab erinevat tüüpi nimelisi veerge
Information gain	Entroopia vähenemise hulk pärast puu hargnemist mingi atribuudi järgi
JSON	Formaat andmete edastamiseks, mis sisaldab teksti kujul liste ning atribuut-väärtus paare
Kasutatavus	Asja või teenuse kasutamise lihtsus ja kergesti õpitavus
Kernel	Funktsioon, mis võimaldab andmed teisendada kõrgema dimensiooniga ruumi ilma koordinaate ruumis arvutamata
SVG	XML-põhine vektorgraafika kujutamise formaat
Üle- ja alasämplimine	Tehnikad andmetes klasside ebatasakaalu vähendamiseks rohkem esineva klassi objektide hulga vähendamise ja vähemesineva klassi objektide hulga suurendamise teel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tallina Tehnikaülikoolis on aastate jooksul olnud probleemiks tudengite väljalangevus. Igal aastal langeb ligi 20% tudengeid ennetähtaegselt koolist välja. Tallinna Tehnikaülikooli huvides on seda protsenti vähendada, kuid selleks puudub hetkel käepärane ja kasulik süsteem, millega riskigruppi kuuluvaid tudengeid tuvastada [5].

Antud töö eesmärk on luua süsteem, mis võimaldab kasutajal tuvastada soorituste ning üldandmete põhjal need tudengid, keda kõige tõenäolisemalt ohustab väljalangemine. Selleks seab autor järgmised eesmärgid:

- Luua masinõppe meetodil mudelid, mis võimaldavad ennustada tudengi kuulumist riskigruppi.
- Valideerida mudelid kasutades TTÜ tudengite anonümiseeritud sooritusi ning üldandmeid sisaldavat andmestikku.
- Luua mudelite kasutamiseks ning tulemuste visualiseerimiseks veebirakendus.

Töö tulemusena valmib veebirakendus, mis pakub kasutajale võimalust treenida uusi mudeleid või ennustada olemasolevate pealt, kasutades enda üleslaetud andmestikke. Ennustuse tulemusi kuvab rakendus välja erinevat tüüpi graafikute ja tabelite abil, andes kasutajale võimaluse saada nii kiiret ja mugavat ülevaadet kui ka detailset infot iga andmestikus esineva tudengi kohta.

Selline tööriist on kasulik vahend leidmaks üles tudengid, keda ohustab kõige tõenäolisemalt väljalangemine. Tööriista rakendamiseks on vajalik, et mudelid on piisavalt täpsed ning autor seab endale eesmärgiks viia mudelite täpsused vähemalt 80%-ni. Nii on kasutajatel võimalik piisava kindlusega leida üles riskigruppi kuuluvad tudengid ning rakendada ennetavaid meetmeid.

2 Teoreetilised alused

Antud peatükis antakse ülevaade masinõppe teoreetilistest alustest ning kasutatud tehnikate ja algoritmide kirjeldusest.

2.1 Masinõppest

Masinõpe on arvutiteaduse valdkond, mis tegeleb selliste algoritmide loomisega, mis võimaldavad andmete põhjal teha otsustusi ja ennustusi. Arthur Samueli (1959) definitsiooni kohaselt on masinõpe „teadusvaldkond, mis annab arvutitele võime õppida ilma neid selleks otstarbeks selgelt programmeerimata“.

Meetodid, mida kasutatakse masinõppes, võib jagada probleemi olemuse põhjal kaheks. Kui tegemist on probleemiga, mille ennustamisel on tegu etteantud väärtuste leidmisega, on tegu juhendatud õppimisega. Vastasel juhul, kui andmehulga pealt leitakse seoseid ning ennustatakse mõnda väärtust, mida pole ette antud, on tegu juhendamata õppimisega. Viimast kasutatakse tihti klasterdamisel, esimest klassifitseerimisel.

Lisaks saab jagada juhendatud õppe probleeme vastavalt oodatavatele väärtustele. Klassifitseerimisprobleemide korral on ennustatavad väärtused diskreetsed (näiteks klassid), regressiooniprobleemide puhul on oodatavad väärtused pidevad (näiteks hinnad ja mõõdud). Antud probleemi puhul on tegu klassifitseerimisega, kuna ennustame tudengi lõpetamist, mida saab väljendada binaarse väärtusega 0 (ei lõpeta) või 1 (lõpetab).

Masinõppe protsessi võib üldistada kolme faasiga: treenimine, arendamine, hindamine. Treenimise etapis leitakse treenimiseks mõeldud andmehulgal põhinedes mudeli parameetrid, mis võimaldavad ennustada võimalikku väärtust. Arendamise faasis on võimalik häälestada erinevaid hüperparameetreid, mis mõjutavad mudeli käitumist ja täpsust. Hindamise faasis toimub mudeli hindamine testimiseks mõeldud andmehulgaga [10, lk 11-13].

2.2 Cross validation

Mudelite loomisel on oluline hinnata kui hästi mudel töötab tundmatul andmehulgal. Mudel käitumise hindamisel võib täheldada kahesugust käitumist – alasobitumist (*underfitting*) ja ülesobitumist (*overfitting*). Alasobitumise korral on mudel kallutatud, sest see on liiga lihtsakoeline. Ülesobitumise puhul on mudel liiga kompleksne, tingides hea soorituse õppimise andmehulgal, sest mudel näeb hästi mustreid sellel andmehulgal. Sama mudel ei suuda aga testandmehulgal piisavalt üldistada. Hea tasakaalu ala- ja ülesobitumise vahel annab *cross validation'*i kasutamine.

*Cross validation'*i meetoditest vaatleb autor kahte - *holdout* ja *k-fold*.

Holdout cross validation on tehnika, kus mudeli treenimiseks kasutatav andmehulk jagatakse omakorda õppimise ja testimise andmehulgaks. Treenimise andmehulka kasutatakse mudeli õpetamiseks ning testimise andmehulgal tehakse hüperparameerite täpsustamine. Sellisel juhul aga õpib mudel ka testimise andmehulga pealt, kui hüperparameetreid täpsustatakse seda andmehulka kasutades. Sellel põhjusel on kasulikum luua eraldi valideerimise andmehulk, mida ei kasutata parameetrite valimisel. Valideerimise andmehulga kasutamine vähendab ka ülesobitumise võimalust, sest mudeli treenimiseks kasutatakse teisi andmehulkasid kui ennustamiseks. *Holdout* tehnikal on miinuseks kõrgem sõltuvus sellest, kuidas jagatakse andmed alamhulkadeks ning millised objektid jäävad millisesse alamhulka.

K-fold cross validation tehnikas jagatakse samuti treenimise andmehulka alamhulkadeks. Siin tehakse seda juhuslikult, jagades andmehulga *k*-ks alamhulgaks, millest *k-1* alamhulka kasutatakse mudeli treenimiseks ja ühte testimiseks. Sellist jagamist korratakse *k* korda. Nii saadakse mudeli täpsus *k* alamhulga mudelite täpsuste keskmisena, mis on vähem tundlikum andmete jagamisel tehtavatele valikutele [10, lk 173-178].

2.3 Hüperparameetrite täpsustamine

Masinõppe puhul on kahesuguseid parameetreid - selliseid, mis mudel treenimise käigus ise leiab, näiteks kaalud logistilise regressiooni puhul, ja selliseid, mis tuleb kasutajal endal optimeerida, näiteks abivektormasina (*support vector machine*) puhul kerneli tüüp.

Viimast nimetatakse ka hüperparameetriteks, mille optimeerimine toimub mudeli arendamisel [10, lk 185].

Antud töös kasutatakse hüperparameetrite optimeerimiseks võrkotsingut (*grid search*), mis leiab parima kombinatsiooni etteantud parameetrite võrgustikust. Selleks on ette antud hulk parameetreid ja nende võimalikke väärtusi, millest võrkotsing kombineerimise teel leiab parima tulemuse tagava kombinatsiooni. Võrkotsingut saab kasutada ka koos *k-fold cross validation*'iga, mille puhul toimub erinevate parameeterkombinatsioonide katsetamine igal *k-fold* iteratsioonil.

2.4 Tunnuste valimine

Masinõppe mudelite puhul ülesobitumise vältimiseks on võimalik mudeli kompleksust vähendada, kasutades selleks tunnuste valimist. Tunnuste valimine võimaldab üles leida sellised andmestiku veerud, mis mõjutavad mudeli ennustust kõige rohkem. Selline tegutsemine võimaldab vähendada nii vajalikku arvutusvõimsust kui ka vähendada mudeli sõltuvust mürast ja ebaolulistest tunnustest, mis omakorda tingib väiksema võimaluse ülesobitumiseks [10, lk 112].

Sobiv arv kasutatavate tunnuste jaoks sõltub kasutatavast andmehulgast ja probleemist. Siiski on siin võimalik kasutada üldisemat rusikareeglit, mida võtta aluseks enne parameetrite optimeerimist oma probleemile vastavaks. Rusikareegli kohaselt on piisavalt heade tulemuste saavutamiseks piisav $\sqrt{n}$ tunnust, kus *n* on kõigi tunnuste arv.

Tunnuste valimist saab teostada mitmel viisil. Tunnuste valimine on implementeeritud regulariseerimisel (*regularization*), hõredatel klassifitseerijatel (näiteks logistiline regressioon, abivektorklassifitseerija) ja ka otsustuspuudel põhinevatel algoritmidel tunnuste tähtsuste kaudu. Antud töös kasutatakse tunnuste valimise meetodina ühemõõtmelistel statistilistel testidel põhineva `SelectKBest` meetodit Scikit-learn teegist, sest antud meetod leiab üles nii mitu tunnust, kui arendaja määrab. Antud juhul määratakse soovitud tunnuste arvuks $\sqrt{n}$ tunnust.

Scikit-learn teegi `SelectKBest` meetod implementeerib tunnuste valimist, leides *k* kõige rohkem mudelit mõjutavat tunnust. Kasutades `SelectKBest` meetodit koos *chi2*

skoorifunktsiooniga, praagitakse välja tunnused, millel puudub kõige tõenäolisemalt seos klassiga ning mis on seetõttu klassifitseerimisprobleemi jaoks kasutud [11].

2.5 Täpsuse hindamise meetodid

Mudeli täpsuse hindamiseks kasutatakse mitmeid meetodeid. Kõigi nende eesmärgiks on hinnata mudeli ennustatud väärtuste ja tegelike väärtuste erinevusi. Oluline on üles leida väärpositiivsed ja väärnegatiivsed väärtused. Tudengite väljalangemise probleemi juures on kõige tähtsam vähendada väärpositiivseid väärtuseid, sest sellises olukorras jäävad osad potentsiaalsed väljalangejad märkamata. Antud juhul vastab positiivne väärtus 1 tudengi lõpetamisele ja negatiivne väärtus 0 tudengi mitte lõpetamisele.

Mudelite käitumise hindamisel on olulised meetrikad täpsus (*precision*) ja saagis (*recall*). Täpsus näitab millise hulga moodustavad õigesti klassifitseeritud positiivsed väärtused tõeliste positiivsete ja väärpositiivsete väärtustest, nagu järeldub antud valemist (1).

$$PRE = \frac{TP}{TP+FP} \tag{1}$$

Valemis (1) on TP tõeliste positiivsete väärtuste hulk (*true positives*) ning FP on valesti klassifitseeritud positiivsete (*false positives*) ehk väärpositiivsete väärtuste hulk [10, lk 192].

Saagis näitab kui palju moodustavad õigesti klassifitseeritud tõelised positiivsed väärtused kogu positiivsete väärtuste hulgast nagu näidatud valemis (2).

$$REC = \frac{TP}{FN+TP} \tag{2}$$

Valemis (2) on TP tõeliste positiivsete väärtuste hulk ning FN on valesti klassifitseeritud negatiivsete (*false negatives*) ehk väärnegatiivsete väärtuste hulk.

Täpsus ja saagis on omavahel tihti pöördvõrdelises seoses. Sellist olukorda nimetatakse täpsuse/saagise kompromissiks. Sellisel juhul võib kasutada mudeli hindamiseks täpsuse ja saagise kombinatsiooni, mida nimetatakse f1-skooriks nagu näidatud valemis (3).

$$F1 = 2\,\frac{PRE\ x\ REC}{PRE+REC} \tag{3}$$

Valemis (3) on PRE täpsus ja REC saagis [10, lk 192].

Täpsuse ja saagise arvutamiseks vajalike mõistete sisu seletab lahti Joonis 1. Tabelis on kirjeldatud väärpositiivsete, väärnegatiivsete, tõeliste positiivsete ja tõeliste negatiivsete väärtuste sõltuvus tegelikust väärtusest ning ennustatud väärtusest.

		ennustatud seisund	
		ennustus positiivne	ennustus negatiivne
tõeline seisund	seisund positiivne	tõeline positiivne (TP)	väärpositiivne (FP)
	seisund negatiivne	väärnegatiivne (FN)	tõeline negatiivne (TN)

Joonis 1. Eksimismaatriks.

Algoritmide soorituste hindamisel kasutatakse ka õigsuse (*accuracy*) meetrikat. Õigsus näitab mitu ennustatud väärtust langeb kokku tõeliste väärtustega ehk tõeliste positiivsete ja negatiivsete hulka kõigist objektide hulgast. Valemis 4 on näidatud õigsuse arvutamise valem.

$$ACC = \frac{TP+TN}{(TP+FP+TN+FN)} \quad (4)$$

Sellist hindamisviisi saab kasutada, kui on olemas andmed tudengite tegeliku lõpetamise või väljakukkumise kohta.

Algoritmide soorituste hindamiseks ja võrdlemiseks antud töös kasutatakse eelnevalt mainitud viise.

2.6 Algoritmide kirjeldused

Siinkohal toob autor välja kasutatud algoritmide kirjeldused ning nende tööpõhimõtted.

2.6.1 Logistiline regressioon

Logistiline regressioon on klassifikatsioonimudel, kus ennustatav muutuja on kategooriline (näiteks inimese sugu, veregrupp või rahvus). Logistilist regressiooni kasutatakse peamiselt mingi sündmuse toimumise tõenäosuse uurimiseks ja selle mingist pidevast muutujast sõltumise uurimiseks. Antud mudel sobib hästi, kui on vaja ennustada binaarset väljundit (näiteks tudengi aine läbimine/mitteläbimine, haiguse avaldumine/mitteavaldumine) mitte pidevat väljundi väärtust, milleks sobib paremini

lineaarne regressioon. Binaarse väljundi ennustamisel on tegemist binomiaalse logistilise regressiooniga.

Logistilise regressiooni puhul arvutatakse sündmuse tõenäosus positiivse sündmuse jaoks, ehk juhuks, kus määratav väljund on "1". Selleks kasutatakse logistilist funktsiooni, et leida tõenäosus positiivse väljundi jaoks sõltuvalt muutujast X. Saadud tõenäosuse puhul määratakse väljundiks "1", kui positiivse sündmuse juhtumise tõenäosus on suurem kui negatiivse sündmuse juhtumise tõenäosus, vastasel juhul määratakse väljundiks "0".

Seega võib logistilist regressiooni võtta kui regressioonimudelit, kus ennustatav tõenäosus on pideval kujul, või klassifitseerimismudelit, kus ennustatav binaarne väärtus on diskreetsel kujul.

2.6.2 K-lähima naabri klassifikatsioon

K-lähima naabri ehk kNN algoritmi (*k-nearest neighbors*) saab võtta nii klassifitseerimis- ka kui regressioonimudelina. Igal konkreetselt juhul sõltub väljund sellest, kas kasutatakse klassifitseerimise või regressioonimudelit. Klassifitseerimise korral on väljundiks objekti kuuluvus mingisse klassi, regressiooni puhul objekti mingi omaduse väärtus.

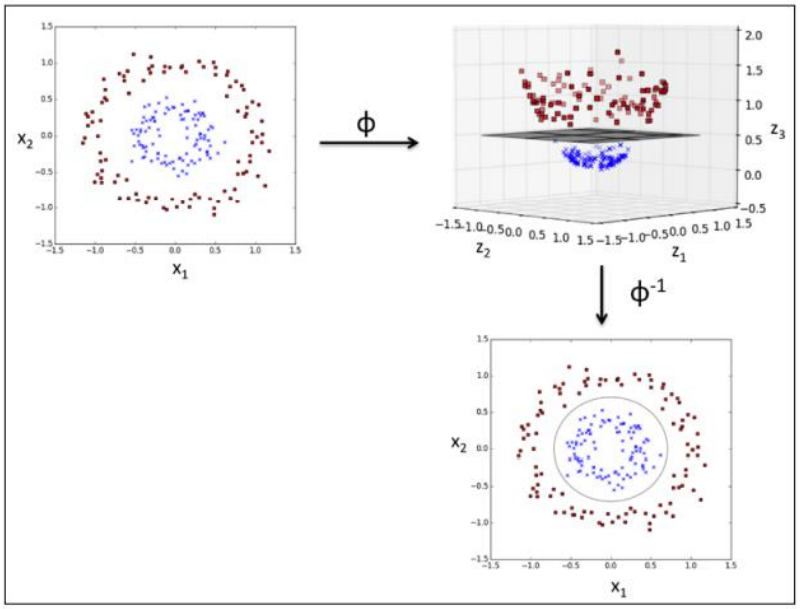
K-lähima naabri klassifitseerimisel määratakse ennustatava objekti klass tema k lähima naabri klasside põhjal. Nimelt saab objekti klassiks tema lähimate naabrite klasside seast kõige rohkem esinev klass. Regressiooni puhul saab ennustatava objekti väärtuseks tema lähimate naabrite väärtuste keskmine. Siit ilmneb ka antud algoritmi üks puudus, milleks on tundlikkus objektide lokaalsusele ja arvule. Näiteks juhul kui klassi A kuuluvaid objekte on tuntavalt rohkem kui klassi B kuuluvaid objekte, on algoritmi klasside määramine kallutatud rohkem esineva klassi poole. Selle probleemi üheks võimalikuks lahenduseks on objektide kaalumine nende distantsi põhjal. Kaalude määramisel kasutatakse näiteks pöördväärtust objektide distantsist testitava objektini. Sellisel viisil määratakse lähimatele naabritele suuremad kaalud kui kaugematele naabritele [7].

Oluliseks parameetriks kNN puhul on k. k väärtusest sõltub mil määral arvestab mudel esineva müraga ning kui täpsed on klasside vahelised piirid. Suurem k väärtus vähendab müra osatähtsust, kuid muudab klasside piirid hägusemaks [9].

2.6.3 Abivektormasin

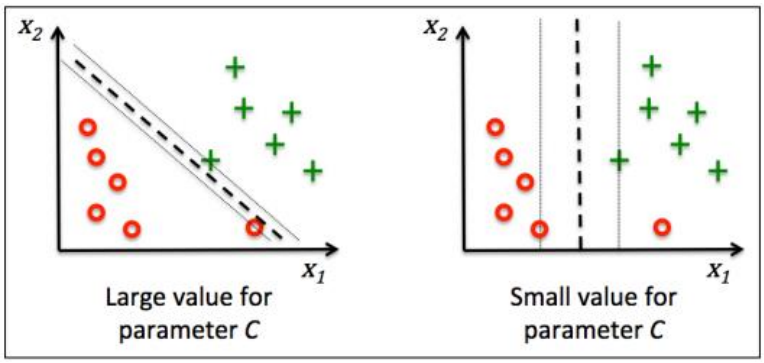
Abivektormasina (*support vector machine*) algoritmi kasutatakse samuti objektide klassifitseerimiseks. Algoritmi tööpõhimõte seisneb objektide klasside määramises, leides objektidele ruumis parima klasse eraldava hüperplaani. Parima hüperplaani leidmiseks võib vaadelda hüperplaanide vahekaugust selle ja lähima testobjekti ehk abivektori vahel, kus suurema vahekaugusega hüperplaan on eelistatum. Vahekauguse maksimeerimisel leitakse mudel, mis üldistab paremini, ning väiksema vahekauguse väärtuse puhul võib tekkida oht ülesobitumiseks [10, lk 69].

Objektide, mida ei saa jagada lineaarse hüperplaaniga, klassifitseerimiseks on vaja kasutada kernelfunktsiooni, mille abil saab leida hüperplaani kõrgemates dimensioonides, kus klasside eristamine on võimalik lineaarselt. Joonis 2 näitab, kuidas kernelfunktsiooni kasutamine projekteerib andmehulga mingisse kõrgemasse dimensiooni, kus tehakse lineaarne eraldamine ning saadud tulemus teisendatakse sama kernelfunktsiooni abil tagasi madalamase dimensiooni. Tulemuseks saame klassidele mittelineaarse eraldaja [10, lk 75-76].



Joonis 2. Kernelfunktsiooni kasutamine SVM puhul.

Abivektormasina puhul on oluline parameeter C, mis mõjutab seda, kui palju trahvitakse valesti klassifitseeritud väärtusi. Suurem C väärtus annab suurema trahvi valesti klassifitseeritud väärtustele, väiksem C väärtus madalama. Sellest sõltub samuti, milline klasside eraldamise hüperplaan valitakse. Joonis 3 näitab kuidas sõltub hüperplaani valik C väärtusest [10, lk 72].



Joonis 3. Valitud hüperplaanid erinevate C väärtuste puhul.

2.6.4 Otsustuspuu

Otsustuspuu (*decision tree*) algoritmid on kasutusel nii klassifitseerimis- kui ka regressiooniprobleemidel. Otsustuspuude puhul leitakse objekti klass puu konstrueerimisega. Igal sammul valitakse muutuja, mis eraldab klasse kõige paremini ning jagatakse objektid muutuja väärtuste järgi. Selline ülalt-alla lähenemine annab puu, kus sisemised sõlmed on muutujad ning lehtsõlmed võimalikud klassid. Niimoodi leitakse mudel, mis klassifitseerib objektid lihtsate otsustusreeglite põhjal.

Otsustuspuude puhul tuleb silmas pidada, et puu ei veniks liiga sügavaks. Mida sügavam puu on, seda keerulisemaks muutub otsuse tegemise funktsioon ning suureneb võimalus ülesobitumiseks. Selle vältimiseks tuleks täpsustada kui sügavat puud soovitakse või mitu objekti ühte lehte minimaalselt jääb.

Otsustuspuude puhul valitakse selline jagamise kriteerium, mis annab parima informatsiooni kasvu (*information gain*) ja vähendab ebapuhtust. Jagamisel kasutatakse põhiliselt kahte erinevat kriteeriumit - Gini kriteerium ja entroopia . Entroopia kriteeriumi puhul tuleb taotleda lehtede puhtust - mida rohkem ühte klassi kuuluvaid näidiseid on ühes lehes, seda väiksem on entroopia. Parim jagamise kriteerium on selline, mis toob kõige suurema informatsiooni kasvu ehk mis jagab objektid kõige puhtamalt. Gini kriteerium töötab analoogselt - mida ühtlasemalt on klassid ühes lehes jagatud, seda kõrgem on Gini indeks. Siinkohal taotletakse samuti Gini indeksi vähenemist, klassifitseerides võimalikult palju sarnaseid näidiseid ühte klassi [10, lk 80-83].

2.6.5 Otsustusmetsad

Otsustusmetsa (*random forest*) algoritmid kuuluvad ansambelõppe algoritmide hulka. Otsustusmetsa tööpõhimõte seisneb mitmete otsustuspuude genereerimisel ning nende kõige rohkem esineva tulemuse väljundiks andmisel. Klassifitseerimisel jagatakse

juhuslik alamhulk objekte sellise muutuja järgi osadeks, mis on selle konkreetse alamhulga puhul parim eristaja. Hiljem leitakse metsa puude keskmine ning väljastatakse klass. Selline juhuslikkuse ja keskmise leidmise kasutamine tagab mudeli, mis on vähem tundlikum ülesobitumisele.

Otsustusmetsa algoritmide puhul võetakse kogu andmehulgast juhusliku suurusega alamhulk, mille põhjal koostatakse otsustuspuu. Otsustuspuu puhul valitakse iga hargnemise puhul juhuslik hulk tunnuseid ning neist kasutatava jagamise kriteeriumi kohaselt parim. Sellist otsustuspuude genereerimist korratakse nii mitu korda kui soovitakse. Mida rohkem puid genereeritakse, seda paremini mets klassifitseerib, kuid seda rohkem on vaja ka arvutusvõimsust.

Otsustusmetsa algoritmid annavad tulemuseks genereeritud otsustuspuude enamiku poolt pakutud tulemuse enamushääletuse meetodil [10, lk 90-91].

2.6.6 Mitmekihiline pertseptron

Mitmekihiline pertseptron (*multilayer perceptron*) on tehisnärvivõrkude alla kuuluv mudel, mis kasutab edasilevi (*feedforward*) arhitektuuri. Mitmekihiline pertseptron koosneb sisendkihist, ühest või mitmest peidetud kihist ning väljundkihist, milledes ühe kihi kõik neuronid on omavahel ühendatud kõikide kõrvalkihtide kõikide neuronitega. Mitmekihiline pertseptron võimaldab eraldada klassidesse ka lineaarselt eraldamatuid objekte.

Mitmekihilise pertseptroni tööpõhimõte seisneb iteratiivsel õppel. Igal iteratsioonil teeb pertseptron tagasilevi meetodil kõigepealt ennustuse, seejärel mõõdab tekkinud vea ja liigub pöördsuunas tagasi sisendikihini. Iga kihi juures arvutab meetod, kui palju iga neuron panustas kogu vea suurusesse. Sisendkihi juures toimub gradientlaskumise arvutus ja kaalude korrigeerimine [4, lk 323].

Mitmekihiline pertseptron õpib tagasilevi (*backpropagation*) meetodil, mis arvutab ümber iga kihi kaalud vastavalt sellele, kui suur on viga väljundi ja tõeliste väärtuste vahel. Tagasilevi eesmärgiks on leida kaalude kombinatsioon, mis vähendab väljundis tekkivat viga kõige rohkem. Selleks kasutatakse gradientlaskumise meetodit, mis minimeerib viga leides osatuletiste gradiendid ja liikudes gradiendi negatiivses suunas.

Mitmekihiline pertseptron õpib funktsiooni, mis kõige paremini teisendab pertseptroni sisendi väljunditeks, millel on väikseim võimalik viga [8].

2.6.7 Sügav närvivõrk

Sügav närvivõrk kuulub tehisnärvivõrkude kategooriasse. Tehisnärvivõrk on arvutuslik arhitektuur, mis põhineb bioloogilise aju arhitektuuril. Närvivõrk koosneb kihtidest, mis on omavahel ühendatud ning koosnevad omakorda neuronitest. Kõige tüüpilisem närvivõrgu arhitektuur koosneb sisendkihist, peidetud kihtidest ning väljundkihist. Sügava närvivõrgu saame, kui peidetud kihte on rohkem kui üks. Närvivõrgu peidetud kihtides olevad neuronid teisendavad neisse tuleva sisendi koos kaaludega väljundiks, kasutades selleks aktiveerimisfunktsiooni. Aktiveerimisfunktsiooni ülesanne on anda konkreetsete sisendite ja kaalude puhul väljund.

Närvivõrkude õppimisel üritab närvivõrk leida mingi ülesande jaoks optimaalse lahenduse. Selleks on vajalik defineerida maksumusfunktsioon, mis näitab kui kaugel on närvivõrk mingi optimaalse lahenduse leidmisest. Klassifitseerimise puhul on tavaliseks maksumusfunktsiooniks keskmine ruutviga, mis näitab võrgu väljundi ja tegelike väärtuste keskmist viga. Kasutades maksumusfunktsiooniks keskmist ruutviga ning selle minimeerimiseks gradientlaskumist, saame närvivõrku õpetada klassifitseerimisprobleemide lahendamiseks [1].

Sügava närvivõrgu implementeerimiseks kasutatakse antud töös Google arendatud TensorFlow raamistikku.

3 Tudengite väljalangevuse ennustamine masinõppe meetodil

Antud töö eesmärgiks on ennustada tudengite väljalangevust ning leida üles faktorid, mis kõige enam mõjutavad tudengite enneaegset õppetöö lõppemist. Kasutatavate andmestike põhjal tehakse katsed erinevate parameetrite alusel valitud alamandmehulkadel. Katsete tulemusel võib teha üldistavaid järeldusi põhjuste kohta, mis mõjutavad väljalangemist kõige otsesemalt ning tuua välja nende tudengite hulga, kes on kõige tõenäolisemalt väljakukkumisohus.

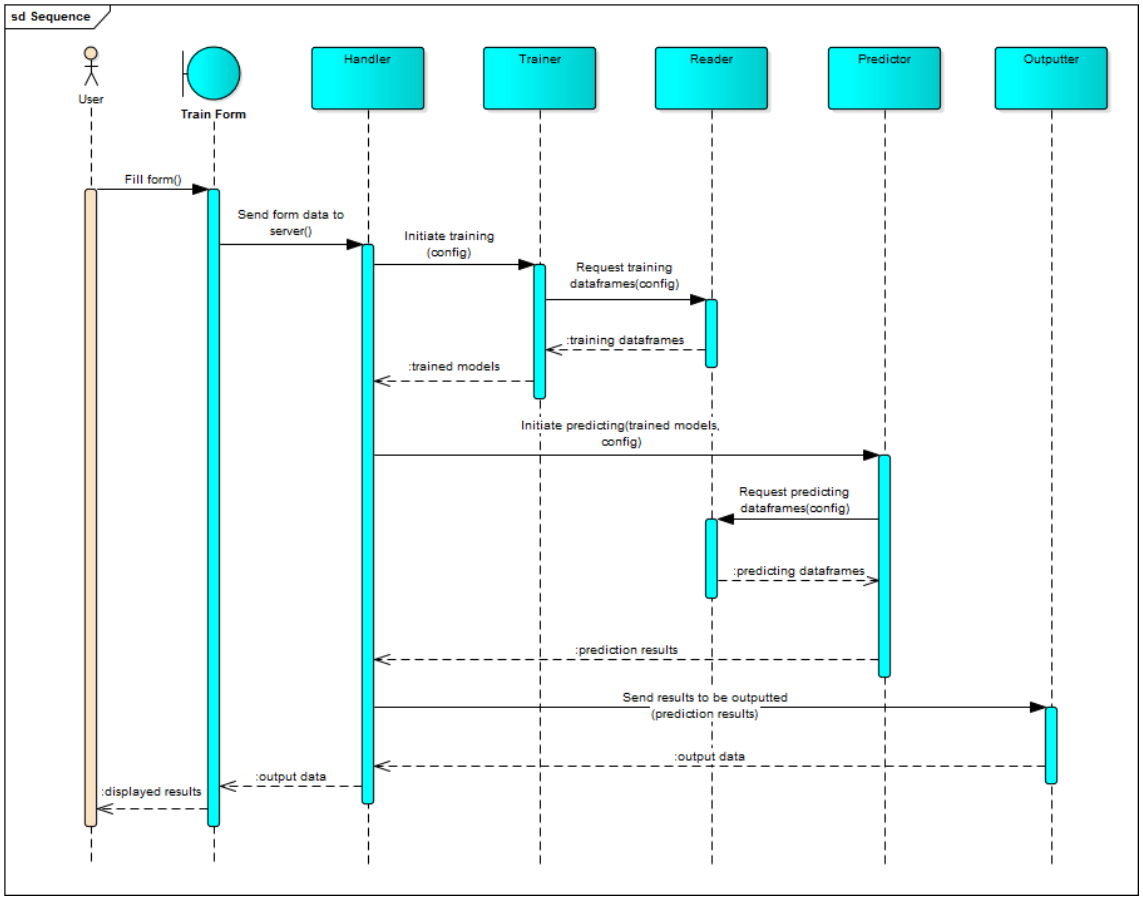
Masinõppe realiseerimiseks kasutatakse antud töös Pythonit ning teeke Scikit-learn ja TensorFlow. Valik langes Pythonile, sest just sellel keelel on pikalt arendatud ja põhjalikult dokumenteeritud masinõppe teek Scikit-learn. TensorFlow kasutamise poolt räägib tõsiasi, et Scikit-learn teegi võimalused sügavate närvivõrkude arendamiseks ei ole piisavalt paindlikud. Sellel põhjusel valis autor sügavate närvivõrkude realiseerimiseks TensorFlow teegi, sest tegemist on kõige uuema (avalikustatud 2015) avaliku lähtekoodiga teegiga, mis on arendatud just närvivõrkudele spetsialiseeritult.

3.1 Arhitektuur

Masinõppe osa realisatsioon on loodud Pythoni keeles. Masinõppe rakendus on jagatud klassideks, millest igaüks pakub ühe tegevuse (andmete lugemine, mudelite treenimine, ennustamine) jaoks loodud funktsioone. Näiteks on eraldi klass loodud andmeallikatest andmete lugemiseks ja masinõppe algoritmidele sobivateks *dataframe*'ideks muutmiseks. Selline tükeldus annab võimaluse uute andmeallikate tekkimisel kergelt rakendust laiendada neid kasutama. Mudelite treenimise osaga liidestuseks peab loodud klass realiseerima abstraktses klassis `Reader` kirjeldatud funktsioone.

Mudelite treenimiseks on loodud `Trainer` klass, mis sisaldab endas masinõppe algoritmide klassifitseerijate optimeerimist ning konkreetsete mudelite treenimist testandmehulgal. `Trainer` klassis on kirjeldatud iga algoritmi jaoks võrkotsingul kasutatav parameetrite võrk, mida on samuti kerge laiendada uute algoritmide kasutamiseks. Pärast mudelite treenimist liigub rakenduse töö edasi mudelitel ennustamise klassi `Predictor`.

Ennustamise funktsionaalsus on realiseeritud `Predictor` klassis, mis kasutab eelnevalt loodud mudeleid ning `Reader` klassist saadud ennustamise andmetel põhinevaid *dataframe*'e. Mudelid ja *dataframe*'id pannakse semestri kaupa paari ning neil viiakse läbi ennustamise protsess. Vastavalt kasutaja soovidele arvutatakse lisaks konkreetsete tudengi klassidele välja ka klassidesse kuulumise tõenäosused ning tunnuste tähtsused. Ennustamise tulemused saadetakse edasi `Outputter` klassi, milles tehakse vajalikud teisendused veebirakenduses kuvamiseks. Joonis 4 kujutab rakenduse tööd ennustamise kasutusjuhu puhul vastavalt arhitektuurile.



Joonis 4. Jadadiagramm ennustamise protsessi kohta.

`Outputter` klassis realiseeritud funktsionaalsus võimaldab erinevatel viisidel salvestada ennustamise tulemusi. Tulemusi võib kasutada lihtsalt tekstifailidena või luua nende põhjal JSON failid veebirakenduse jaoks. `Outputter` klassi saab laiendada, lisades funktsionaalsust erinevatesse formaatidesse salvestamiseks.

3.2 Andmestik

Töös kasutatakse kahte andmestikku, mis on erinevas formaadis. Kasutusel on CSV formaadis andmed ning JSON formaadis andmed, mis mõlemad sisaldavad tudengi kohta käivaid üldisikuandmeid ning tema soorituste andmeid. Kõik isikuandmed on esitatud anonüümselt, kuid konkreetse isiku andmed saab kokku viia unikaalse identifikaatori järgi.

CSV formaadis andmed sisaldavad eraldi failides tudengite sooritusi ning omaette failis ka tudengite üldandmeid. JSON formaadis andmed on pärit Õppeinfosüsteemi andmetel põhinevast Oracle andmebaasist ning koosnevad kümnest erinevast JSON failist iga andmebaasi tabeli kohta. Andmete kasutamiseks on vajalik eelnevalt andmed filtreerida püstitatud kriteeriumite järgi ning teisendada sobilikku formaati.

Üldandmete osas teeb autor valiku olulisemate isikutunnuste järgi, mis eristavad tudengeid kõige laiemalt, näiteks sugu, sünniaeg, kodakondsus, õppekeel, sisseastumisaasta, õppekava ja õppevorm. Nende andmete põhjal koostatakse eraldi mudel, kuhu soorituste andmeid ei lisata. Selline viis võimaldab leida üldisemaid seoseid lõpetamise tõenäosuse ja isiku üldandmete vahel.

Soorituste andmete kasutamisel ühendatakse isiku identifikaatori kaudu isiku ainete sooritused üldisikuandmetest valitud veergudega. Mõlemad kasutatud andmestikud sisaldavad üle 60 000 tudengi info, kuid need erinevad aastate poolest. CSV andmed on aastatest 1983 kuni 2013 ning JSON andmed aastatest 1990 kuni 2017. Andmestike puhul on vajalik leida aktuaalsem info tänasele, mistõttu piiratakse sisseastumisaastat CSV andmestiku viimasele viiele aastale ehk valitakse välja tudengid, kelle sisseastumisaasta on 2008 või hilisem. Samuti toimitakse ka JSON andmetega, et mõlema andmekogu peal tehtud katsed oleksid omavahel võrreldavad. Lisaks sellele saab sellise aastate piiranguga vaadelda ühe õppekavaversiooni piires tudengeid. Mõlema andmestiku puhul viiakse läbi teisendused nii, et tulemuseks saadud andmed oleks samas formaadis masinõppe mudelite treenimiseks ja nendel ennustamiseks.

Samuti piiratakse ainete soorituste infot õppekava kohustuslike ainetega. Kohustuslike ainete info pärineb Õppeinfosüsteemist ning iga proovitud õppekava kohta on välja toodud eraldi kohustuslike ainete loetelu. Selliselt tagatakse, et väljalangevust hinnatakse

vaid nende ainete tulemuste järgi, mis iga tudeng peab läbima õppekava lõpetamiseks ning et kõigi tudengite puhul on kasutusel samad tunnused.

Masinõppe mudelid luuakse kõige suurema tudengite arvuga õppekava jaoks antud andmestikus, milleks on informaatika. Selline piirang on vajalik, et saada võimalikult suur tudengite valim, mille alusel tehtud mudelid annaksid kõige kaalukamat infot. Samuti kasutab autor informaatika õppekava mudelite loomiseks, sest saab nii mudelite tööd kontrollida iseenda näitel.

Mudelite treenimiseks on vaja ka ennustatavat väärtust, mis antud töös on tudengi lõpetamise binaarväärtus. Lõpetamise klass 0 tähendab tudengi väljalangemist, klass 1 tähendab lõpetamist.

3.3 Andmete teisendused

Tudengite üldandmete ja soorituste info põhjal loob autor Pythoni Pandas teeki kasutades *dataframe'*id ehk andmete töötlemiseks loodud tabelilaadsed andmestruktuurid. Need võimaldavad teha SQL-i laadseid päringuid ja ühendamisoperatsioone ning filtreerida andmeid. Iga õppekava semestri jaoks luuakse oma *dataframe,* milles ühendatakse kokku tudengi sooritused kohutuslikes ainetes ning tema lõpetamise info. Lisaks ainult soorituste info põhjal ennustamisele, teeb autor katsed ka andmetega, milles on soorituste info ühendatud ka tudengi üldandmetega.

Katseteks jagatakse soorituste andmed erineval viisil. Esimese katse jaoks jagab autor kohutuslikud ained eraldi semestrite kaupa ning iga semestri kohta teeb eraldi mudeli. Teisel katsel võtab iga semestri korral konkreetse semestri info ning lisab ka kõigi eelnevate semestrite sooritused. Vastavalt nõutud semestrite arvule arvutatakse treenimise andmehulga jaoks sisseastumise aasta, mis peaks olema kõige hilisem aastakäik, mille puhul on olemas lõpetamise info ennustamise semestriks. Näiteks soovides ennustada 2011/2012 sisseastunud tudengite peal pärast teist semestrit, tuleb mudelid treenida 2009/2010 õppeaastal sisseastunud tudengite peal, kellel on 2012 aasta kevadeks läbitud 6 semestrit.

Semestrite soorituste infoga liidetakse tudengite üldinfost valitud veerud. Valitud veerud on JSON andmete puhul sugu, sünniaeg, sisseastumisaasta, õppekeel, õppekoormuse tähis ja õppevormi tähis. CSV formaadis andmete puhul kasutab autor lisaks eelnevatele

veel ka kodakondsust, eelmise õppetaseme tähist, keskmist hinnet, kogutud EAP-de arvu. Valik selliste veergude kasuks langes põhjusel, et need on tunnused, mis tudengeid kõige üldisemalt jagavad ning selliselt on nende kasutamisel võimalik leida kõige laiemad väärtuste vahemikud iga klassi jaoks. Teisendamise tulemusena saadud *dataframe*'ides on lisaks üldandmete veergudele iga aine jaoks üks veerg ning iga rida tähistab ühte tudengit. Veerus olev väärtus on aine puhul aines saadud hinne ning üldandmete veergude puhul tunnuse väärtus konkreetse tudengi puhul.

Õppimise andmehulga teisendamisel loetakse tudeng väljakukkunuks ehk määratakse "Lõpetanud" veergu väärtus "0", kui tudengi puhul on CSV formaadis andmehulgal määratud "Lõpetanud" veergu "0", ning määratakse väärtus "1", kui andmestikus esineb vastavas veerus väärtus "1". JSON formaadis andmestiku puhul on olemas tudengi kohta käivate dokumentide info, millest saab välja lugeda, kas tudengile on määratud lõpetamise dokument ja eksmatrikuleerimist tähistav dokument. Sellisel juhul määratakse väärtuseks "1", kui esineb lõpetamise dokument ja puudub eksmatrikuleerimise dokument. Esimese dokumendi puudumisel ja teise dokumendi esinemisel määratakse väärtuseks "0".

Saadud *dataframe*'id antakse edasi Python masinõppe teegile Scikit-learn, mis võimaldab luua erinevaid algoritme kasutades mudeleid. Samuti tehakse katsed läbi ka Google'i loodud tehisnärvivõrkude teegi TensorFlow'ga.

3.4 Mudelite treenimine

Mudelite treenimisel tehakse katsed seitsme erineva klassifitseerimisalgoritmiga, milleks on

- logistiline regressioon
- k-lähima naabri klassifitseerimine
- abivektormasina klassifitseerija
- otsustuspuu
- otsustusmets
- mitmekihiline pertseptron
- sügav närvivõrk

Selline algoritmide valik tuleneb sellest, millised algoritmid sobivad antud probleemi lahendamiseks paremini. Tegu on klassifitseerimise ülesandega, seega valib autor algoritme klassifitseerimisalgoritmide seast. Jäetakse välja kõik regressioonimudelid, sest ennustatav väärtus on antud juhul klass, mitte pidev suurus. Samuti ei tehta katseid jälgimiseta õppe algoritmidega, sest antud probleemi lahendamisel on ennustatavad väärtused olemas. Olulisteks kriteeriumiteks on ka mudelite treenimise kiirus ning piisav täpsuse skoor kaheklassilise klassifitseerimise jaoks.

Enne mudelite treenimist tehakse katsed kasutades tunnuste valimist, et vähendada müra, mille õppimisel muutub mudel tundlikumaks ülesobitumisele. Katsetel, kus semestrite ained on *dataframe*'ides kumulatiivselt, on näiteks neljanda semestri puhul juba üle 15ne tunnuse. Samas tunnuste tähtsuste arvutamisel osutusid ainult 3-4 tunnust mudelis määravaks ning ülejäänud tunnuseid mudel ei kasutanud. Mõistlik on siinkohal valida välja 3-4 neist, et mitte treenida mudelit müra ja ebaoluliste tunnuste pealt.

Mudelite treenimisel kasutatakse *stratified k-fold cross validation'*it, mis tagab väiksema tundlikkuse ülesobitumisele. Erinevalt tavalisest *k-fold cross validation'*ist, jagab *stratified* alamhulgad nii, et erinevate klasside proportsioonid on erinevate alamhulkade vahel võrdsed. Koos *cross validation'*iga kasutab autor hüperparameetrite täpsustamiseks võrkotsingut, mis leiab parima parameetrite kombinatsiooni iga kalkuleerija (*estimator*) jaoks. Selliselt leitakse parim võimalik kalkuleerija iga *dataframe*'i jaoks. Võrkotsingu parameetrid iga algoritmi kohta on ära toodud lisas 1. Pärast võrkotsingut eraldatakse ülejäänud veergudest ennustatav tulemus *y,* milleks on veerg „Lõpetanud".

Iga kalkuleerija sobitatakse andmetega ning viiakse läbi ennustus iga tudengi kohta. Ennustamise tulemuse täpsust hinnatakse erinevaid meetrikaid kasutades. Iga mudeli täpsust hindab autor f1-skoori, täpsust, saagist ja õigsust kasutades. Nii on kõigi katsete mudelite sooritused omavahel võrreldavad.

Seda protsessi korratakse iga loodud *dataframe*'i peal ning tulemused kogutakse kokku täpsuste hindamiseks.

3.4.1 Närvivõrgud TensorFlow raamistikuga

Lisaks Scikit-learn teegi kasutamisele, teeb autor katsed läbi ka närvivõrkude peal. Selleks kasutab ta TensorFlow raamistikku, mis võimaldab lihtsalt teha ennustusi

kasutades närvivõrke. TensorFlow raamistikus on implementeeritud juba ka närvivõrkude klassifitseerimise mudelid, mis lihtsustavad närvivõrkude ehitamist. Katsete käigus proovitakse luua mudelid nii sisseehitatud klassifitseerijaga kui ka enda loodud implementatsiooniga sügavast närvivõrgust.

Närvivõrgu ise defineerimise eeliseks on suurem kontroll võrgu parameetrite üle. Nii saab defineerida igale kihile eraldi aktiveerimisfunktsioonid, neuronite arvud, arhitektuuri. Suurem kontroll närvivõrgu parameetrite üle võimaldab luua mudeleid, mille täpsus ning tundlikkus ülesobitumisele on piisavalt head konkreetse probleemi jaoks.

Autor lähtub parameetrite optimeerimisel soovitustest, mis pärinevad A. Géroni raamatust „Hands-On Machine Learning with Scikit-Learn“. Nimelt soovitatakse selles kasutada närvivõrgu peidetud kihtide arvuna 1-2, mis on vähe kompleksete probleemide korral piisav. Katsed tehakse läbi erineva arvu peidetud kihtidega vahemikus 1-10. Samuti soovitab teose autor neuronite arvuks igal peidetud kihil 150 või rohkem ning aktiveerimisfunktsiooniks ReLU funktsiooni [4, lk 334-337]. Antud töö autor kasutab igal kihil 150-t neuronit ning aktiveerimisfunktsiooni vastavalt teoses soovitatule.

Ristentroopia arvutatakse `weighted_cross_entropy_with_logits` meetodiga, mis arvestab ka klasside ebatasakaaluga. `GradientDescentOptimizer`, mis kasutab kadufunktsiooni optimeerimiseks gradientlaskumist, on kasutusel optimeerimise sammus [13].

Selliselt ülesehitatud närvivõrk tuleb toime väiksest andmestikust põhjustatud ülesobitumisega ning annab head tulemused nii täpsuse kui saagise osas.

3.5 Mudelite hindamine

Mudelite täpsust hindab autor erinevaid meetrikaid kasutades. Valik on tehtud selline, et hinnata mudeli käitumist just antud probleemile kohaselt.

Tudengite väljalangevuse hindamisel on oluline leida kui suur on väärpositiivsete ennustuste hulk. Mida rohkem on ennustamisel väärpositiivseid tulemusi, seda suurem on oht, et potentsiaalsed väljakukkujad jäävad identifitseerimata. Väärpositiivsete tulemuste arvu hindamiseks on kasulik täpsus, mis näitab kui suure osa moodustavad

väärpositiivsed kõikidest positiivsetest tulemustest. Scikit-learn teegis implementeerib täpsust `precision_score` meetod.

Samuti hinnatakse mudelite saagist `recall_score` meetodi abil. Saagis annab mudeli töö kohta teada, kui hästi saab mudel hakkama kõigi positiivsete väärtuste leidmisega. Selliselt saab hinnata kui täielikud on mudeli ennustused.

Mudelite hindamisel kasutab autor ka f1-skoori. f1-skoori võib võtta kui täpsuse ja saagise kaalutud keskmist, mille puhul 1 on parim tulemus ja 0 halvim. Kasutatakse `f1_score` meetodit Scikit-learn teegist.

Lisaks neile meetoditele, on mõistlik leida ka mitu ennustatud väärtust langeb kokku tegelike väärtustega. Selle kontrollimiseks kasutab autor `accuracy_score` meetodit, mis arvutab mitu protsenti ennustatud väärtustest olid samased tegelike väärtustega.

Antud andmestikuga töötades on probleemiks klasside ebatasakaal, mis väljendub selles, et tudengeid, kellele on andmestikus märgitud lõpetamise väärtuseks „1“, on oluliselt vähem kui neid, kellele on märgitud väärtuseks „0“. Keskmiselt aastatel 2010-2013 on lõpetajaid umbes 20% kogu tudengite hulgast, mis teeb klasside suhteks 4:1.

Ebatasakaalu arvestamiseks mudelite hindamisel on vajalik kasutada täpsuse, saagise ja f1-skoori hindamiseks meetoditel *average* parameetrit, millega saab määrata millisel viisil tehakse ennustatud tulemuste puhul keskmistamist. Autor kasutab parameetri väärtuseks ’*weighted*’, mis arvutab iga klassi puhul meetrika väärtuse ning arvutab seejärel nende kaalutud keskmise. Sellisel viisil on kaaludeks iga klassi puhul tõeliste väärtuste arv, mis omakorda tagab korrektsema tulemuse klasside ebatasakaalu all kannatavate andmestike puhul.

Erinevate mudelite võrdlemisel pannakse kõige suurem kaal täpsuse skoorile, mis on antud probleemi puhul kõige olulisemaks meetrikaks.

Täpsusi hinnatakse eraldi mudelitel, mis on loodud 2011/2012 õppeaastal sisseastunud tudengite andmehulgal ennustamiseks. Mõlema andmehulga puhul on kõigi meetrikate puhul tulemused sarnased ning tabelis on kuvatud nende keskmised väärtused. Tabel 1 toob välja skoorid esimese semestri soorituste ja üldandmete põhjal ning nelja esimese semestri soorituste ja üldandmete põhjal. Tabelis on nelja semestri puhul lahtrites toodud

tulemus iga semestri kohta kronoloogilises järjekorras. Samuti toob Tabel 2 välja skoorid ainult üldandmete põhjal loodud mudelitel ennustamise tulemuste kohta. Tabelite tulemusi on analüüsitud peatükis 5.1.

Tabel 1. Erinevate algoritmide skoorid soorituste ja üldandmetega ennustades.

		Logistic Regression	KNNClassifier	SVC	DecisionTree Classifier	RandomForest Classifier	MLPClassifier	DNN
1 semester eraldi	**GridSearch parim õigsus**	0.79	0.86	0.83	0.9	0.91	0.9	
	Õigsus	0.7	0.71	0.73	0.65	0.63	0.83	0.52
	Täpsus	0.86	0.86	0.88	0.88	0.86	0.91	0.83
	Saagis	0.7	0.71	0.73	0.65	0.63	0.83	0.52
	F1-skoor	0.74	0.75	0.77	0.66	0.68	0.85	0.58
4 semestrit eraldi	**GridSearch parim õigsus**	0.86 / 0.87 / 0.92 / 0.93	0.80 / 0.87 / 0.92 / 0.93	0.87 / 0.87 / 0.94 / 0.95	0.94 / 0.98 / 0.97 / 0.96	0.94 / 0.93 / 0.97 / 0.97	0.84 / 0.90 / 0.90 / 0.95	
	Õigsus	0.75 / 0.70 / 0.68 / 0.74	0.80 / 0.80 / 0.76 / 0.66	0.78 / 0.76 / 0.72 / 0.74	0.85 / 0.88 / 0.88 / 0.88	0.81 / 0.89 / 0.87 / 0.85	0.85 / 0.76 / 0.79 / 0.76	0.48 / 0.58 / 0.57 / 0.63
	Täpsus	0.85 / 0.84 / 0.82 / 0.88	0.86 / 0.84 / 0.88 / 0.84	0.88 / 0.87 / 0.86 / 0.88	0.88 / 0.93 / 0.93 / 0.92	0.87 / 0.91 / 0.89 / 0.90	0.86 / 0.83 / 0.88 / 0.88	0.77 / 0.67 / 0.72 / 0.75
	Saagis	0.75 / 0.70 / 0.68 / 0.74	0.80 / 0.80 / 0.76 / 0.66	0.78 / 0.76 / 0.72 / 0.74	0.85 / 0.88 / 0.88 / 0.88	0.81 / 0.89 / 0.87 / 0.85	0.85 / 0.76 / 0.79 / 0.76	0.48 / 0.58 / 0.57 / 0.63
	F1-skoor	0.78 / 0.74 / 0.72 / 0.77	0.82 / 0.81 / 0.79 / 0.69	0.81 / 0.78 / 0.75 / 0.77	0.86 / 0.89 / 0.89 / 0.89	0.83 / 0.90 / 0.87 / 0.86	0.85 / 0.78 / 0.81 / 0.78	0.55 / 0.62 / 0.61 / 0.67

Tabel 2. Erinevate algoritmide skoorid ainult üldandmetega ennustades.

	Logistic Regression	**KNNClassifier**	**SVC**	**DecisionTree Classifier**	**RandomForest Classifier**	**MLPClassifier**	**DNN**
GridSearch parim õigsus	0.78	0.88	0.84	0.92	0.94	0.90	
Õigsus	0.7	0.78	0.74	0.75	0.81	0.87	0.60
Täpsus	0.81	0.85	0.85	0.72	0.80	0.89	0.64
Saagis	0.7	0.78	0.74	0.75	0.81	0.87	0.60
F1-skoor	0.72	0.79	0.76	0.73	0.79	0.87	0.61

3.6 Väljalangevuse põhjuste leidmine

Antud töö eesmärgiks on leida potentsiaalsed väljakukkujad ning analüüsida andmestikust tulenevaid tõenäolisemaid väljalangemise ohu tekitajaid. Üheks viisiks, mille järgi leida põhjuseid tudengite väljalangevuses on analüüsida tunnuste panustamist kasutatud masinõppe mudelitesse. Selleks on võimalik kasutada Scikit-learn teegi pakutavat tunnuste tähtsuste meetodit, mis on implementeeritud otsustuspuude ja otsustusmetsade algoritmide puhul. Selliselt saab teada, milliseid kohustuslikud õppeained ning isikutunnused kõige rohkem mõjutavad tudengi lõpetamist ja mittelõpetamist.

Kasutatud algoritmidest implementeerivad `feature_importances` meetodit `DecisionTreeClassifier` ja `RandomForestClassifier`. Kasutades ennustamiseks neid algoritme, saab leida ka ainete Gini skoorid. Gini puhtuse skoori kasutavad otsustuspuudel põhinevad algoritmid igal sammul hindamaks seda, kui puhtalt mingi konkreetne tunnus objekte jagab. Kõrgem Gini skoor on indikaatoriks sellest, et antud tunnus klassifitseerib kõige suurema puhtusega [2].

Mõjukamate tunnuste pealt põhjuste leidmiseks kasutab autor mitmeid visualiseerimise meetodeid nii arvutatud tunnuste tähtsuste kuvamiseks kui ka näiteks otsustuspuude visualiseerimiseks, et leida konkreetsed tunnuste väärtused, mille järgi otsused tehti.

Nii saab leida üles tunnused, mis kõige rohkem mõjutavad seda, kas tudeng on väljalangemise ohus või mitte.

3.7 Väljalangevuse ennustamine

Treenitud ja hinnatud mudelite peal sooritab autor ennustamise katsed. Katsed viiakse läbi erinevate andmehulkade ja algoritmidega. Testandmestikuga on läbi viidud samasugused teisendused, mis treenimise andmehulgaga, et treenitud mudelid saaksid andmehulga vastu võtta ja ennustada.

Ennustamise tulemusena saadakse binaarne väärtus tulpa „Lõpetanud“, mille puhul 0 tähistab tudengi väljalangemist ning 1 tudengi lõpetamist. Lisaks binaarväärtusele saab

osade algoritmide puhul leida ka tõenäosused, millega tudeng ühte või teise klassi kuulub. Algoritmid, mis klassidesse kuulumise tõenäosusi leida võimaldavad, on logistiline regressioon, k-lähima naabri klassifitseerimine ning mitmekihiline pertseptron. Kasutades neid algoritme, saab tulemuseks anda ka mõlemasse klassi kuulumise tõenäosuse.

Samuti on võimalik, kasutades tunnuste tähtsuste leidmist implementeerivaid algoritme, leida mudelis leiduvate tunnuste Gini puhtuse skoore. Otsustuspuudel põhinevad algoritmid võimaldavad leida, mil määral mõjutab iga konkreetne tunnus objekti kuulumist teatud klassi. Nii on ennustamise tulemuseks võimalik anda ka õppeainete ja isikuandmete tähtsuse skoorid.

Kõik tulemused, mis mudelitel ennustamine annab, kuvatakse välja veebirakenduses kasutajale lihtsalt mõistetaval teel. Samuti kuvatakse rakenduses kasutatud andmestiku põhjal leitud tähtsamaid seoseid ja tunnuseid visualiseeritult graafikute näol.

4 Tulemuste visualiseerimine veebirakenduses

Ennustamise protsessi käivitamiseks ja tulemuste visualiseerimiseks loob autor veebipõhise rakenduse. Rakenduses on võimalik erinevate andmehulkadega viia läbi ennustamist, treenida mudeleid ning näha ennustatud tulemusi tabelite ja graafikute näol.

4.1 Nõuded rakendusele

Veebirakendusele seab autor mitmed eesmärgid, mida töö käigus täidab. Funktsionaalseteks nõueteks on:

- Veebirakendus võimaldab masinõppe protsessi käivitada ja tulemusi vaadata
- Rakendus võimaldab kasutajal üles leida väljalangemisohus tudengid
- Rakendus pakub võimalust analüüsida tudengi andmestikust tulenevaid tunnuste väärtusi ning nende seost ennustatud tulemusega
- Veebirakendus kuvab ülevaatlikku infot, mille alusel võib teha üldisemaid järeldusi andmestiku ja tulemuste kohta

Mittefunktsionaalseteks nõueteks on:

- Rakendust on tulevikus kerge laiendada ning sellele uusi võimalusi lisada
- Rakendust on mugav kasutada ning arvestatud on juurdepääsetavuse printsiipidega

Veebirakenduse esmaseks eesmärgiks on olla ennustamise protsessi liideseks, mille kaudu on võimalik ennustamist käivitada ning selle tulemusi näha. Selle eesmärgi täitmiseks on veebirakendus loodud selliselt, et kasutaja saab sisestada nõutud info vormidesse ning selle info põhjal pannakse serveris käima masinõppe osa. Samuti saab rakenduse kaudu välja kuvada ennustamise tulemusi mitmel erineval kujul.

Teiseks eesmärgiks on võimaldada kasutajal leida üles tudengid, keda tõenäoliselt ohustab väljalangemine. Selleks on loodud masinõppe tulemuste visualiseerimiseks tabelid konkreetsete tudengite infoga ning nende kohta käivate ennustustega. Samuti peab rakendusest olema näha tudengile ennustatud klasside esinemise tõenäosused ning tunnuste tähtsused, mis on põhjustanud sellise otsuse masinõppe mudelites.

Eelnevaga on seotud kolmas eesmärk - saada iga tudengi kohta konkreetset infot tema õppetulemuste ja väljakukkumise tõenäosuse seoste kohta. Selle eesmärgi täitmiseks on rakenduses võimalik näha nii tudengi kohta käivat väljalangemise tõenäosust ning selle tõenäosuse sõltumist kõige määravamast tunnusest. Info on kuvatud hajuvusdiagrammil, kus on ära toodud iga tudengi kohta käiv positiivse klassi esinemise tõenäosus ning väärtus kõige määravama tunnuse puhul. Selliselt on võimalik saada ka seletust, miks on masinõppe meetodil ennustatud üks või teine klass.

Lisaks eelnevale on kolmanda eesmärgi täitmiseks võimalik vaadata ka iga tudengi kohta käivat infot, mis pärineb ennustamise andmestikust. Nii on võimalik kasutajal näha iga tudengi konkreetseid soorituste tulemusi ning üldandmeid ja analüüsida neid vastavalt tunnuste tähtsustele ning tudengile ennustatud klassile.

Neljandaks eesmärgiks rakenduse puhul on pakkuda kasutajale ülevaadet ennustamise protsessi tulemuste kohta graafikute ja tabelite kasutamise kaudu. Selleks loob autor graafikud ülevaatlikuma info saamiseks. Visualiseerimisel kasutatakse erinevat tüüpi graafikuid, mis konkreetse kuvatava infoga paremini sobivad. Üldiste graafikute loomisel pakub autor kasutajale võimaluse teha üldisemaid järeldusi tunnuste kohta, mis põhjustavad suurema tõenäosusega väljalangemise ohu teket ning võimalust uurida, millised on nende tunnuste puhul konkreetsed väärtused, mis põhjustavad ennustatava klassi sellise väärtuse.

Viiendaks eesmärgiks on luua rakendus selliselt, et see oleks hõlpsasti laiendatav ja edasiarendatav. Selle eesmärgi täitmiseks on loodud masinõppe osas abstraktne klass `Reader` objektide jaoks, mis võimaldab tulevikus toetada erinevaid andmeallikaid, näiteks jooksvat info semestri kohta Moodle keskkonnast. Samuti on võimalik lisada erinevaid masinõppe algoritme mudelite treenimiseks loodud klassi. Ka veebirakenduses olevaid erinevaid visualiseerimise võimalusi on võimalik lisada JavaScripti teel.

Eraldi eesmärgiks on rakenduse arendamine kasutajamugavuse ning juurdepääsetavuse printsiipe järgides, mille kasutamine tagab mugavama kasutatavuse kõigi võimalike kasutajate jaoks.

4.2 Veebirakendus

Veebirakenduse loomisel kasutab autor Flask raamistikku, mis põhineb nii nagu masinõppe osagi Pythonil. Valik Flaski kasuks langes põhjusel, et rakenduse *back-end* ühilduks kergelt masinõppe osaga, ning põhjusel, et antud rakendus ei ole väga mahukas, mille korral oleks sobivamaks valikuks näiteks Django. Lisaks Flaski raamistikule kasutatakse ka mitmeid teeke, mis teevad rakenduse valmimise kiiremaks. Nendeks on

- Flask-Login[1] - sessioonihalduseks ja kasutajate haldamiseks
- Flask-SQLAlchemy[2] - andmebaasisuhtluseks ja objekt-relatsioonilise mudeli realiseerimiseks
- Flask-Migrate[3] - andmebaasimigratsioonide kasutamiseks
- Flask-WTForms[4] - veebivormide loomiseks ja valideerimiseks
- Flask-Bcrypt[5] - *bcrypt* räsifunktsiooni kasutamiseks paroolide turvaliseks haldamiseks

Veebirakendus luuakse MVC mustrit kasutades. Nii on eraldi failides ja osades kasutajale näidatavad vaated, andmebaasimudelid ning neid kahte ühendav kontroller.

Veebirakenduse väljanägemise loomisel kasutab autor *dashboard* tüüpi, mis antud rakenduse andmekeskse funktsionaalsusega kõige paremini sobib. *Dashboard* sobib hästi andmete visualiseerimiseks ning kiiresti olulistest andmetest ülevaate saamiseks. *Front-end* arendamisel kasutatakse HTMLi, CSSi, JavaScripti, samuti Bootstrap[6] teeki, et muuta rakendus kergelt kasutatavaks kõigil seadmetel.

Rakenduses saab teha valiku kas treenida ja ennustada koos soorituste infoga ja üldandmete infoga või ainult üldandmeid kasutades. Samuti saab valida, kas ennustada soovitakse ise treenitud mudelitel või eelnevalt õpitud mudelitel.

---

[1] https://flask-login.readthedocs.io/en/latest/

[2] http://flask-sqlalchemy.pocoo.org/2.1/

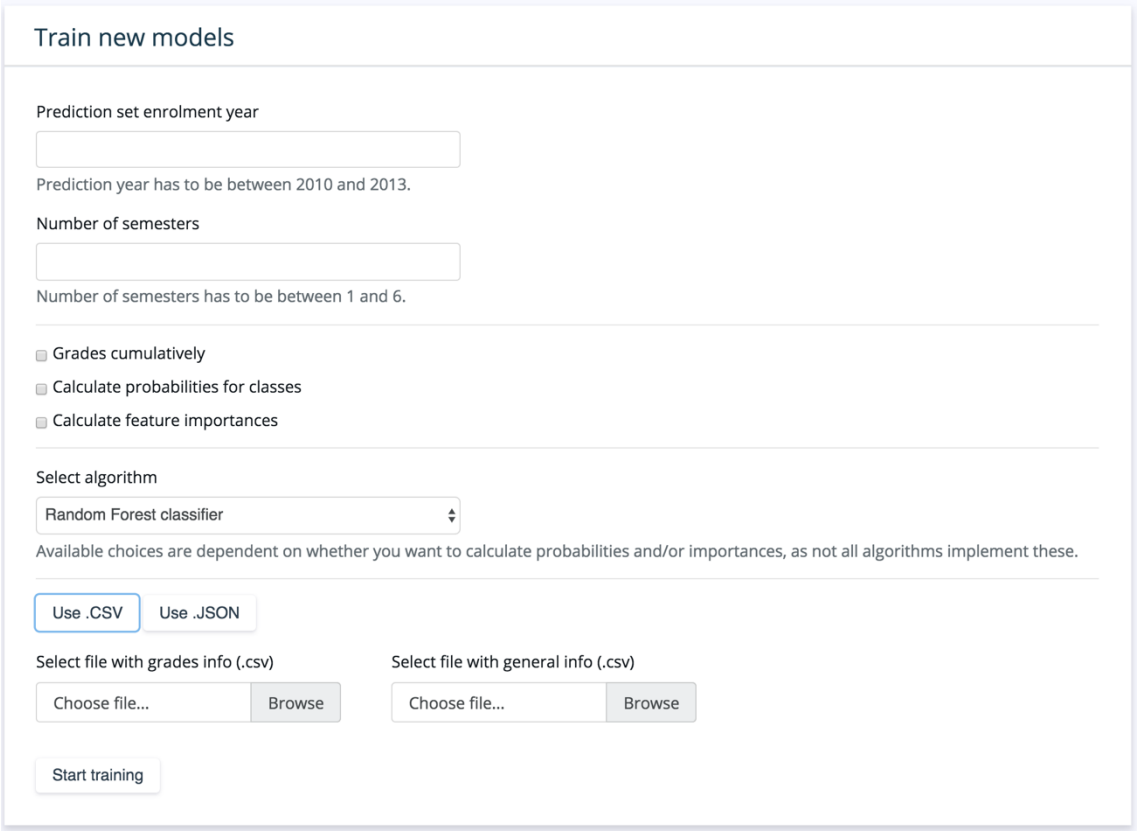
[3] https://flask-migrate.readthedocs.io/en/latest/

[4] https://flask-wtf.readthedocs.io/en/stable/

[5] https://flask-bcrypt.readthedocs.io/en/latest/

[6] https://v4-alpha.getbootstrap.com/

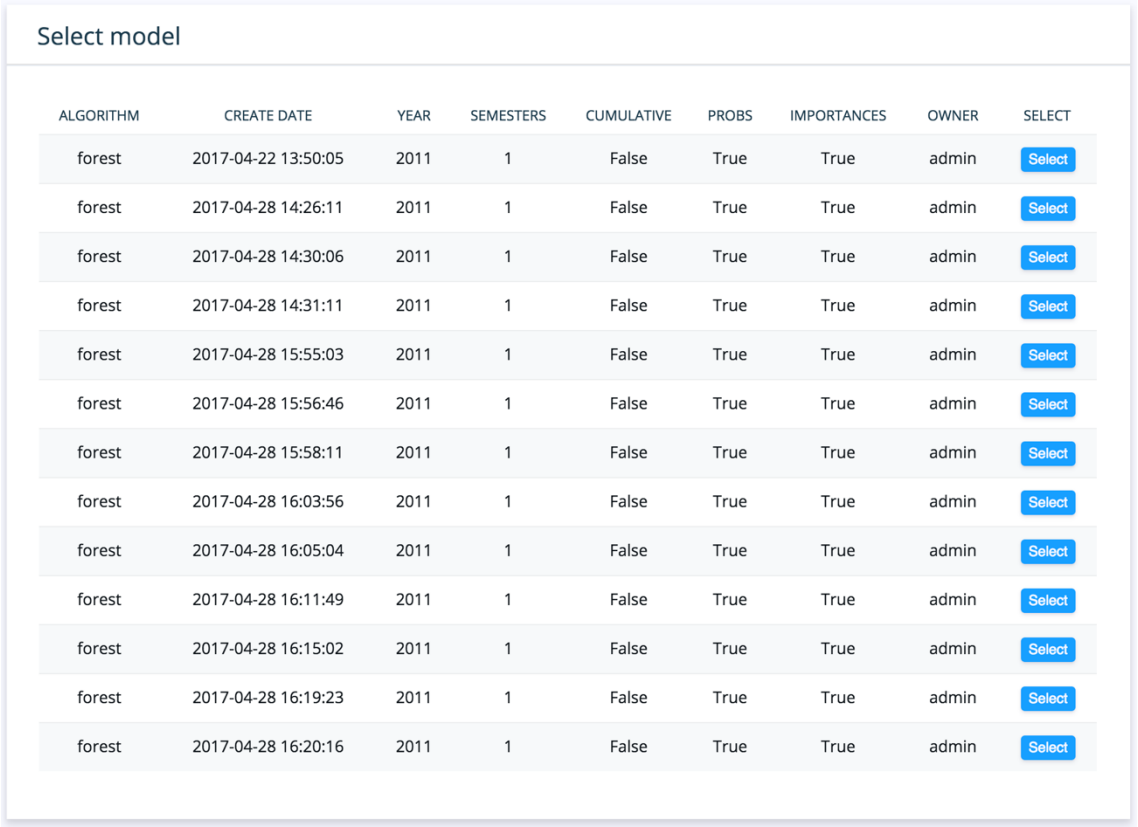
Soorituste info põhjal mudelite treenimiseks või ennustamiseks on vajalik täita ära vorm (Joonis 5), kuhu saab sisestada ennustamise andmestiku tudengite sisseastumisaasta, mitme semestri hinnete andmed andmestikus on, kas soovitakse semestrite sooritusi rakendada kumulatiivselt, kas soovitakse arvutada klasside tõenäosused, kas soovitakse arvutada tunnuste tähtsused ning sisestada CSV formaadi valimisel soorituse ja üldandmete CSV fail. JSON andmete puhul rakendus hetkel kasutajalt failide üleslaadimist ei nõua, vaid kasutab andmebaasist pärit JSON andmeid, mis on rakenduses salvestatud. Lisaks eelnevale on võimalus valida algoritm, millega mudel treenitakse.



Joonis 5. Mudelite treenimise vorm rakenduses.

Üldandmete põhjal mudelite treenimiseks või ennustamiseks on vajalik täita sama vorm mis eelnevalt, kuid sealt puudub soorituste faili lisamine ning semestrite numbri ja kumulatiivsuse sisestamine.

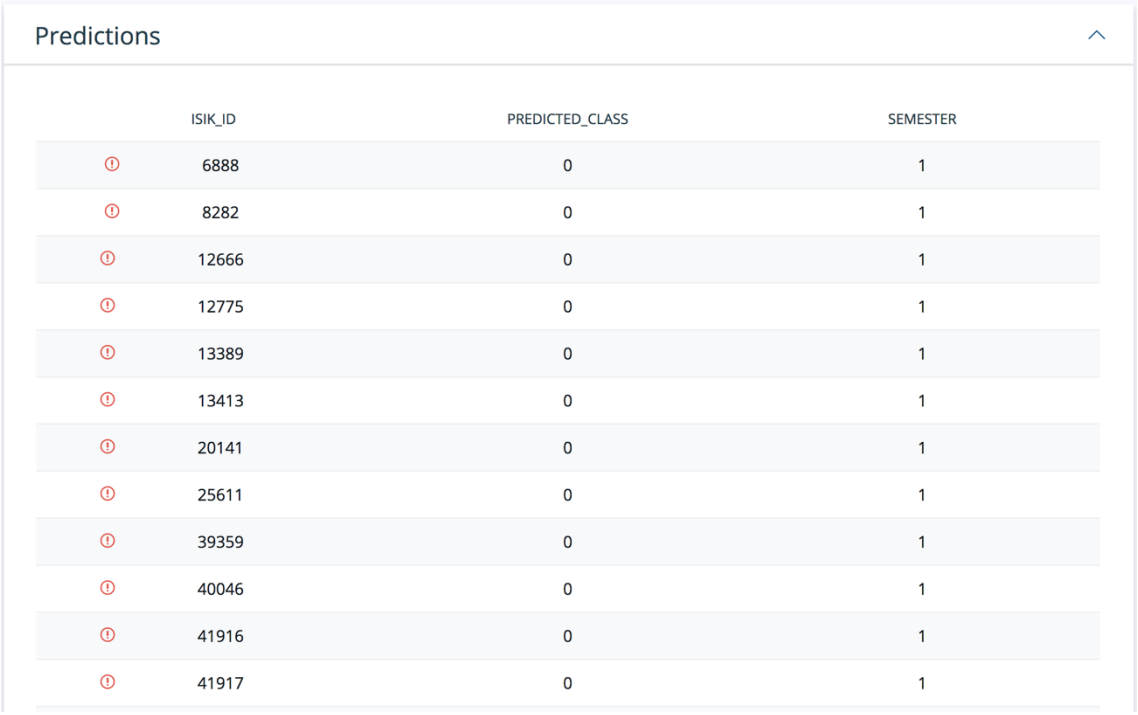
Soovides ennustada eelnevalt õpitud mudelitel, tuleb pärast parameetrite sisestust valida sobivaim mudel olemasolevaist ning siis saab toimuda ennustamine valitud mudeliga (Joonis 6).



Joonis 6. Olemasolevatest mudelitest sobivaima valimine rakenduses.

Ennustamise tulemuste valmimisel kuvatakse kasutajale ennustused iga andmestikus olnud tudengi kohta (Joonis 7). Samuti kuvatakse tunnuste tähtsused ja klasside tõenäosused, kui neid on kasutaja soovinud arvutada.

Tabelites on kuvatud iga soovitud semestri kohta tudengi kohta käiv ennustus ning vajadusel ka erinevate klasside tõenäosused. Nii on ühe tudengi kohta ühes tabelis mitu rida. Ridade arv ühe tudengi kohta on üldjuhul võrdne soovitud semestrite arvuga, kuid võib olla ka väiksem, kui antud tudeng mõnel soovitud semestril ei ole aineid sooritanud. Sellisel viisil on iga semestri puhul välja toodud ennustus selle semestri kohta, mis võib erineda teiste semestrite ennustustest. Nii on võimalik vaadata, kuidas tudengi sooritused erinevatel semestritel mõjutavad tema väljalangemise riskigruppi kuulumist. Soorituste ja üldandmete info leiab iga tudengi kohta talle kuuluva tabelirea peal klikkides, mille peale avaneb modaalaken tabeliga. Teades tudengi soorituste infot ning semestrite ennustust, on võimalik analüüsida, kuidas tehtud sooritused mõjutavad väljalangemise ohtu.



Joonis 7. Ennustuse tulemuste kuvamine rakenduses.

Lõputöö raames realiseeritud versioon rakendusest ei ole täielik ning seda on võimalik tulevikus lihtsalt laiendada. Rakenduse kitsaskohaks on praegu valikute puudumine andmestiku formaadi osas, nimelt rakendus ootab kasutajalt kindla struktuuriga faile, mille põhjal koostatakse *dataframe*'id. Täpsem info rakenduses nõutud formaatide ja kasutamise kohta on ära toodud antud lõputöö Githubi lehel[1].

4.3 Visualiseerimine

Lisaks ennustamise tulemustele on rakenduses võimalik vaadata erinevate andmete visualiseeritud kujutusi. Visualiseerimisel kasutatud graafikud ja tabelid on loodud kasutades JavaScripti teeki Chart.js[2]. Töö käigus proovis autor kasutada ka JavaScripti teeki D3.js[3], kuid ajakulu, mis teegi kasutama õppimisele kulus oli liiga suur. D3.js teegi puhul on plussiks suurem paindlikkus eri tüüpi graafikute loomisel, kuid Chart.js pakub kõigi vajalike graafikutüüpide tuge ilma SVG kirjutamise vaevata.

[1] https://github.com/BrendaUga/BrendaUga.github.io

[2] http://www.chartjs.org/

[3] https://d3js.org/

Rakenduse masinõppe osas on realiseeritud `Outputter` klass, milles on implementeeritud mitmed väljundi väljastamisega tegelevad funktsioonid. Graafikute loomiseks vajaliku info salvestab rakendus JSON failidesse, mis on eristatavad kasutajanime ja ajatempliga. Antud JSON faile kasutab JavaScripti teek Chart.js, mis loeb info failist ning loob selle põhjal eri tüüpi graafikuid.

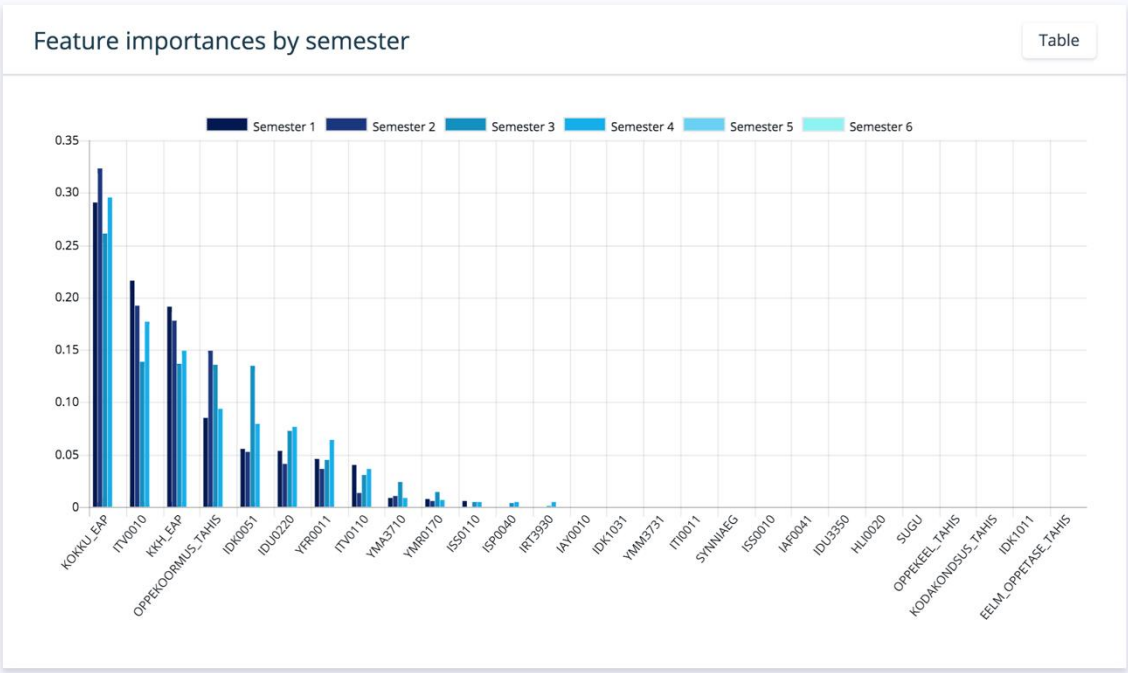
Rakenduse visualiseerimise osa eesmärk on pakkuda kiiret ja lihtsat ülevaadet ennustamise tulemuste kohta. Selline ülevaade on kasulik, et pakkuda kasutajale võimalust leida seoseid tulemuste ja algandmete vahel ning et näha muutusi aja ja erinevate andmehulkade muutumisel. Rakendusel on ka konkreetsemad eesmärgid, mida aitab lahendada visualiseeritud andmete osa.

Kõige olulisemaks eesmärgiks rakenduse kasutamisel on leida üles tudengid, keda ohustab väljalangemise oht. Sellise ennustuse teeb masinõpe, kuid tulemus on vaja välja kuvada kasutajale mugaval viisil. Tulemus kuvatakse põhjaliku tabeli kaudu, kus on kirjas tudengi identifikaator, tema kohta käiv ennustus, võimalusel ka tõenäosused mõlema klassi kohta ning semester, mille soorituste põhjal otsus tehti. Täiendavalt on ära märgitud riskigruppi kuuluvate tudengite read tabelis, et neile oleks võimalik tähelepanu juhtida. Tõenäosustega tabel on eraldatud ainult ennustuse binaarväärtusega tabelist, et oleks mugavam jälgida erinevat tüüpi infot ning et ennustuse tabel oleks järjepidevalt sama erinevate algoritmide kasutamisel. Tõenäosuste tabel kuvatakse välja juhul, kui on kasutatud algoritmi, mis võimaldab tõenäosuste arvutamist.

Ennustuste tabeleid toetab ülevaatlik sektsioon graafikute lehel. Selles sektsioonis on kuvatud kiirelt leitav numbriline info väljalangemise ohus ja mitte väljalangemise ohus olevate tudengite arvu kohta. Samuti on antud sektsioonis leitav protsendiline väärtus, mis näitab väljalangemisohus ja mitte väljalangemise ohus olevate tudengite arvu kõigist andmehulgas olnud tudengite arvust. Selline tähtsa üldinfo näitamine tekstilisel kujul on efektiivsem kui graafikuna kujutamine juhul, kui puudub vajadus pakkuda võrdlusmomenti erinevat väärtuste vahel [3, lk 133]. Ülevaatlik sektsiooni aitab kasutajal kiirelt leida infot selle kohta, kui paljud tudengid on üldse väljalangemisohus ja kui paljusid tudengeid see oht ei puuduta.

Rakenduse eesmärgiks on lisaks eelnevale ka hõlbustada kasutajal kõige rohkem väljalangemisohtu tekitavate tunnuste leidmist ja analüüsimist. Selleks pakub rakendus

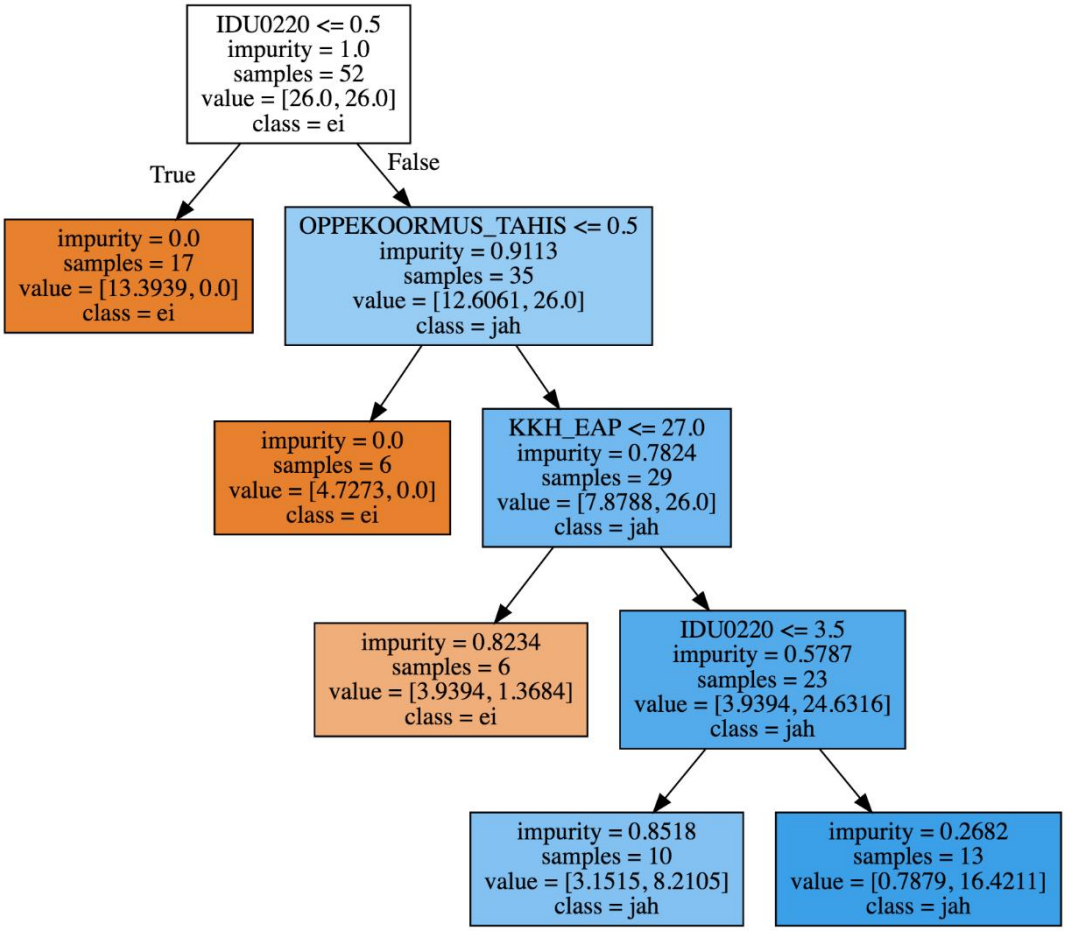
võimalust mitmel viisil vaadata tunnuste tähtsusi juhul, kui on valitud algoritm, mis tunnuste tähtsust võimaldab arvutada. Üheks viisiks tunnuste tähtsuse näitamiseks on tavaline tabel, kus on ära märgitud tunnuste nimetus, tunnuse Gini puhtuse koefitsient ning semester, mille puhul see tunnus esines antud koefitsiendiga. Teiseks viisiks on tulpdiagramm, mis näitab tunnuste tähtsusi semestrite lõikes ning pakub lihtsamat viisi erinevate tunnuste tähtsuste võrdlemiseks ning semestrite lõikes muutumise vaatlemiseks (Joonis 8). Tulpdiagramm on valitud selle info kuvamiseks, sest tulpdiagrammid on heaks valikuks kategoorilise ja kvantitatiivse info samaaegseks kuvamiseks. Samuti on tulpdiagramm parim valik kvantitatiivse info võrdlemiseks kategooriate vahel [3, lk 110].



Joonis 8. Tulpdiagramm tunnuste tähtsuste kohta.

Tunnuste tähtsused üksi ei paku täit ülevaadet väljalangemise ohu tekkimise põhjustest. Oluline on leida üles konkreetsed väärtused iga tunnuse puhul, mis on läveks ühte või teise klassi liigitamisel. Sellist võimalust pakub kasutatud Scikit-learn teegis ainult otsustuspuu klassifitseerija, mille puhul on võimalik visualiseerida konkreetset otsustuspuud. Otsustuspuus on näha iga sõlme puhul valitud muutuja, mis liigitas objektid klassidesse kõige puhtamalt (Joonis 9). Muutuja juures on ka ära toodud väärtus, millest kõrgem või madalam väärtus tingib ennustamisel väljalangemise ohu puudumise või olemasolu. Iga sõlm on värvitud vastavalt määratud klassile ning tumedamad värvitoonid tähendavad suuremat puhtuse skoori antud sõlmes. Otsustuspuu visualiseerimine pakub võimaluse leida andmestikust konkreetsed põhjused, miks tudeng on klassifitseeritud

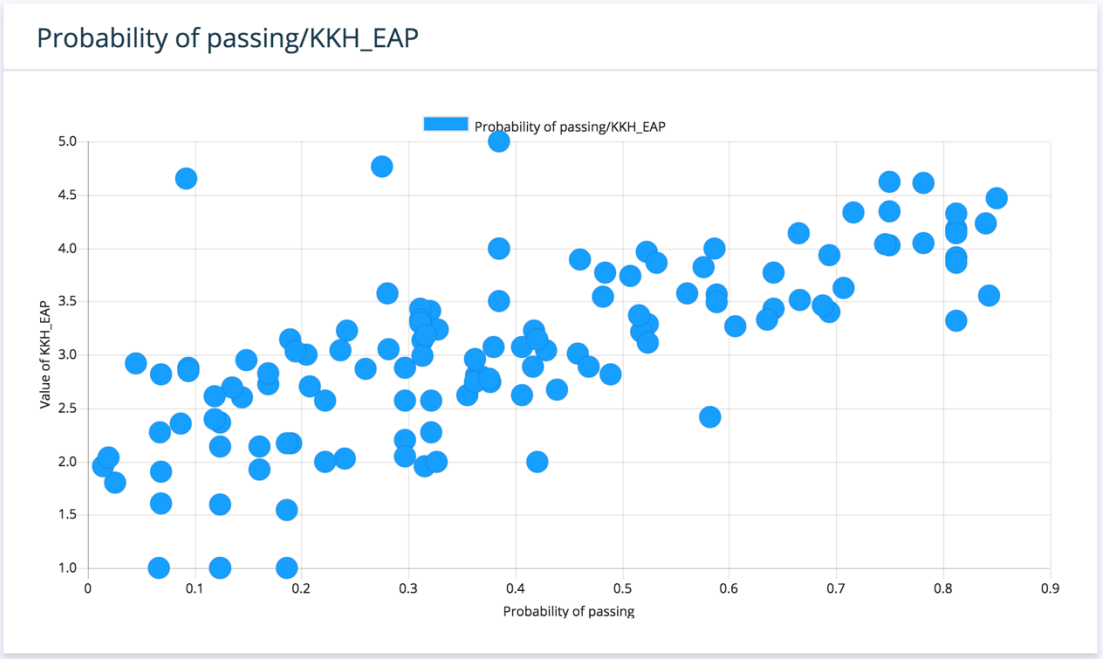
ohugruppi või mitte. Selline info annab võimaluse pakkuda tudengile konkreetseid juhiseid väljalangemisohu vähendamiseks. Otsustuspuu visuaal on loodud Scikit-learn teegi ja pyDot[1] teegi võimalusi kasutades. Otsustuspuu klassifitseerijast genereeritud `Tree` objekti on võimalik pyDot teegis salvestada pildifaili ning seda rakenduses kuvada.



Joonis 9. Otsustuspuu visualiseerimine veebirakenduses.

Lisaks otsustuspuu visualiseerimisele on rakenduses tunnuste tähtsuste ja väljalangemise tõenäosuse seose uurimiseks ka hajuvusdiagramm (Joonis 10). Hajuvusdiagrammil seatakse vastavusse kõige määravama tunnuse väärtus iga andmestikus esineva tudengi kohta ning tõenäosus saavutada positiivne klass ehk tõenäosus mitte kuuluda väljalangemise riskigruppi. Sellisel viisil on kasutajal võimalus leida ka konkreetsemaid põhjuseid väljalangemise ohu tekkimise kohta, nähes milline tunnus kõige rohkem seda mõjutab ning milline on tudengi puhul väärtus selle tunnuse kohta.

[1] https://github.com/erocarrera/pydot



Joonis 10. Hajuvusdiagramm rakenduses.

4.3.1 Tabelite ja graafikute loomisel järgitud põhimõtted

Rakenduses kasutavate tabelite ja graafikute loomisel on autor kasutanud mitmeid üldlevinud põhimõtteid, mis tagavad mugava kasutatavuse ning hea ligipääsetavuse kõigile kasutajatele. Arvestatud on nii ligipääsetavuse tagamise põhimõtetega nagu näiteks piisava kontrasti tagamisega ja selge värvipaleti kasutamisega, kui ka visualiseerimise meediumite loomise põhimõtetega nagu näiteks paigutuse, võrdlusmomendi tekitamise ja õiget tüüpi graafikute kasutamisega.

Dashboard tüüpi rakenduste põhiidee on näidata välja võimalikult palju olulist infot korraga nähtavalt. Selleks on parim viis paigutada kõik info selliselt, et lehte ei peaks kerima. Antud rakenduse puhul on kuvatavat infot nii palju, et sellist ilma kerimiseta lehte oleks väga keeruline või lausa võimatu luua. Sellel põhjusel kasutab autor põhimõtet hoida alati kõige tähtsam info ekraani nähtavas osas (*above the fold)* ning teisene info võib tulla nähtavale lehte kerides. Paigutuse alla liigitub ka graafikute ja tabelite asetamine selliselt, et samalaadne info on lähestikku. Nii tekib võimalus kasutajal kergemalt võrrelda ja korraga vaadelda sama tüüpi infot. Lisaks paigutusele pakub rakendus ka võimalust graafikute ja neile vastavate tabelite vahel liikuda nuppudega, mis tekitavad ühenduse omavahel seotud graafiku ja tabeli vahel.

Tabelite loomisel on arvestatud faktiga, et nende kasutamine on parem informatsiooni otsimiseks kui võrdlemiseks, võrreldes eri tüüpi graafikutega. Nimelt on tabelitest lihtsam

üles leida konkreetselt üksikuid huvitavaid väärtusi, kuid graafikud on paremad võrdlusmomendi tekitamiseks erinevate väärtuste vahel. Sellel põhjusel on konkreetsete tudengite väljakukkumise ennustused kuvatud just tabelitega.

Tabelite loomisel on oluline silmas pidada, et tabelis kuvatud info oleks prominentsem kui tabel ise. Selleks on vaja hoida tabeli read üksteisest eraldatud joonteta ning õrnade värvidega. Kõik pikslid, mis ei kuva tabeli infot pigem segavad tabeli lugemist ning nende hulka on tarvis vähendada, et parandada tabeli loetavust [3, lk 86-92]. Tabelite ridade eraldamiseks piisab hallikatest toonidest paaris ja paaritute ridade taustal ning piisava suurusega tühemikest. Sellisel viisil loodud tabelid on lihtsamad lugeda ning need ei sega infole fokuseerimist.

Graafikud on loodud andmete võrdlemiseks, mitte üksikute väärtuste leidmiseks ning rakenduses realiseeritud graafikud toetavad erinevate muutujate võrdlemist. Graafikute loomisel tuleb arvestada, et graafikud oleksid loetavad kõigile kasutajatele. Ligipääsetavuse põhiprintsiibid nagu piisav kontrast tekstide ja tausta vahel, värvipalett, mis on nähtav ka eri tüüpi värvipimeduse puhul ning õigete HTML atribuutide kasutamine ekraanilugeritel lugemise võimaldamiseks on kõik kasutusel ka antud rakenduses. Kontrasti suurendamine taustade ja tekstide vahel aitab parandada ka terve silmanägemisega kasutajate kogemust, tõstes esile rakenduses olulist ehk infot, mida kuvatakse. Värvipaleti loomisel on arvestatud tõsiasjaga, et eri toonide asemel on parem kasutada sama tooni erinevate intensiivsustega. Nii on värvidega kodeeritud info üheselt mõistetav nii värvipimeduse all kannatavatele kasutajatele kui ka terve silmanägemisega kasutajatele. Lisaks eelnevale on tarvis kasutada ka ekraanilugeritele mõeldud HTML atribuute, mis annavad ekraanilugeri kasutajale heliliselt edasi sellist infot, mida tavakasutaja ekraanil silmadega näeks. Sellisteks atribuutideks on näiteks piltide *alt* atribuut, veaolukordade puhul vormiväljadel kasutatavad viga tähistavad klassid ning graafikute info kuvamine ka tabelite kujul. Nende põhimõtetega arvestatav rakendus on ligipääsetav kõigile kasutajatele [6].

5 Tulemuste analüüs

Treenitud mudelite peal ennustatud tulemustest saab teha mitmeid järeldusi. Antud peatükis analüüsib autor saadud tulemusi.

5.1 Järeldused treenitud mudelitest

Töö käigus treenitud mudelid erinevate andmehulkade ja algoritmidega annavad mitmesugust infot. Erinevate semestrite sooritustega saadud tulemused samade algoritmidega erinevad üksteisest kõigi täpsuse hindamise meetrikate skooride poolest. Paremaid tulemusi näitas hilisemate semestrite hinnete põhjal ennustamine, mis on seletatav asjaoluga, et kumulatiivselt semestrite sooritusi võttes on hilisematel semestritel rohkem kasulikke tunnuseid, mille järgi mudel õppida saab. Rohkemate tunnuste kasutamisel saadi paremad tulemused õigsuse, saagise ja f1-skoori puhul. Täpsus on stabiilselt vahemikus 0.80 - 0.90 mõlema soorituste liitmise viisi puhul.

Algoritmid, mis andsid paremaid tulemusi kõigil meetrikatel, on otsustuspuu, otsustusmets ning mitmekihiline pertseptron (Tabel 1). Võrreldes teiste proovitud algoritmidega on neil 10-20% paremad tulemused õigsuse, saagise ja f1-skoori puhul. Vahe on tingitud andmestiku klasside ebatasakaalust. Nimelt on otsustuspuudel põhinevatel algoritmidel ning närvivõrkude algoritmidel võimalik määrata klasside esinemise suhet. See aitab vältida klasside ebatasakaalust tulenevat käitumist, kus mudel eelistab ennustamisel tugevalt rohkem esinevat klassi. Kõige paremini esinevaks algoritmiks kõigi meetrikate poolest on otsustuspuu, mis on realiseeritud Scikit-learn teeki kasutades.

Võrreldes omavahel soorituste ja üldandmete ning ainult üldandmete põhjal ennustamise tulemusi, on näha, et nende kõikide meetrikate skoorid jäävad üldiselt samasse vahemikku. Erinevused tekivad otsustuspuude ja otsustusmetsade kasutamisel, kui võrrelda esimese semestri soorituste info ja kõigi semestrite üldinfoga saadud tulemusi. See on selgitatav sellega, et otsustuspuudel põhinevad klassifitseerimisalgoritmid vajavad suuremat andmehulka, et luua kasulikke üldistusi andmete peal. Siiski on esimese semestri soorituste pealt saadud tulemused piisavalt kõrge täpsuse skooriga, mis on eelduseks sellise mudeli kasutamisel tudengite väljalangemisohu hindamisel ka juba varases ülikooliõpingute staadiumis.

Üldiselt esines kõigi mudelite puhul mõningast ülesobitumist. Ülesobitumine võib olla tingitud nii väiksest andmete hulgast, vähesest tunnuste arvust kui ka asjaolust, et erinevate aastate tudengite ja õppeainete puhul kehtivad erinevad seosed. Ülesobitumise vältimiseks kasutatud *cross-validation*, hüperparameetrite täpsustamine, eraldi treenimise, valideerimise ja testimise andmehulkade kasutamine ning andmete standardskaleerimine olid osaliselt efektiivsed. Siiski ei piisanud kasutatud meetmete rakendamisest, et täielikult eemaldada ülesobitumist. Võimalikuks variandiks, mida saaks proovida oleks üle- või alasämplimine, et vähendada klasside ebatasakaalu. Samuti aitaks ülesobitumist vältida rohkemate andmete kasutamine mudelite treenimiseks.

Rohkemate andmete olemasolu oleks mudelite treenimise ja hindamise seisukohalt kindlasti kasulik. Mida rohkem on tudengeid, kelle kohta on andmeid mudelite treenimiseks, seda paremini õpib mudel üldistama, mitte konkreetsete andmete mustreid meelde jätma. Nii väheneks nii ülesobitumine kui paraneksid ka ennustamise puhul kõigi hindamise meetrikate skoorid. Samuti aitaks suurem andmehulk teha kasulikemaid järeldusi mudelite ja tulemuste osas.

5.2 Järeldused ennustatud tulemustest

Ennustatud tulemustest saab järeldada, et üldiselt suudavad treenitud mudelid ennustada tudengite väljalangemist piisavalt hästi, et ennustusi kasutada ennetavate tegevuste rakendamisel.

Antud probleemi puhul on tähtsaimaks leida üles riskigruppi kuuluvad tudengid võimalikult vara, et nendega oleks võimalik õigeaegset sekkumist rakendada. Selleks on võimalik ennustada tudengi väljakukkumist juba enne esimese semestri algustki – kasutades ainult üldandmetel põhinevat ennustamist. Samuti on võimalik juba esimese semestri soorituste põhjal öelda üle 85% täpsusega, kes on potentsiaalselt väljakukkumisohus. Rakenduses välja toodud ennustamise tulemused ja ainete tähtsused on abiks riskigrupi tudengite ja riskitekke põhjuste leidmiseks.

Soorituste ja üldandmete põhjal ennustades on kõige olulisemaks faktoriks, mille järgi saab klassifitseerida tudengit väljalangejaks, kokku kogutud EAP-de arv. Antud leid on kooskõlas ka TTÜ õppeosakonna poolt koostatud väljalangenute uuringuga, milles on välja toodud, et kogutud EAP-de arv on üheks suurimaks mõjutajaks väljalangemise või

koolis jätkamise vahel. Nimelt on TTÜ uuringus analüüsitud väljalangenud tudengite erinevaid statistilisi andmeid ning läbi viidud ka uuring nende seas. Uuringust tuleb välja, et tudengid, kes on kogunud 151 või enam EAP-d, on kõige motiveeritumad õpinguid jätkama [5]. Sarnaselt antud töö leidudega, on kogutud EAP-de arvu suur määramisfaktor kooskõlas TTÜ uuringu leituga [5].

Ainete tähtsuste puhul võib järeldada, et erialaained on kaalukamad tudengi väljalangemise määramisel. Üle erinevate katsete informaatika õppekava tudengite puhul olid stabiilselt kõige määravama tähtsusega ained Programmeerimise algkursus (IDK1011), Lineaaralgebra (YMA3710), Objektorienteeritud programmeerimine keeles Java (IDK0051), Andmebaasid I (IDU0220) ja Arvutid (IAF0041). Viiest kõige kaalukamast ainest on neli erialaained. Samuti on kõrge kaaluga ka matemaatika ained Lineaaralgebra ja Matemaatiline analüüs I. Need matemaatika ained osutuvad tudengitele raskeks mitmel põhjusel. Autori arvamusel põhjustavad ainete raskust ainete programmiga eelneva kokkupuute puudumine ja väiksem seos erialaga kui näiteks Diskreetse matemaatika ainetel.

Üldandmete põhjal ennustamisel on kõige tähtsamad tunnused EAP-de arv, keskmine hinne, eelmise õppetaseme tähis ja sisseastumisaasta. Kõige väiksema kaaluga on sugu ja kodakondsus. Võib järeldada, et tudengite õppetööga seotud faktorid on väljalangemise ohu kujunemisel suuremaks määrajaks, sest on otseselt seotud õppeedukuse ja õppetööst osavõtmise motiveeritusega, mis mängivad suuremat rolli omakorda väljalangemise ohu tekkimisel. Samal ajal on sugu ja kodakondsus madala koefitsiendiga. Võib järeldada, et mittemuudetavad isikuomadused nagu sugu, kodakondsus ja sünniaasta ei oma tähtsust väljalangemise ohu tekkimisel, sest ei ole seotud õppetööga. Samuti võib öelda, et nii naised kui mehed ning nii Eesti kui Venemaa kodanikud on võrdselt võimekad õppetööd lõpetama õigeaegselt. Üldandmete seast välja tulnud suure faktoriga tunnused võimaldavad ennustada tudengite väljalangemise ohtu ka juba enne õppetöö algust, võttes aluseks tudengi sisseastumisaasta, eelmise õppetaseme tähise ja sünniaja, mis kõik mõjutavad tudengi kuulumist riskigruppi või mitte.

Kuigi kogutud EAP-de arv osutus kõige kaalukamaks faktoriks väljalangemise riskigruppi kuulumisel, on saadud info petlik. Nimelt on kasutatud CSV andmestikus KOKKU_EAP tulbas välja toodud väärtused andmete viimase uuendamise seisuga. Ühendades tudengite sooritused üldandmetega, tekib olukord, kus näiteks esimese

semestri sooritused on ühes *dataframe*'s koos hilisematele semestritele vastava kogutud EAP-de arvuga. Antud olukorras osutub just kogutud EAP-de arv kõige määravamaks tunnuseks nii suure määraga, et teised tunnused jäävad mudeli poolt üldse kasutamata. Kuna kogutud EAP-de arv on andmestikus viimase läbitud semestri seisuga, siis varasemate semestrite peal ennustamisel ei paku see tunnus kõige õigemat infot. Sellisel põhjusel jätan edaspidi EAP-de arvu välja kasutatavast andmehulgast.

Mõlema kasutatud andmestiku puhul on tulemused kõigi meetrikate osas samas täpsusvahemikus. JSON formaadis andmete puhul on kasutatavad andmed uuemad ning võimaldavad ennustada ka tänaste tudengite puhul. CSV formaadi kasutamisel on tänaste tudengite puhul ennustamine raskendatud, sest õppimise andmestikus olevate tudengite puhul on tegemist teise õppekava versioonil õppivate tudengite ning soorituste puhul on raskendatud selliste õppeainete leidmine, mis oleksid kõigis eri versioonides esindatud. Samuti on uuemate andmete kasutamise puhul kasulik andmete aktuaalsus nii õppekava kohustuslike ainete suhtes kui ka ainete sisu suhtes. Sellega on põhjendatav ka JSON formaadis andmetel põhinevate mudelite parem sooritus tänaste tudengite peal ennustades.

Kuigi ennustamise täpsused jäävad soovitud täpsusvahemikku ning täidavad püstitatud eesmärke, on võimalik andmete puhul teha ka täiendavaid teisendusi ja kasutada erinevaid tehnikaid mudelite arendamisel. Tulevikus oleks kasulik muuta ennustamine kaheklassilisest mitmeklassiliseks, et ennustamisel määrata mitte lõpetamine, lõpetamine nominaalajaga ja lõpetamine rohkem kui ettenähtud 6 semestriga. Selliselt saadud info võimaldaks rakendada erinevaid meetmeid tudengitele, kellele ennustati lõpetamist nominaalajast rohkema ajaga ning neile, kellele ennustati mitte lõpetamist. Samuti oleks võimalik eraldi analüüsida põhjuseid, mis tingivad erinevate klasside esinemist ennustamisel. Lisaks mitmeklassilisele ennustamisele on võimalik mudelid arendada veelgi täpsemaks erinevaid tehnikaid kasutades. Võimalikeks parandusteks pakub autor näiteks suurema andmehulga kasutamist, hüperparameetrite täiendavat täpsustamist ning erinevate algoritmide proovimist koos regulariseerimisega. Samuti pakub autor välja kasutada ka semestri jooksul saadud tulemusi erinevatest ainetest, mis võimaldab väljalangemisohu tekkimise põhjuseid otsida rohkem sügavuti.

6 Kokkuvõte

Antud töö eesmärgiks oli luua süsteem, mis võimaldab kasutajal hõlpsalt leida üles väljalangemise riskigruppi kuuluvad tudengid. Eesmärgi täitmiseks loodi masinõppe mudelid kasutatavate andmestike põhjal, valideeriti ja hinnati mudelite tööd neile tundmatutel andmetel ning loodi veebirakendus masinõppe protsessi käivitamiseks ning tulemuste visualiseerimiseks.

Töös püstitatud eesmärk viia masinõppe mudelite täpsuste skoorid 80% peale sai täidetud. Eesmärgi täitmiseks tehti katsed mitut erinevat klassifitseerimisalgoritmi kasutades, samuti loodi närvivõrgu arhitektuur. Parimad mudelid ennustavad tundmatute andmete peal rohkem kui 85% täpsusega tudengite väljalangemise riskigruppi kuulumist. Samuti on mudelite pealt võimalik leida andmestikus esinevaid tunnuseid, mis kõige rohkem mõjutavad tudengi kuulumist riskigruppi. Töö tulemusena loodud mudelid on piisavalt täpsed, et leiaksid rakendust nii õpingute varases kui ka hilises staadiumis väljalangemise ennetamisel.

Töö teine fookus oli andmete visualiseerimisel, mille tulemusena valmis veebirakendus, kus on võimalik käivitada masinõppe protsessi sisestatud andmete põhjal ning vaadata visualiseeritud kujul erinevaid andmeid nii antud sisendist kui ka ennustamise tulemusest. Kasutatavuse ning andmete visualiseerimise parimaid praktikaid järgides loodud tabelid ning graafikuid eri tüüpi andmete kuvamiseks on abiks ennustuse analüüsimiseks.

Töö tulemusena valminud rakendus on kergelt laiendatav nii masinõppe kui ka veebirakenduse poolel, mis võimaldab rakenduse jätkuva edasiarendamise. Edaspidi on plaanis kasutada hiljuti saadud uuemaid andmeid mudelite loomiseks, mis võimaldavad paremat täpsust praeguste tudengite peal ennustades. Lisaks eelnevale on võimalik tulevikus lisada mitmeid erinevaid andmeallikaid, kaasates infot ka semestri jooksul saadud soorituste kohta õppeainetes.

Kirjanduse loetelu

[1] Artificial neural network. [WWW] https://en.wikipedia.org/wiki/Artificial_neural_network (09.04.2017)

[2] Decision Tree Classifier. [WWW] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier (09.04.2017)

[3] Few, S. Information Dashboard Design: The Effective Visual Communication of Data. Sebastopol : O'Reilly Media, 2006.

[4] Géron, A. Hands-On Machine Learning with Scikit-Learn. Sebastopol : O'Reilly Media, 2017.

[5] Tudengimonitooring: rahvusvahelise kogemuse kaardistus / H. Haavapuu, L. Mere, V. Kütt, I. Pürjema, K. Rebane. Tallinn, 2016.

[6] How to Meet WCAG 2.0. [WWW] https://www.w3.org/WAI/WCAG20/quickref/ (24.04.2017)

[7] K-nearest neighbors algorithm. [WWW] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (09.04.2017)

[8] Multilayer perceptron. [WWW] https://en.wikipedia.org/wiki/Multilayer_perceptron (09.04.2017)

[9] Nearest Neighbors. [WWW] http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors (09.04.2017)

[10] Raschka, S. Python Machine Learning. Birmingham : Packt Publishing, 2015.

[11] sklearn.feature_selection.SelectKBest. [WWW] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest (09.04.2017)

[12] Šults, K. Õpingute katkestamise põhjused ja õpingutega jätkamise motivatsioon. Tallinn, 2015.

[13] TensorFlow. [WWW] https://www.tensorflow.org/ (09.04.2017)

Lisa 1 – Võrkotsingu parameetrid

Klassifitseerija	Parameetrid
LogisticRegression	{'C': [1, 10, 100, 1000], 'class_weight': ['balanced']}
KNeighborsClassifier	{'n_neighbors': [3, 5, 10], 'weights': ['distance', 'uniform'], 'algorithm': ['brute', 'kd_tree', 'ball_tree']}
SVC	{'C': [1, 10, 100, 1000], 'kernel': ['linear'], 'class_weight': ['balanced']}, {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf', 'poly'], 'class_weight': ['balanced']}
DecisionTreeClassifier	{'criterion': ['entropy', 'gini'], 'min_samples_split': [15, 20, 25], 'min_samples_leaf': [5, 10, 15], 'class_weight': ['balanced']}
RandomForestClassifier	{'criterion': ['entropy'], 'n_estimators': [10, 20, 30], 'min_samples_split': [10, 15, 25, 30], 'class_weight': ['balanced']}
MLPClassifier	{'hidden_layer_sizes': [(5,2), (15,), (10,10,10)], 'solver': ['lbfgs'], 'alpha': [1e-5, 0.01, 0.1]}