TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Gunnar Joosep Hint
203888IABM

Tootetoe probleemide jaotamine arendusmeeskondadesse kasutades teksti klassifitseerimise meetodeid

Magistritöö

Juhendaja: Ahti Lohk
PhD

Tallinn 2023

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: Gunnar Joosep Hint

03.01.2023

Annotatsioon

Käesoleva töö eesmärgiks on luua EasyPark ettevõtte näitel lahendus, mis suudaks teksti kujul esitatud tootetoe probleeme jaotada vastavatesse arendusmeeskondadesse kasutades teksti klassifitseerimise meetodeid. Töö eesmärgini jõudmiseks selgitakse kirjanduse ja etalonkorpuste alusel välja kaks pinnapealset meetodit (Naive Bayes, SVM) ning kaks süvameetodit (RoBERTa, XLNet) ja analüüsitakse, milline neist on klassifitseerimistäpsuse osas efektiivseim.

Parima mudeli leidmiseks kasutatakse kordustäpsust, saagist ja F1-skoori. Seejärel võrreldakse parima mudeli täpsust inimesega. Samuti selgitatakse välja, millistesse arendusmeeskondadesse (klassidesse) suudab parim klassifitseerimismeetod kõige paremini klassifitseerida ja millistesse kõige halvemini. Vastavalt analüüsi järeldustele pakutakse soovitused tootetoe probleemide paremaks klassifitseerimiseks.

Töö väljund koosneb parima mudeli kasutusele võtmisest Jira keskkonnas. Selleks luuakse Jira pistikprogramm, mida toetab vastav taustteenus koos parima treenitud mudeliga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 17 joonist, 14 tabelit.

Abstract

Distribution of product support problems to development teams using text classification methods

The purpose of this thesis is to create a solution based on EasyPark company that could distribute product support problems to the respective development teams using text classification methods. To reach the goal of the thesis, two shallow methods (Naive Bayes, SVM) and two deep methods (XLNet, RoBERTa) were used, which were identified based on literature overview and benchmark datasets. An analysis was made to identify which of the four methods was the most effective in terms of classification accuracy.

The best model is found by using precision, recall and F1-score. The best model is then compared to human classifications and an overview is provided as to which development teams (classes) can be classified better or worse. According to the conclusions of the analysis, recommendations are also made to improve the product support problem classification.

The realization of the thesis consists of training the models and deploying them in the Jira environment. For deployment, a Jira plugin is created, backed by a corresponding backend service with the trained model.

The thesis is in Estonian and contains 41 pages of text, 6 chapters, 17 figures, 14 tables.

Lühendite ja mõistete sõnastik

Jira	IT-ettevõtete haldustarkvara, mida pakub Atlassian
TPR	*True positive rate* ehk saagis (*recall*)
PPV	*Positive predivice value* ehk kordustäpsus (*precision*)
ACC	Täpsus (*accuracy*)
TF-IDF	Sõnade olulisusskoori arvutamise meetod
XLNet	Eeltreenitud keele mudel
RoBERTa	*Robustly optimized BERT approach* – eeltreenitud keele mudel, edasiarendus BERT mudelist
BERT	*Bidirectional encoder representations from Transformers* – eeltreenitud keele mudel
SVM	Tugivektormasin (*Support vector machine*), binaarse klassifitseerimise mudel
C	Regularisatsiooni parameeter, mida kasutatakse SVM mudelis
NB	Naive Bayes, Bayesi teoreemil põhinev masinõppe mudel
MNB	Multinomiaalne Naive Bayes
F1-skoor	Harmooniline keskmine kordustäpsuse ja saagise vahel
Epohh	*Epoch*, üks treeningtsükkel kogu treeningandmestikuga
Learning rate	Hüperparameeter, millega kontrollitakse, kui palju mudelit vastavalt hinnangulisele veale muuta iga kord, kui mudeli kaalusid uuendatakse
Ploki suurus	*Batch size*, hüperparameeter, millega kontrollitakse treeningandmete arvu igal iteratsioonil

Sisukord

1 Sissejuhatus 10
2 Tootetoe ülevaade 12
2.1 Ettevõtte tootetoe taust 12
2.2 Tootetoe andmestiku kirjeldus 13
3 Metoodika 16
3.1 Teksti klassifitseerimismeetodite ülevaade 16
3.1.1 TF-IDFi ülevaade 18
3.1.2 RoBERTa ülevaade 19
3.1.3 XLNeti ülevaade 20
3.1.4 Tugivektormasina ülevaade 21
3.1.5 Naive Bayesi ülevaade 22
3.2 Tulemuste valideerimine 24
4 Realisatsioon 27
4.1 Töö tegevuskäik 27
4.2 Andmete eeltöötlus 28
4.3 Klassifitseerimismudelite realisatsioon 31
4.4 Taustteenuse kirjeldus 34
4.5 Jira integratsioon 35
5 Analüüs 37
5.1 Klassifitseerimismudelite võrdlus 37
5.2 Parima mudeli võrdlus inimesega 42
5.3 Parima mudeli klassifitseerimisedukus tiimide põhiselt 44
5.4 Võrdlus varasemate töödega 48
5.5 Piirangud ja edasised arengud 49
6 Kokkuvõte 51
Kasutatud kirjandus 52
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 55
Lisa 2 – Mudelite täpsus ja saagis testandmestiku põhjal 56

Lisa 3 – Mudelite täpsus ja saagis valideerimisandmestiku põhjal 57
Lisa 4 – W&B keskkonna kuvatõmmised 58

Jooniste loetelu

Joonis 1 Tootetoe probleemi näide 13
Joonis 2 Treening- ja testandmestiku probleemid arendustiimide lõikes 14
Joonis 3 Uued tootetoe probleemid arendustiimide lõikes 15
Joonis 4 XLNet mudeli arhitektuur [19] 21
Joonis 5 SVM optimaalse hüperplaani näide [21] 21
Joonis 6 Veamaatriksi näide [29] 24
Joonis 7 Mitme klassi veamaatriksi näide [29] 25
Joonis 8 Töö tegevuskäigu diagramm 27
Joonis 9 Treening- ja testandmete jaotus arendustiimide põhiselt 29
Joonis 10 Taustteenuse päringu ja vastuse näide 34
Joonis 11 Jira integratsiooni näide 35
Joonis 12 Mudelite F1-skoorid arendusmeeskondade lõikes testandmestikul 38
Joonis 13 Mudelite F1-skoorid arendusmeeskondade lõikes valideerimisandmestikul . 40
Joonis 14 RoBERTa mudeli veamaatriks 45
Joonis 15 Olulisemad 20 sõnaosa arendusmeeskonna parkimisluba ennustamise suhtes 46
Joonis 16 Olulisemad 20 sõnaosa arendusmeeskonna ärikliendid ennustamise suhtes . 47
Joonis 17 Olulisemad 20 sõnaosa arendusmeeskonna maksed (rakendus) ennustamise suhtes 48

Tabelite loetelu

Tabel 1 Erinevate klassifitseerimismudelite täpsuste ülevaade levinud andmestike põhjal [8], [9], [10], [11] * - Pinnapealsed mudelid 17
Tabel 2 SVM ja Naive Bayes mudeli täpsuse ja treenimiskiiruse võrdlus filmiarvustuste andmestiku põhjal (*Movie review dataset 2.0*) [14] 18
Tabel 3 Probleemide jaotus arendustiimide lõikes 30
Tabel 4 Mudelite hüperparameetrid 33
Tabel 5 Forge käskude kirjeldused [36] 36
Tabel 6 Mudelite kordustäpsus, saagis ja F1-skoor makro keskmise alusel testandmestikul 37
Tabel 7 Mudelite kordustäpsus, saagis ja F1-skoor kaalutud keskmise alusel testandmesikul 38
Tabel 8 Mudelite kordustäpsus, saagis ja F1-skoor makro keskmise alusel valideerimisandmestikul 39
Tabel 9 Mudelite kordustäpsus, saagis ja F1-skoor kaalutud keskmise alusel valideerimisandmestikul 40
Tabel 10 Treenimisaeg mudelite lõikes 41
Tabel 11 Inimeste poolt tehtud ümbersuunamiste arv probleemide lõikes testandmestikul 42
Tabel 12 Inimeste poolt tehtud ümbersuunamiste arv probleemide lõikes valideerimisandmestikul 42
Tabel 13 RoBERTa mudeli ja inimese täpsuse võrdlus test- ja valideerimisandmestikul 43
Tabel 14 RoBERTa mudeli tegeliku tiimi ennustusvead ja ümbersuunatud probleemid tiimide lõikes test- ja valideerimisandmestikul 43

1 Sissejuhatus

Tootearenduse üks lahutamatu osa on tootetugi, mis enamasti tähendab kliendi probleemide lahendamist, toote parandamist ja ka kasutajate koolitamist. Tootetoe eesmärk on võimaldada klientidel saada pakutavalt tootelt võimalikult suurt väärtust. Efektiivse tootetoe pakkumine on ettevõtetes äärmiselt oluline, kuna see võib suurel määral mõjutada klientide rahulolu ning ettevõtete mainet. [1]

Tootetugi on ettevõtete puhul vägagi arvestatav kulu, ulatudes teatud ettevõtete puhul kuni miljonite eurodeni aastas. Lisaks on suurematel ettevõtetel probleemiks see, et pakutavaid tooteid või teenuseid on palju, mistõttu võib klientide probleemide lahendamiseks kuluda kaua aega. Selle põhjusteks võivad olla aeglane probleemide jaotamine tiimidesse või probleemidest valesti aru saamine, mistõttu võidakse neid suunata valedesse tiimidesse. [1], [2]

Kui probleemide jaotamist tehakse inimeste poolt, ei ole see hästi skaleeritav, kuna ettevõtte kasvades, suureneb ka sissetulevate probleemide hulk. Lisaks, kui ettevõttel on rohkem tooteid ja tiime, suureneb inimestel kognitiivne koormus, kuna nad peavad omama ülevaadet ettevõtte struktuuri kohta, et jaotamise osas otsuseid teha.

Käesoleva magistritöö eesmärk on luua EasyPark ettevõtte näitel lahendus, mis suudaks jaotada tootetoe probleeme vastavatasse arendusmeeskondadesse kasutades teksti klassifitseerimise meetodeid. Eesmärgini jõudmiseks on uuritud nelja erinevat meetodit, millest kaks on pinnapealsed meetodid ning kaks on sügavad meetodid. Lahenduse kasutuselevõtmiseks luuakse integratsioon Jira keskkonnaga. Töös on püstitatud neli küsimust, millele analüüsi käigus vastatakse:

- Milline on kõige sobivam mudel tootetoe probleemide klassifitseerimiseks?
- Kuidas toimib parim klassifitseerimismudel võrreldes inimesega?
- Millistesse tiimidesse toimib klassifitseerimine kõige paremini? Miks?
- Millistesse tiimidesse toimib klassifitseerimine kõige halvemini? Miks?

Käesolevas töös on uuritud ettevõtte EasyPark tootetoe probleemide klassifitseerimist, mis on tekstide kujul ja pärinevad Jira keskkonnast. Andmete omanik on EasyPark ning sisaldab sensitiivset informatsiooni, mistõttu pole neid töö raames avalikustatud.

Kuigi tootetoe klassifitseerimise valdkond on võrdlemisi spetsiifiline, siis on varasemalt tehtud ka sarnaste valdkondade klassifitseerimise kohta uuringuid nagu näiteks Istanbuli Tehnikaülikooli kasutajatugi või defektide raporteerimine ABB ettevõttes. Nendes töödes on kasutatud meetoditeks TF-IDF, Naive Bayes, tugivektormasin, otsustuspuud ja KNN (*K-nearest-neighbor*). [1], [3]

Jira poolt pakutakse samuti automatiseerimise töövahendeid, mis põhinevad kasutajate poolt seadistatud reeglitele. Reegleid saab määrata vastavalt pileti tüübile ja pileti väljade väärtustele. Kuigi see süsteem on reeglite loomise osas paindlik, siis masinõppel ja tekstanalüüsil põhinevat automatiseerimist Jira ei paku. [4]

Antud töö peamine panus on välja tuua meetod, mille abil oleks võimalik automatiseerida tootetoe probleemide klassifitseerimist, mida oleks võimalik rakendada ka teistes ettevõtetes.

Töö on jaotatud viide peatükki, kus teises peatükis antakse ülevaate tootetoe toimimisest ja selle andmestikust, kolmandas peatükis tuuakse välja ülevaade klassifitseerimis- ja valideerimismeetoditest, neljandas peatükis antakse ülevaade tehnilise realisatsiooni kohta meetodite treenimise ja pistikprogrammi osas. Viiendas peatükis tuuakse välja tulemused, tehakse nende põhjal analüüs ning vastatakse püstitatud küsimustele.

2 Tootetoe ülevaade

Käesolevas peatükis on kirjeldatud tootetoe tausta ning selgitatud selle olulisust. Lisaks on kirjeldatud uuritava ettevõtte tootetoe ajalugu ning on välja toodud lühike ülevaade varasematest uurimistöödest.

2.1 Ettevõtte tootetoe taust

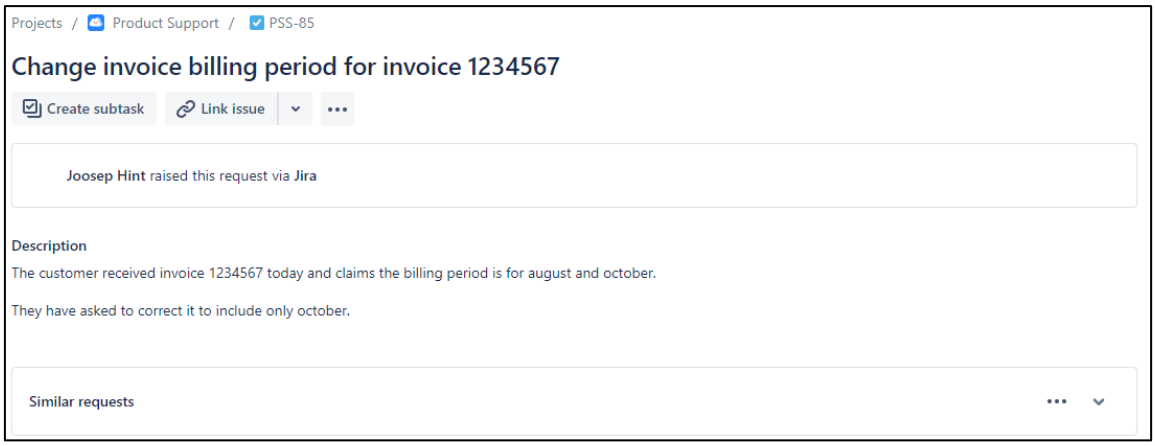
Antud töös on kasutatud tootetoe uurimiseks ettevõtet EasyPark[1], mis on globaalne parkimisi vahendav platvorm. Ettevõte tegutseb 25 erinevas riigis ja enam kui 3200 erinevas linnas. Lisaks parkimise vahendamisele pakutakse ka parkimislubade teenuseid, elektriautode laadimise vahendamist ning andmeanalüüsi teenuseid parkimisoperaatoritele jpm. [5]

Tootetoe sisuks on lahendada erinevaid probleeme, mis klientidel võivad tekkida EasyParki toodete kasutamisel. Tootetoe probleemide haldamiseks kasutab ettevõte Jira platvormi, kus on võimalik probleeme edasi suunata vastavalt sellele, millise arendustiimi valdkonda antud probleem kuulub. Antud ettevõtte puhul toimib tootetugi kolmekihiliselt.

Esimeses kihis on tootetoe töötajad, kes suhtlevad lõppklientidega otse ja osutavad abi probleemi lahendamiseks. Juhul kui esimene kiht ei saa probleemi lahendada, luuakse uus pilet Jira keskkonda, mis suunatakse teise kihi poole, kelleks on tootetoe agendid. Juhul kui ka agendid ei saa probleemi lahendada, suunatakse see probleem agendi poolt edasi vastavale arendusmeeskonnale, mida nimetatakse tinglikult kolmandaks kihiks. Käesolevas töös on uurimisobjektiks kolmanda kihi probleemide jaotamine vastavatesse arendustiimidesse.

Tootetoe probleemide näideteks on probleemid arvete maksmisel, vead kliendi andmetes, probleemid parkimiskohtade otsimisel. Joonisel 1 on välja toodud ümber sõnastatud näide tootetoe probleemist. Joonisel on oluliseks osaks probleemi pealkiri ja kirjeldus, mis on klassifitseerimise sisendiks.

[1] https://easyparkgroup.com/



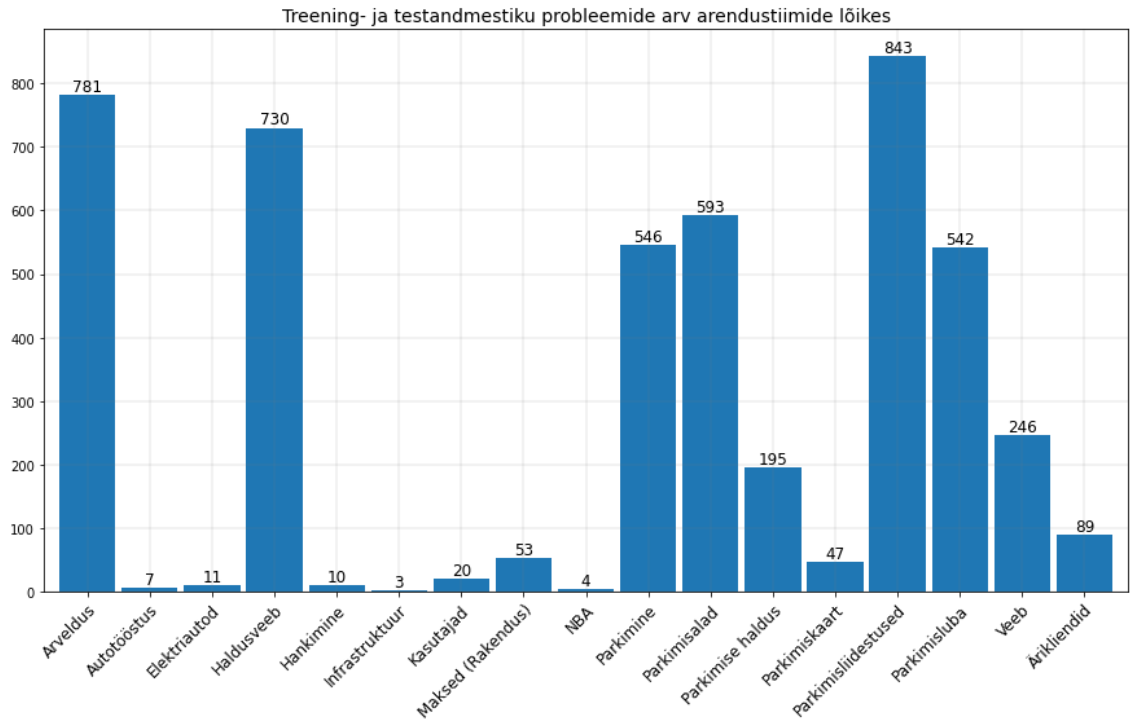
Joonis 1 Tootetoe probleemi näide

2.2 Tootetoe andmestiku kirjeldus

Käesoleva töö raames on andmestikuna kasutatud EasyPark ettevõtte tootetoe ajalugu, mis sisaldab inglisekeelseid tekste erinevate probleemide näol. See ajalugu pärineb Jira keskkonna piletitest aastast 2020-2022. Kuigi ettevõttel on tootetoe ajalugu aastast 2012, siis ei olnud mõistlik antud ülesande raames varasemaid probleeme kasutada. Seda peamiselt seetõttu, et ettevõtte struktuur on aastate jooksul oluliselt muutunud, mistõttu varasemad probleemid ei ole enam klassifitseerimiseks relevantsed.

See ajalugu pärineb kahest erinevast Jira projektist, millest esimene projekt on ajalooline ja pole enam kasutusel. Selle projekti probleemid on perioodist 2012 kuni 2022 algus. Selle projekti puhul oli jaotamine tehtud siltide alusel, millel olid spetsiifilised jaotusreeglid. Nende samade reeglite alusel tehti ka mudeli sisendandmete jaotus tiimidesse. Teine projekt on aktiivselt kasutusel, kus probleemide ajalugu algab aastast 2021. Probleemide jaotus tiimidesse tehti eraldi välja alusel, kus on ära määratud erinevad arendustiimid, mistõttu sai kasutada seda välja, et määrata probleemide kuuluvust. Kahe projekti peale kokku pärinevad andmed 2020 aasta jaanuarist kuni 2022 novembri alguseni.

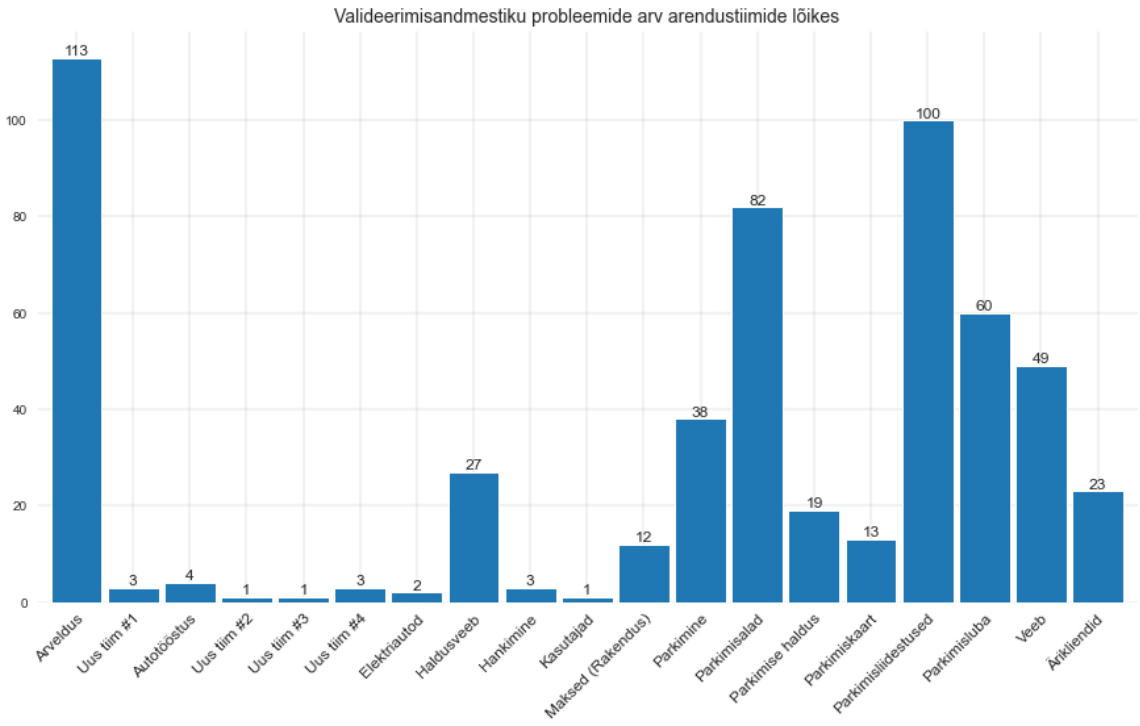
Test- ja treeningandmestiku probleeme kokku on 4720, mille hulgas on arendustiime kokku 17. Keskmine tootetoe probleemi pikkus on 73 sõna. Joonisel 2 on välja toodud arendustiimid koos tootetoe probleemide arvudega.



Joonis 2 Treening- ja testandmestiku probleemid arendustiimide lõikes

Lisaks eelnevalt kirjeldatud andmestikule on kasutatud ka valideerimiseks eraldi andmestikku, mille probleemid pärinevad 2022 aasta oktoobrist kuni novembrini. Selle andmestiku eesmärgiks oli teostada lõplikku valideerimist ja analüüsi. Selle andmestiku puhul oli probleeme kokku 554 ja arendustiime oli 19. Võrreldes eelmise andmestikuga on oluliseks erinevuseks, et ettevõttesse oli juurde tekkinud neli uut arendustiimi.

Joonisel 3 on välja toodud arendustiimid koos uute tootetoe probleemide arvudega. Antud joonisel ei ole kuvatud kahte varasemat tiimi, kuna nendele polnud uusi probleeme tekkinud. Uued andmestiku tiimid on tähistatud viisil ’Uus Tiim #X’, kuna need ei olnud mudelite analüüsi jaoks olulised.



Joonis 3 Uued tootetoe probleemid arendustiimide lõikes

3 Metoodika

Käesolevas peatükis on välja toodud klassifitseerimismeetodite ülevaade koos põhjendustega, miks need valituks said. Lisaks on välja toodud tulemuste valideerimise meetodid, mille alusel on võimalik teha edasist analüüsi.

3.1 Teksti klassifitseerimismeetodite ülevaade

Teksti klassifitseerimine on levinud ülesanne loomuliku keele töötlemise valdkonnas ning selle jaoks on loodud mitmeid meetodeid. Tabelis 1 on välja toodud erinevad klassifitseerimismeetodid levinud etalon-andmestike põhjal, milleks on IMDB, SST-2 ja DBPedia.

SST-2 ja IMDB andmestikud on binaarse sentimentanalüüsi andmestikud, mis koosnevad filmiarvustustest. IMDB koosneb 25 000 filmiarvustusest treenimiseks ja 25 000 arvustusest testimiseks. SST-2 andmestik koosneb 11 855 filmiarvustusest. Mõlema andmestiku klassideks on positiivne või negatiivne. [6], [7]

DBpedia on suuremahuline mitmekeelne teadmistebaas, mis on loodud Wikipedia kõige sagedamini kasutatavatest infokastidest. Antud DBPedia andmestiku versioon sisaldab 560 000 teksti treeningandmete jaoks ja 70 000 teksti valideerimiseks. Andmestiku klasse on 14, mille näideteks on artist, loom, album, film, transpordivahend. [8]

Meetod	**IMDB täpsus (%)**	**SST-2 täpsus (%)**	**DBPedia täpsus (%)**
Naive Bayes*	85.66	81.80	-
LDA*	67.40	-	-
SVM + BoW*	87.80	79.40	-
SVM + TF-IDF*	90.00	-	98.69
Logistiline regressioon + BoW*	89.00	-	-
Logistiline regressioon + TF-IDF*	90.00	-	-
Deep Pyramid CNN	-	84.46	99.12
ULMFiT	95.40	-	99.20
BLSTM-2DCNN	-	89.50	-
NeuralSemanticEncoder	-	89.70	-
BCN+Char+CoVe	91.80	90.3	-

Meetod	**IMDB täpsus (%)**	**SST-2 täpsus (%)**	**DBPedia täpsus (%)**
Virtualadversarialtraining	94.10	-	-
Block-sparseLSTM	94.99	93.20	-
BERT-large	95.79	94.9	99.32
ALBERT	-	95.20	-
Multi-TaskDNN	83.20	95.60	-
SnorkelMeTaL	-	96.20	-
RoBERTa	-	96.40	-
XLNet-Large	96.21	96.80	99.38

Tabel 1 Erinevate klassifitseerimismudelite täpsuste ülevaade levinud andmestike põhjal [8], [9], [10], [11]
* - Pinnapealsed mudelid

Artiklite alusel andsid IMDB andmestikul sügavate meetodite puhul kõige kõrgemaid tulemusi XLNet mudel täpsusega 96.21% ning BERT mudel täpsusega 95.79%. SST-2 andmestiku puhul oli samuti kõige parema tulemusega XLNet mudel täpsusega 96.80%. Sarnast tulemust andis ka RoBERTa mudel täpsusega 96.40%.

DBPedia andmestikul andis samuti XLNet mudel parima täpsuse 99.38% ja sellele sarnase tulemuse andis ka BERT mudel täpsusega 99.32%. Lisaks andis lähedase tulemuse ka pinnapealne meetod SVM koos TF-IDF'ga (98.69%).

Võttes arvesse, et keerulisemate meetodite seas andis artiklite alusel kõige paremaid tulemusi XLNet mudel, siis osutus see üheks mudeliks antud ülesande lahendamisel. Lisaks sellele meetodile andis IMDB andmestiku põhjal hea tulemuse ka BERT mudel. Kuna RoBERTa mudel on BERT mudeli edasiarendus [12] ja andis SST-2 andmestiku põhjal lähedase tulemuse XLNet mudelile, siis osutus teiseks sügava meetodi valikuks RoBERTa.

Pinnapealsete meetodite seas andis IMDB andmestikul kõige paremaid tulemusi SVM (tugivektormasin) + TF-IDF ja Logistiline regressioon + TF-IDF täpsusega 90.00%. Kuna SVM meetodi kohta oli ka informatsiooni DBPedia andmestiku täpsuse kohta ning DBPedia teemade kategoriseerimise ülesanne on sarnane käesoleva töö ülesandega, siis osutus see esimeseks valitud pinnapealseks meetodiks.

Tabelis 2 on näidatud SVM ja Naive Bayes mudelite võrdlus täpsuse ja treenimiskiiruse osas filmiarvustuste andmestiku (*Movie review dataset 2.0*) näitel. Antud filmiarvustuste andmestik koosneb 1000st positiivsest ja negatiivsest arvustusest [13]. Tabeli põhjal on näha, et Naive Bayes treenimise aeg oli peaaegu 135 korda kiirem, kuid täpsuse erinevus oli alla 2%.

Mudel	**Täpsus (%)**	**Kiirus (ms)**
Naive Bayes	79.70	110
SVM	81.35	14 840

Tabel 2 SVM ja Naive Bayes mudeli täpsuse ja treenimiskiiruse võrdlus filmiarvustuste andmestiku põhjal (*Movie review dataset 2.0*) [14]

Seetõttu osutus teiseks pinnapealseks mudeliks Naive Bayes, kuna see on üks lihtsamaid ja kiiremaid masinõppe mudeleid, samal ajal andes võrreldavaid tulemusi keerulisemate meetoditega [14], [15]. Naive-Bayes on samuti kombineeritud TF-IDF eeltöötlusega.

3.1.1 TF-IDFi ülevaade

TF-IDF (*term frequency – inverse document frequency*) on sõnade olulisusskoori arvutamise meetod. TF-IDF skoor leitakse iga sõna jaoks igas dokumendis. See meetod määrab ära sõnade olulisuse aidates välistada erinevates dokumentides tihtiesinevaid sõnu, mis on ebaolulised, nagu näiteks sidesõnad. [16]

Valemis 1 on välja toodud TF-IDF meetod, mis koosneb kahest komponendist, millest esimene näitab lokaalset kaalu (olulisuskaal dokumendi alusel, kus sõna esineb) ja teine globaalset kaalu (olulisuskaal arvestades sõna esinemist teistes dokumentides). Valemis 1 määratakse sõnale *t* kaal dokumendis *d*, kus $\mathrm{tf}_{\mathrm{t,d}}$ komponent tähistab sõna *t* esinemise sagedust dokumendis *d*. Sõna kaal on kõige suurem siis, kui see esineb väikeses hulgas dokumentides väga tihti. [17]

$$\mathrm{tf} - \mathrm{idf}_{\mathrm{t,d}} = \mathrm{tf}_{\mathrm{t,d}} \times \mathrm{idf}_{t} \qquad (1)$$

Valemis 2 on esitatud TF-IDF valemi idf_t komponent, mis koosneb logaritmi jagatisest, kus kogu dokumentide hulga suurus *N* on jagatud df_t ehk sõna *t* esinemise sagedusega dokumentide hulgas. [17]

$$\mathrm{idf}_t = \log\left(\frac{N}{\mathrm{df}_t}\right) \quad (2)$$

3.1.2 RoBERTa ülevaade

RoBERTa on eeltreenitud keele mudel, mis on edasiarendus BERT (*Bidirectional Encoder Representations from Transformers*) mudelist. RoBERTa mudel on üles ehitatud samale arhitektuurile nagu BERT mudel, kuid andmete eeltöötluse ja eeltreenimise osas on tehtud olulisi muudatusi. [12], [18]

BERT mudeli eeltreenimise sisendid koosnesid kahest liidetud järjestusest, kus iga järjestusest koosnes mitmest lausest, mille kombineeritud pikkus oli kuni 512 märki. RoBERTa mudeli sisendid koosnesid lihtsalt järjestustest, mis koosnesid lausetest, mille kombineeritud pikkus oli kuni 512 märki. Kui lausete vahel muutus dokument, siis lisati ka eraldi dokumente eristav märk. [12]

Nii BERT kui ka RoBERTa mudelitel on kaks variatsiooni *base* ja *large*, mis erinevad mudelite keerukuste poolest. Variatsioon *base* koosneb 110 miljonist sisemisest parameetrist, kuid variatsioon *large* koosneb 340 miljonist sisemisest parameetrist. Käesolevas töös on kasutatud *base* variatsiooni, kuna *large* variatsioon on liiga ressursinõudlik. [12]

Eeltreenimise käigus treeniti BERT mudelit kahe sammuga, kuid RoBERTa puhul tehti eeltreenimine ainult ühe sammuga. Vastavalt lõppülesande vajadustele tehti peenhäälestamine, mille käigus treeniti eeltreenitud mudel vastavalt ülesandele muutes mudeli väljundkihti. [12]

BERT mudeli eeltreeningu andmeteks olid BooksCorpus ja inglise Vikipeedia, mis olid kokku 16GB. RoBERTa *base* variatsioon kasutas sama andmestikku nagu BERT. Mudeli *large* eeltreenimise andmestik oli oluliselt suurem kasutades BERT andmestikke ja lisaks veel *CC-news*, *OpenWebText* ja *Stories* andmestikke, mis olid kokku 160GB. [12]

Eeltreeningu esimeses etapis maskeeriti treeningandmetes 15% sõnadest *[mask]* märgiga, mida hakati ennustama. Oluliseks erinevuseks BERT ja RoBERTa mudeli puhul on, et BERT mudeli puhul olid maskeeritud sõnad staatilised, mistõttu võis osa sõnade vahelisi

seoseid ära kaduda. RoBERTa puhul aga kasutati dünaamilist maskeerimist, ehk sisendandmestikku dubleeriti 10 korda, mille puhul iga tekstijärjestus oli maskeeritud 10 erinevat viisi. [12], [18]

BERT mudeli eeltreeningu teises etapis toimus järgmise lause ennustamine, mille eesmärgiks oli lausetevaheliste seoste tuvastamine. Treeningandmeteks olid A ja B järjestuste hulk, kus 50% ajast järgnes järjestusele A järjestus B ning 50% ajast oli järjestusele A lisatud juhuslik järjestus B, mis algtekstides ei järgnenud järjestusele A. RoBERTa mudeli puhul jäeti teine etapp tegemata, kuna selle lisamine vähendas lõpliku mudeli tulemusi. [12], [18]

3.1.3 XLNeti ülevaade

XLNet on eeltreenitud keele mudel, mis põhineb Transformer-XL mudelil, mis kasutab segmentide kordumise mehhanismi ja relatiivset kodeerimisskeemi, kus iga segment on fikseeritud pikkusega osa sisendteksti järjestusest. [19]

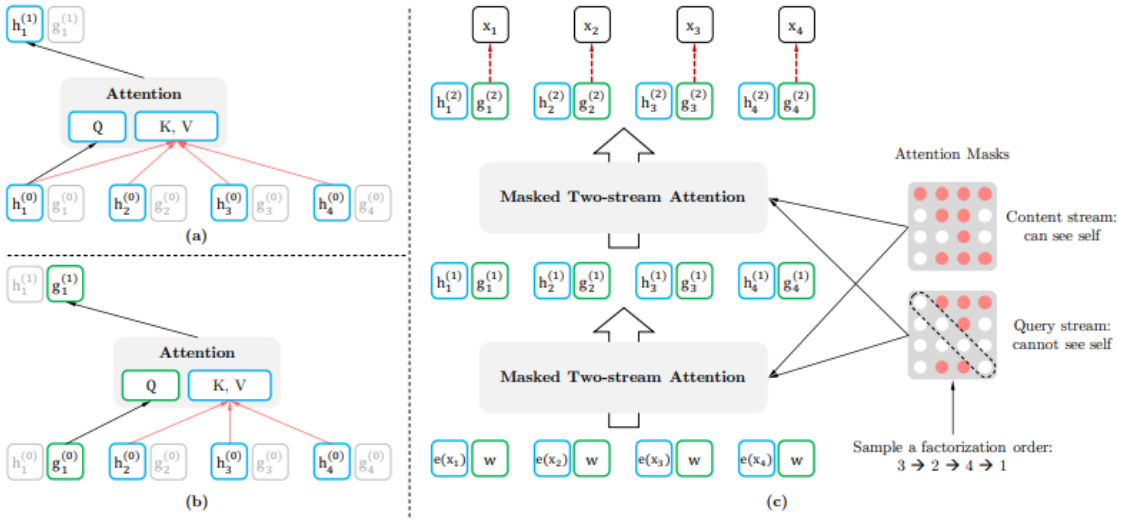
Segmentide kordumise mehhanismi põhimõtteks on talletada vahemälusse treenitud segmendi väljundit, et seda saaks kasutada laiendatud kontekstina järgmise segmendi treenimiseks. Relatiivne kodeerimisskeem on oluline selleks, et talletatud segmendi ja uue segmendi sõnade asukohad oleksid eristatavad kahe segmendi vahel. Selline lähenemine võimaldab saavutada paremaid tulemusi, eriti pikemate lausete puhul. [20]

Sarnaselt RoBERTa mudelile, on ka XLNet mudelil olemas *base* ja *large* variatsioonid, mille seast on antud töös kasutatud *base* variatsiooni. Mudeli *base* variatsiooni eeltreenimiseks on kasutatud BooksCorpus ja inglise Vikipeedia andmestikke, mis olid ka RoBERTa *base* mudeli sisendiks. [12], [19]

Peamiseks erinevuseks XLNet ja RoBERTa mudeli puhul on, et RoBERTa kasutas eeltreenimiseks maskeeritud tähiseid, kuid XLNet kasutas eeltreenimiseks permutatsioone sõnade faktoriseerimise järjekorrast. [19]

Kuna lausetes on sõnade järjekord oluline, siis oli seoste tuvastamiseks kasutatud kahe-voolulist (*two-stream*) tähelepanumehhanismi, mis koosnes kahest representatsioonist: sisu ja päringu representatsioon. Sisu *(content)* representatsioon kodeerib sõnadevahelist konteksti ja sõna ennast, kuid päringu (*query)* representatsioonil on ainult ligipääs sõna

kontekstile ja sõna asukohale. Päringu representatsiooni abil tehti lõplik sõna ennustamine antud lauses. [19]

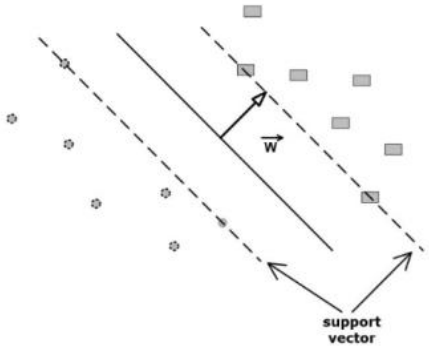


Joonis 4 XLNet mudeli arhitektuur [19]

Joonisel 4 on näidatud XLNet mudeli arhitektuuri, kus sõnajärjestusele $x_1 \ldots x_4$ on näidatud kahe-voolulist tähelepanumehhanismi (sektsioon c), kus päringu representatsioon on *g* (sektsioon b) ja sisu representatsioon on *h* (sektsioon a). [19]

3.1.4 Tugivektormasina ülevaade

Tugivektormasin (*support vector machine*, SVM) on binaarse klassifitseerimise mudel, mida kasutatakse erinevates ülesannetes nagu sõnade tuvastamine, objektide tuvastamine ja tekstide klassifitseerimine. See mudel toimib põhimõttel, kus sisendandmed $x_1 \ldots x_n$ on ette antud vektori kujul koos siltidega $y_1 \ldots y_n$, kus y kuulub hulka {-1, 1}. SVM eesmärgiks on leida hüpertasand, mis lineaarsel viisil maksimaalselt eraldab eri klasside andmeid, mis on näidatud joonisel 5. [21]



Joonis 5 SVM optimaalse hüperplaani näide [21]

Hüpertasandi ühel poolel on kõik vektorid, mis kuuluvad klassi -1 ja teisel poolel on kõik vektorid, mis kuuluvad klassi 1. Treeningandmete näited, mis asuvad hüpertasandile kõige lähemal, nimetatakse tugivektoriteks (*support vector*). [21]

Mittelineaarsete andmete puhul kasutatakse SVM puhul *kernel* funktsioone, mis viib andmed kõrgemasse funktsionaalsesse ruumi, et oleks võimalik neid lineaarselt eraldada. Üheks levinud kernel funktsiooniks on *Radial basis function* (RBF), mida on antud töös kasutatud. RBF kerneli meetod on välja toodud valemis 3. [22]

$$K(x, y) = \exp\left(-\frac{\left||x - y|\right|^2}{2\sigma^2}\right) \tag{3}$$

Valemis 3 välja toodud RBF kerneli meetodi sisendid x ja y on treeningandmete vektorid. Konstant gamma (σ) defineerib iga treeningvektori individuaalset mõju klassifikatsiooni tegemisele. Väiksem gamma väärtus muudab mudelit lineaarsemaks, kuid suurem gamma väärtus mõjutab mudelit rohkem järgima andmestiku kuju. [23], [24]

Kuna SVM mudel on binaarne klassifitseerimismudel, siis mitme klassi klassifitseerimisülesande jaoks luuakse mitu SVM hüpertasandit, kasutades erinevaid strateegiaid nagu *one-against-all* (OAA) või *one-against-one* (OAO). Antud töös on kasutatud OAO lähenemist, kus on loodud n-arvulise klasside jaoks $n(n-1)/2$ hüpertasandit, mis võrdlevad kahte klassi omavahel. Igat sisendit võrreldakse kõikide tasanditega ja igale klassile määratakse skoor vastavalt iga hüpertasandi tulemusele. Ennustusväljundiks kasutatakse klassi, mis saab kõige kõrgema skoori. [25]

3.1.5 Naive Bayesi ülevaade

Naive Bayes ehk NB algoritm on statistiline (masinõppe) klassifitseerimisalgoritm, mis põhineb Bayesi teoreemil. Selle eesmärk on leida kahe sündmuse toimumise tingimuslik tõenäosus, mis põhineb iga üksiku sündmuse toimumise tõenäosustel. [26]

Naive Bayes algoritmi variatsioonideks on multinomiaalne Naive Bayes, Bernoulli Naive Bayes, Gaussi Naive Bayes jpm. Antud töös on kasutatud multinomiaalset Naive Bayesi algoritmi. [26]

Multinomiaalne Naive Bayes (MNB) toimib sõnade esinemissageduse põhimõttel, kus korrutatakse dokumendi kõigi sõnade esinemise tõenäosused kõikide klasside vastu. Kõrgeim klassi tõenäosus näitab antud dokumendi klassi kuuluvuse. [27]

Valemis 4 on välja toodud MNB algoritm, kus on arvutatud dokumendi t_i klassi c kuuluvuse tõenäosus $Pr(c|t_i)$. Klasside hulk on tähistatud C. [27]

$$Pr(c|t_i) = \frac{Pr(c)Pr(t_i|c)}{Pr(t_i)}, c \in C \tag{4}$$

MNB valemis $Pr(c)$ komponenti on võimalik välja arvutada jagades klassi c kuuluvate dokumendi hulga suurus kogu dokumentide hulga suurusega. Komponent $\Pr(t_i|c)$ näitab t_i dokumendi tõenäosust kuuluda klassi c, mis on näidatud valemis 5. [27]

$$Pr(t_i|c) = (\sum_n \mathrm{f}_{ni})! \prod_n \frac{Pr(w_n|c)^{\mathrm{f}_{ni}}}{\mathrm{f}_{ni}!} \tag{5}$$

Valemis 5 tähistab f_{ni} komponent sõna n sagedust dokumendis t_i ja $Pr(w_n|c)$ tähistab sõna n tõenäosust kuuluda klassi c. Valemis 6 on esitatud treeningandmete põhjal ennustatud $Pr(w_n|c)$ komponent, kus N tähistab sõnade arvu dokumentide sõnastikus ja α tähistab silumiskonstanti. [27], [28]

$$\widehat{Pr}(w_n|c) = \frac{\alpha + F_{nc}}{\alpha N + \sum_{x=1}^{N} F_{xc}} \tag{6}$$

Valemis 6 tähistab F_{xc} komponent sõna x kogust kõikides treeningdokumentides, mis kuuluvad klassi c. Lisaks on valemis kasutatud silumist (konstant α), et vältida null-sageduse probleemi, kus sõna dokumendis ei esine või kui dokumendi sisu on tühi. [26], [27], [28]

MNB valemis $Pr(t_i)$ komponent on normaliseerimistegur, mille arvutuskäik on näidatud valemis 7. [27]

$$Pr(t_i) = \sum_{k=1}^{|C|} \Pr(k) \Pr(t_i|k) \tag{7}$$

Kuna valemis 5 olevad arvutuslikult keerulised komponendid $(\sum_n f_{ni})!$ ja $\prod_n f_{ni}!$ ei sõltu klassist c, siis võib need ära eemaldada. Sellisel juhul saab valemit 5 kirjutada välja lihtsamalt, mis on näidatud valemis 8. Antud valemi puhul on parameeter β konstant, mis langeb välja normaliseerimisteguri tõttu. [27]

$$Pr(t_i|c) = \beta \prod_n Pr(w_n|c)^{f_{ni}} \tag{8}$$

3.2 Tulemuste valideerimine

Mudelite analüüsimiseks on levinud meetodiks veamaatriks, mille abil on võimalik arvutada kordustäpsust, saagist ja F1-skoori. Joonisel 6 on kujutatud binaarse klassifitseerimise veamaatriksit, kus on näha tõesed ja valed ennustused. Roheline diagonaal kujutab õigesti ennustatud väärtuseid ning punane diagonaal kujutab valesti ennustatud väärtusi. [29]

		Tegelik klass: Tõene	Tegelik klass: Väär
Ennustatud klass	Tõene	Tõene positiivne (TP)	Vale positiivne (FP)
	Väär	Vale negatiivne (FN)	Tõene negatiivne (TN)

Joonis 6 Veamaatriksi näide [29]

Kui ennustus ja tegelik väärtus on tõesed, siis on tegu tõese positiivsega (TP), kuid kui ennustus ja tegelik väärtus on väärad, siis on tegu tõese negatiivsega (TN). Kui ennustus on tõene, kuid tegelik on väär, siis on tegu vale negatiivsega (FN). Kui tegelik on väär, kuid ennustus on tõene, siis on tegu vale positiivsega (FP). [29]

Kordustäpsus (PPV – *positive predictive value* ehk *precision*, valem 9) on mõõdik, mis näitab õigesti klassifitseeritud positiivsete väärtuste (TP) osakaalu kõikide positiivselt ennustatud väärtuste (FP + TP) suhtes. Mõõdiku skaala on 0–1 ning kõrgemad väärtused näitavad suuremat täpsust. [29]

$$PPV = \frac{TP}{FP + TP} \tag{9}$$

Saagis (TPR – *true positive rate* ehk *recall*, valem 10) näitab õigesti klassifitseeritud positiivsete väärtuste osakaalu õigesti klassifitseeritud positiivsete väärtuste ja vale negatiivsete väärtuste suhtes. Mõõdiku skaala on 0–1 ning kõrgemad väärtused näitavad suuremat saagist. [29]

$$TPR = \frac{TP}{FN + TP} \tag{10}$$

F1-skoor (*F1-score*, valem 11) näitab harmoonilist keskmist täpsuse ja saagise suhtes. Mõõdiku skaala on 0–1 ning kõrgemad väärtused näitavad paremat mudeli klassifitseerimise edukust. [29]

$$F1 - skoor = \frac{2 * PPV * TPR}{PPV + TPR} = \frac{2 * TP}{2TP + FP + FN} \tag{11}$$

Täpsus (ACC – *accuracy*, valem 12) on mõeldud üldiseks mudeli edukuse hindamiseks, mis näitab õigesti klassifitseeritud väärtuste osakaalu kõigi väärtuste suhtes. Mõõdiku skaala on 0–1 ning kõrgemad väärtused näitavad paremat mudeli täpsust. [29]

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Joonisel 7 on kujutatud mitme klassi veamaatriksit, kus on binaarsete klasside asemel kolm klassi A, B ja C. Antud joonisel on korrektsed ennustused esitatud viisil TP_x, kus x tähistab klassi. Väärad ennustused on esitatud viisil E_{xy}, kus x tähistab tegelikku klassi ja y tähistab ennustatud klassi.

		Tegelik klass		
		A	B	C
Ennustatud klass	A	TP_A	E_{BA}	E_{CA}
	B	E_{AB}	TP_B	E_{CB}
	C	E_{AC}	E_{BC}	TP_C

Joonis 7 Mitme klassi veamaatriksi näide [29]

Erinevalt kahe klassi veamaatriksist, arvutatakse vale positiivsete ja vale negatiivsete hulk klasside põhiselt. Näiteks A klassi valepositiivsed on arvutatud valemiga $FP_A = E_{BA} + E_{CA}$, kus E_{BA} tähistab ennustusi klassi A, kuid tegelik on klass B ja E_{CA} tähistab ennustusi klassi A, kuid tegelik on klass C. Sarnaselt arvutatakse ka valenegatiivsed

klassile A valemiga $FN_A = E_{AB} + E_{AC}$, kus E_{AB} tähistab ennustusi klassi B, kuid tegelik on klass A ning E_{AC} tähistab ennustusi klassi C, kuid tegelik on klass A. [29]

Kordustäpsuse, saagise ja f1-skoori arvutamine käib samuti igale klassile eraldi. Selleks, et välja selgitada mudeli üldine kordustäpsus, saagis ja f1-skoor, on võimalik kasutada makro- ja kaalutud keskmise skoore. [30]

Valemites 13 ja 14 on näidatud makro- ja kaalutud keskmise skoori arvutuskäik, kus q on klasside arv, TPM_c on antud klassi valideerimismõõdik (f1-skoor, saagis või täpsus) ja S_c on klassi c dokumentide arv. Makro- ja kaalutud keskmise meetodite oluliseks erinevuseks on, et kaalutud keskmise meetod võtab arvesse klasside kaalusid mõõdiku arvutamisel. See on oluline balansseerimata andmestiku puhul, kus mõnes klassis on rohkem dokumente kui teistes. [30]

$$TPM_{makro} = \frac{1}{q}\sum_{c=0}^{q-1} \mathrm{TPM}_c \tag{13}$$

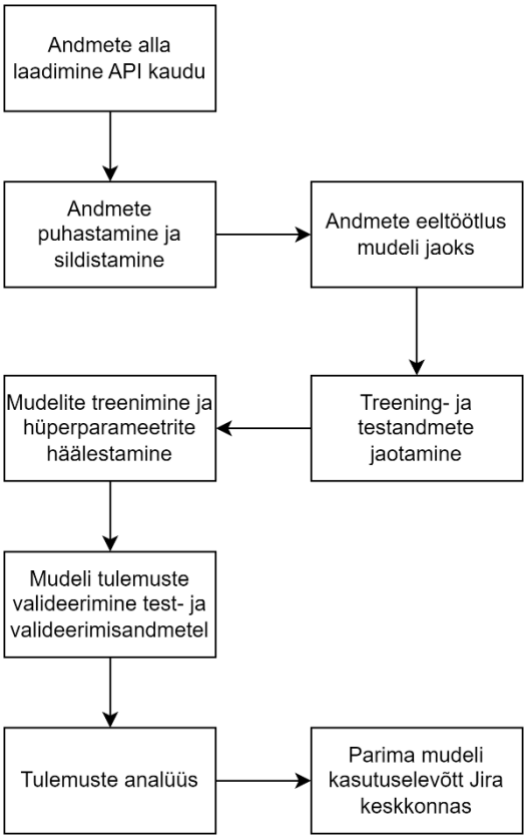
$$TPM_{kaalutud} = \frac{\sum_{c=0}^{q-1} \mathrm{TPM}_c \mathrm{S}_c}{\sum_{c=0}^{q-1} \mathrm{S}_c} \tag{14}$$

4 Realisatsioon

Käesolevas peatükis on kirjeldatud mudelite treenimise protsessi ning nende tehnilist realisatsiooni. Lisaks on välja toodud Jira integratsioon kui ka seda toetava taustteenuse tehniline kirjeldus.

4.1 Töö tegevuskäik

Joonisel 8 on esitatud töö tegevuskäigu diagramm, mis algab andmete alla laadimisest *Jira* keskkonnast. Seejärel toimub probleemtekstide filtreerimine, puhastamine ebavajalikest sõnadest ja tähemärkidest ning toimub andmete sildistamine vastavasse arendusmeeskonda.



Joonis 8 Töö tegevuskäigu diagramm

Puhastatud tekstidele toimub eeltöötlus vastavalt mudelile, mille järel jaotatakse andmed treening- ning testandmeteks. Seejärel treenitakse mudeleid ning tehakse hüperparameetrite otsing. Treenitud mudelitele tehakse valideerimine test- ja valideerimisandmestikel, mille põhjal tehakse tulemuste analüüs.

Pärast parima mudeli leidmist, toimub selle implementeerimine Jira platvormis. See tähendab pistikprogrammi arendamist Jira platvormi jaoks ning seda toetava taustteenuse arendamist, mis kasutab treenitud mudelit klassifitseerimise teostamiseks.

4.2 Andmete eeltöötlus

Andmestiku allalaadimiseks loodi eraldi skript, mille abil sai automaatselt alla laadida tootetoe probleeme läbi Jira API liidese. Skript on käivitatav käsureal ja vajab sisendparameetritena kasutajanime, API võtit, probleemi eesliidet, Jira domeeni (ettevõtte põhine domeen) ja probleemide arvulist vahemikku.

Probleemid on talletatud JSON failide kujul, kus on kirjas probleemi ajalugu, pealkiri ja tekst. Kuna need võivad sisaldada sensitiivseid andmeid, siis on turvalisuse tagamiseks kõik probleemifailid krüpteeritud. Selle jaoks nõuab skript lisaks eelnevalt nimetatud sisendile ka Fernet võtit, mis on genereeritav Fernet[1] teegiga. Sisemiselt kasutab Fernet teek AES-128 krüpteerimist. [31]

Andmestiku eeltöötluse jaoks loodi mitmeid reegleid, mille alusel välja filtreerida mittesobivaid probleeme. Järgnevalt on välja toodud need reeglid koos põhjendustega:

- Staatus ei ole "Tehtud" – Lahendamata tootetoe probleeme võidakse veel ümber klassifitseerida teise tiimi, mistõttu ei olnud mõistlik neid treenimiseks kasutada.
- Otsus on "Ei Tee" või "Ei Paranda" – Sellist tüüpi probleemid olid enamasti sellise otsuse saanud juba tootetoe agentide poolt (mitte arendustiimilt), mistõttu olid need enamasti klassifitseeritud valesse tiimi. Seega ei olnud sobilik neid kasutada.
- Lahendustiimiks on "Tootetugi" – See tiim on üldine lahendustiim, kuhu kuuluvad uued klassifitseerimist ootavad tootetoe probleemid. Kuna sellese gruppi kuuluvad erinevate arendustiimide probleemid, siis ei saanud selle põhjal mudelit treenida.

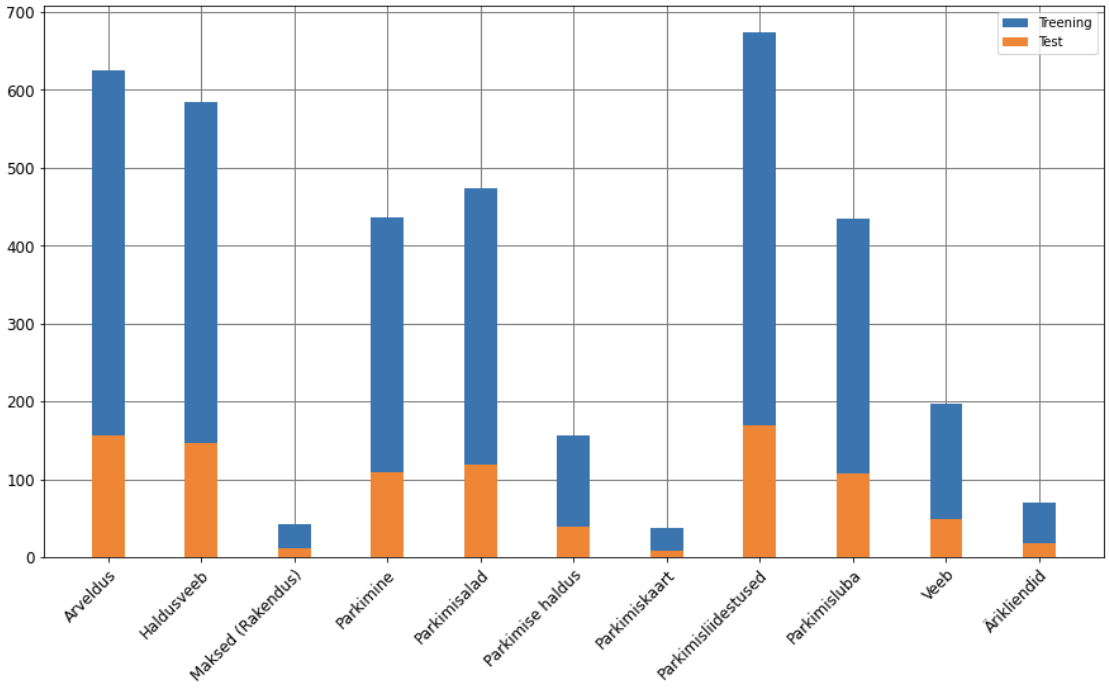
[1] https://cryptography.io/en/38.0.1/fernet/

- Lahendustiimiks on "Kasutaja tiim" ja probleemi otsus on "Duplikaat" – Sellised probleemid ei olnud seotud antud arendustiimiga, mistõttu olid need mudeli treenimiseks madala kvaliteediga. Selliseid probleeme oli kokku viis.

- Lahendustiimiks on "Parkimine" või "Arveldus" ja probleem on vanem kui Jaanuar 2021 – Need tiimid olid varasemalt sama lahendustiimi all, mistõttu olid vanemad probleemid klassifitseeritud segamini tiimide "Parkimine" või "Arveldus" alla.

- Lahendustiimi ajalugu on väiksem kui 25 probleemi – Kuna alla 25 probleemi suuruse andmestiku põhjalt oli keeruline mudelitel luua seoseid, siis filtreeriti sellised lahendustiimid välja. Selliseid tiime oli kokku 6 ja nende tiimide keskmine probleemide arv oli 9. Kuna selline filtreerimine võib näilikult tõsta mudelite efektiivsust, siis on peatükis 5.1 välja toodud mudelite võrdlus nii filtreeritud probleemide puhul kui ka täiesti uute probleemide puhul, kus ei ole selle reegliga filtreerimist tehtud.

- Lahendustiimiks on "PSM" – Selle tiimi probleemid ei vaja automaatset klassifitseerimist, kuna tegu ei ole arendustiimiga ja silt lisatakse juba probleemi loomisel.



Joonis 9 Treening- ja testandmete jaotus arendustiimide põhiselt

Mudeli sisendi jaoks loodi treening- ja testandmed jaotusega 80/20. Andmete jaotused tehti iga klassi kohta eraldi. Joonisel 9 on esitatud treening- ja testandmete hulgad arendustiimide lõikes.

Jooniselt 9 on näha, et sisendandmed olid balansseerimata, kuna kolmel arendustiimil oli oluliselt vähem andmeid kui teistel klassidel. Nendeks tiimideks olid ärikliendid, maksed (rakendus) ja parkimiskaart. Lisaks oli võrdlemisi vähe andmeid ka parkimise halduse ning veebi tiimidel. Kõige rohkem andmeid on tiimidel haldusveeb, arveldus ja parkimisliidestused. Filtreeritud probleeme kokku oli 4665 ning arvuline probleemide jaotus arendustiimide lõikes on näidatud tabelis 3.

Arendustiim	**Probleemide arv**
Arveldus	781
Haldusveeb	730
Maksed (Rakendus)	53
Parkimine	546
Parkimisalad	593
Parkimise haldus	195
Parkimiskaart	47
Parkimisliidestused	843
Parkimisluba	542
Veeb	246
Ärikliendid	89

Tabel 3 Probleemide jaotus arendustiimide lõikes

Eeltöötluse üheks osaks oli ka tekstide puhastamine ebavajalikest tähemärkidest, mille jaoks kasutati clean-text[1] teeki. See teek eemaldas automaatselt veebilingid, numbrid ja e-mailid, mis olid klassifitseerimiseks ebaolulised.

Naive Bayes ja SVM meetodite puhul toimus ka stopp-sõnade eemaldamine ja sõnade lemmatiseerimine, et luua võimalikult sobiv sisend. Stopp-sõnade eemaldamine tehti nltk[2] teegi abil, mis võimaldas automatiseerida stoppsõnade loendi alla laadimist ja

[1] https://pypi.org/project/clean-text/

[2] https://www.nltk.org/index.html

kasutamist. Lemmatiseerimine, ehk sõnade algvormi viimine, oli samuti tehtud kasutades nltk teeki. Kuna lemmatiseerimise jaoks oli vaja teada sõna konteksti, siis selle jaoks oli kasutatud WordNet'i. See on leksikaalne inglise keele andmebaas, kus nimisõnad, tegusõnad, omadussõnad ja määrsõnad on rühmitatud kognitiivsete sünonüümide hulkadesse ehk sünohulkadesse, millest iga rühm väljendab oma mõistet. Sünohulgad on omavahel seotud läbi erinevate semantiliste ja leksikaalsete suhete. [32]

4.3 Klassifitseerimismudelite realisatsioon

Käesolevas töös kasutati mudelite realisatsiooniks ja eeltöötluseks erinevaid teeke sõltuvalt klassfitseerimismudeli tüübist. Kõik mudelid olid loodud Jupyter töövihikutes, mis võimaldas mudeleid kasutada erinevates keskkondades.

SVM ja Naive Bayes mudelite loomiseks kasutati scikit-learn[1] teeki, mis sisaldas nende mudelite implementatsioone. See teek lihtsustas mudelite kasutamist, kuna selle abil oli võimalik kiiresti luua baaslahendus ning võimaldas muuta vastavate mudelite sisendparameetreid. Nende mudelite TF-IDF eeltöötlus tehti samuti scikit-learn teegi abil.

Nii SVM kui ka Naive Bayes mudelite optimaalsete hüperparameetrite leidmiseks kasutati *Grid-search* meetodit. Selle meetodi tööpõhimõtteks on läbi proovida kõik sisendparameetrite kombinatsioonid, et leida kõige parema täpsusega mudel. Otsingu eesmärgiks oli leida kõige kõrgema täpsusskooriga mudel testandmetel. [33]

XLNet ja RoBERTa mudelite laadimiseks ja treenimiseks kasutati Transformers[2] teeki, mis automatiseeris eeltreenitud mudelite alla laadimist ja treenimist. Lisaks pakkus see teek ka Trainer moodulit, mille abil oli võimalik teostada mudelite treenimist. See moodul võimaldas automaatselt sünkroniseerida mudelite treenimise väljundit W&B[3] keskkonda. Lisaks oli selle mooduli abil võimalik näha mudeli täpsust testandmetel igal epohhi sammul, mistõttu sai kiiresti ülevaate, kuidas mudel suudab õppida vastavalt etteantud parameetritega.

[1] https://scikit-learn.org/stable/

[2] https://huggingface.co/docs/transformers/main/en/index

[3] https://wandb.ai/site

Hüperparameetrite leidmiseks sügavate meetodite puhul kasutati W&B poolt pakutud teeki, mis võimaldas defineerida mudelitele parameetrid, et leida kõige paremini toimiv mudel. Hüperparameetrite otsingu eesmärgiks oli leida kõige väiksema kaoga mudel. Otsingu viisiks kasutati Bayes meetodit, kuna traditsionaalne *grid-search* meetod ei olnud optimaalne, arvestades et sügavatel meetoditel kulub tavaliselt treenimiseks oluliselt kauem aega. [33]

Mudelite hindamiseks kasutati scikit-learn teeki, mis võimaldas genereerida klassifitseerimisraporteid. Raportite haldamiseks kasutati W&B keskkonda, mis võimaldas keskselt vaadata ning võrrelda erinevate mudelite treenimise tulemusi. Pärast iga mudeli treenimissessiooni laeti info mudelite kohta W&B keskkonda. XLNet ja RoBERTa mudelite puhul oli võimalik näha ka erinevate parameetrite mõju mudelite edukusele. Lisas 4 on välja toodud W&B keskkonna kuvatõmmised mudeli treenimisprotsessist.

Mudelite treenimise jaoks kasutati Amazon SageMaker Studio[1] keskkonda, mis pakub Jupyter töövihikute pilveplatvormi koos erinevate masinatega. Kuna sügavad meetodid nõudsid suurel hulgal videokaardi mälu, siis kasutati nende mudelite treenimiseks Amazon EC2 g5.xlarge masinaid. Need masinad kasutavad Nvidia A10G videokaarti, millel on 24GB videomälu. Pinnapealsete meetodite puhul kasutati samu masinaid, kuid treenimine oli tehtud teise generatsiooni AMD EPYC protsessoriga. [34]

Tabelis 4 on välja toodud mudelite optimaalsed hüperparameetrid, mis olid leitud parameetrite otsingu abil. Lisaks on välja toodud ka muud parameetrid, mis olid mudelite treenimisel olulised. Mõistlike parameetrite vahemike leidmiseks kasutati mudelite autorite soovitusi ning tehti ka eksperimenteerimisi.

[1] https://aws.amazon.com/sagemaker/

Hüperparameeter	**Optimaalne väärtus**	**Parameetri vahemik / selgitus**
XLNet		
Epohhide arv	5	4, 5, 6
Learning rate	0.00003	0.00002, 0.00003, 0.00005
Adam epsilon	0.000001	Väärtus vastavalt mudeli autorite soovitustele
Ploki suurus treenimisel	18	Maksimaalne väärtus, limiteeritud videokaardi mälust
Ploki suurus valideerimisel	56	Maksimaalne väärtus, limiteeritud videokaardi mälust
Maksimaalne sisendi pikkus	256	Maksimaalne väärtus, limiteeritud videokaardi mälust
RoBERTa		
Epohhide arv	7	5, 6, 7
Learning rate	0.00002	0.00001, 0.00002, 0.00003
Adam epsilon	0.000001	Väärtus vastavalt mudeli autorite soovitustele
Ploki suurus treenimisel	18	Maksimaalne väärtus, limiteeritud videokaardi mälust
Ploki suurus valideerimisel	56	Maksimaalne väärtus, limiteeritud videokaardi mälust
Maksimaalne sisendi pikkus	256	Maksimaalne väärtus, limiteeritud videokaardi mälust
SVM		
C	20	1, 10, 15, 20, 25
Gamma	0.05	0.04, 0.05, 0.1, 0.15
Naive Bayes		
Alpha	0.006	0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01

Tabel 4 Mudelite hüperparameetrid

4.4 Taustteenuse kirjeldus

Kasutades parimat treenitud mudelit, loodi taustteenus, mis klassifitseerib uusi teenusetoe probleeme vastavalt nende sisule. Taustteenus on kirjutatud Python keeles ja kasutab Flask[1] raamistikku, mis on mõeldud veebiteenuste arendamiseks. [35]

Taustteenus võtab sisendina Jira probleemi numbri ja laeb sisemiselt alla probleemi sisu kasutades teenuses talletatud Jira API võtit. Probleemi sisu antakse edasi klassifikaatorile, mis väljastab tõenäosused tiimide kaupa, kuhu see probleem võiks kuuluda. Klassifikaatori tõenäosuste alusel tagastab teenus tulemuse JSON-kujul. Joonisel 10 on esitatud päringu ja vastuse kuju.

```
Päring:

POST /classifier
Api_key: XXX

{
    "task_tag": "Jira probleemi number, nt PPT-1000"
}

Vastus:

{
    "classification": "Klassifikatsiooni tulemus, nt WEB (96%)"
    "raw_classifications": [
        {"team": "WEB", "probability": 0.96}
    ]
}
```

Joonis 10 Taustteenuse päringu ja vastuse näide

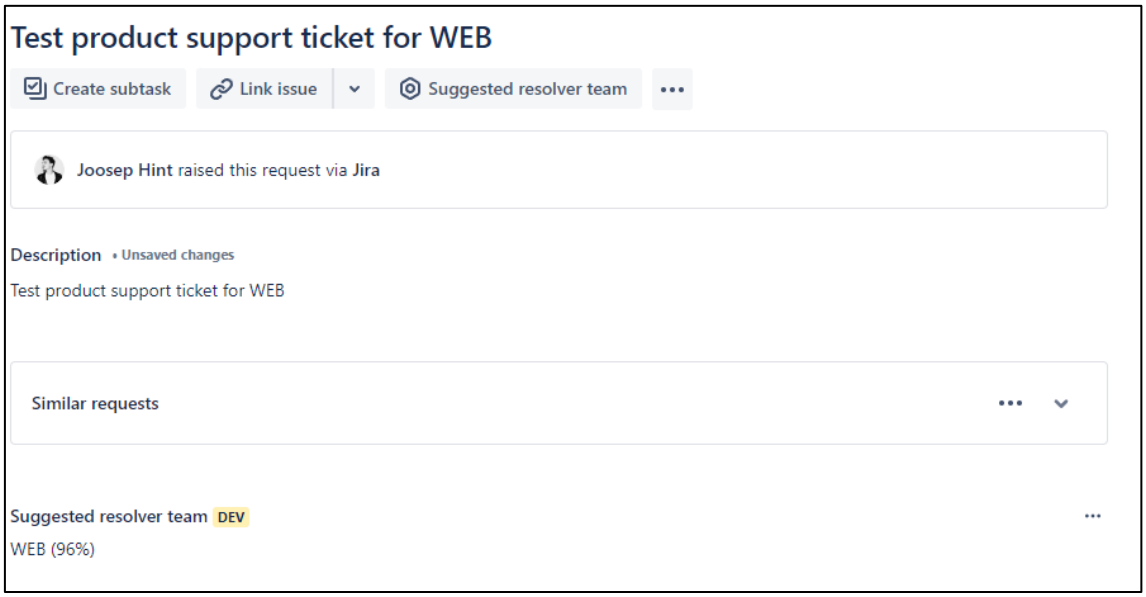
Antud taustteenus toimib selliselt, et ühtegi teenusetoe probleemi ei võeta sisendiks ega ka kuvata väljundis, et vältida võimalikke andmete lekkimisi. Lisaks on ka sellel teenusel API võtmete valideerimine, mille alusel saab teenuse poole pöördumisi teha. Lubatud võtmed on defineeritud konfiguratsiooni failis. Need API võtmed ei ole Jira API võtmetega seotud ja on mõeldud turvalisuse tagamiseks, et välised osapooled ei saaks teenust pahatahtlikult kasutada.

[1] https://flask.palletsprojects.com/en/2.2.x/

4.5 Jira integratsioon

Jira integratsiooni jaoks kasutati Atlassiani Forge[1] lahendust, mis on JavaScript keelel põhinev NPM moodul[2]. Forge poolt pakutakse palju erinevaid standard-komponente, mida on võimalik valmislahendusena kasutada.

Töö lahenduse puhul kasutati IssuePanel[3] komponenti, mis ilmub eraldi alamsektsioonis vajutades „Suggested resolver team“ nupule. Joonisel 11 on näidatud, kuidas see komponent avatuna välja näeb.



Joonis 11 Jira integratsiooni näide

Sisemiselt teeb see pistikprogramm taustteenuse poole pöördumise vastavalt eelmises peatükis nimetatud API kirjeldusele ning kuvab komponendi sisuna *classification* välja. Seda välja programm ise ei töötle ega muuda, mis annab võimaluse teha taustteenuses klassifitseerimise muudatusi pistikprogrammi muutmata.

[1] https://developer.atlassian.com/platform/forge

[2] https://www.npmjs.com/package/@forge/cli

[3] https://developer.atlassian.com/platform/forge/ui-kit-components/jira/issue-panel/

Forge lahendus pakub palju käsurea käske, mille abil on võimalik pistikprogrammi koodi luua, pakendada ja Jira keskkonda üles laadida. Tabelis 5 on välja toodud erinevad käsud ja nende kirjeldused. Käsk nr 1 on vajalik esmaseks projekti üles seadmiseks. Käsk nr 2 on vajalik koodimuudatuste üles laadimiseks Jira platvormile. Käsk nr 3 on vajalik muudatuste testimiseks mõnes keskkonnas (peamiselt testkeskkonnas). [36]

Nr	**Käsk**	**Selgitus**
1	`forge create`	Loob projekti failid ja baasstruktuuri. Lisaks tekitab ka baaskoodi valitud UI komponendile
2	`forge deploy`	Pakendab, kompileerib ja laeb pistikprogrammi koodi üles Jira platvormile
3	`forge install`	Installeerib pistikprogrammi valitud keskkonda

Tabel 5 Forge käskude kirjeldused [36]

Käesoleva töö raames oli Jira pistikprogramm tehtud soovitusi pakkuva lahendusena. Teisisõnu see programm ei jaota uusi probleeme automaatselt, vaid soovitab tootetoe agentidele, millisesse tiimi antud probleem kuulub.

5 Analüüs

Käesolevas peatükis on välja toodud mudelite tulemused ning tehtud nende põhjal analüüs. Lisaks vastatakse püstitatud uurimistöö küsimustele ning tuuakse välja ka olulised piirangud ja edasised arengud.

5.1 Klassifitseerimismudelite võrdlus

Mudelite võrdluseks on kasutatud kahte erinevat andmestikku, millest esimene on testandmestik, mis tulenes 80/20 andmete jaotusest. See andmestik sisaldas väheoluliste klasside filtreerimist, mistõttu võisid sellest tulenevad tulemused olla kõrgendatud.

Tabelis 6 on välja toodud mudelite kordustäpsus, saagis ja F1-skoor makro keskmise alusel testandmestikul. Tabelist on näha, et kõige kõrgema kordustäpsuse skoori sai Naive Bayes mudel (0.76), kuid samas selle mudeli saagise tulemus oli kõige madalam (0.59), mistõttu oli ka F1-skoor madalaim (0.62). Kõige kõrgema F1-skoori sai RoBERTa mudel (0.70). Väga lähedal oli ka XLNet, kuid selle F1-skoor (0.69) oli madalam saagise tõttu.

Mudel	**Kordustäpsus**	**Saagis**	**F1-skoor**
Naive Bayes	**0.76**	0.59	0.62
SVM	0.69	0.66	0.67
XLNet	0.72	0.68	0.69
RoBERTa	0.72	**0.71**	**0.70**

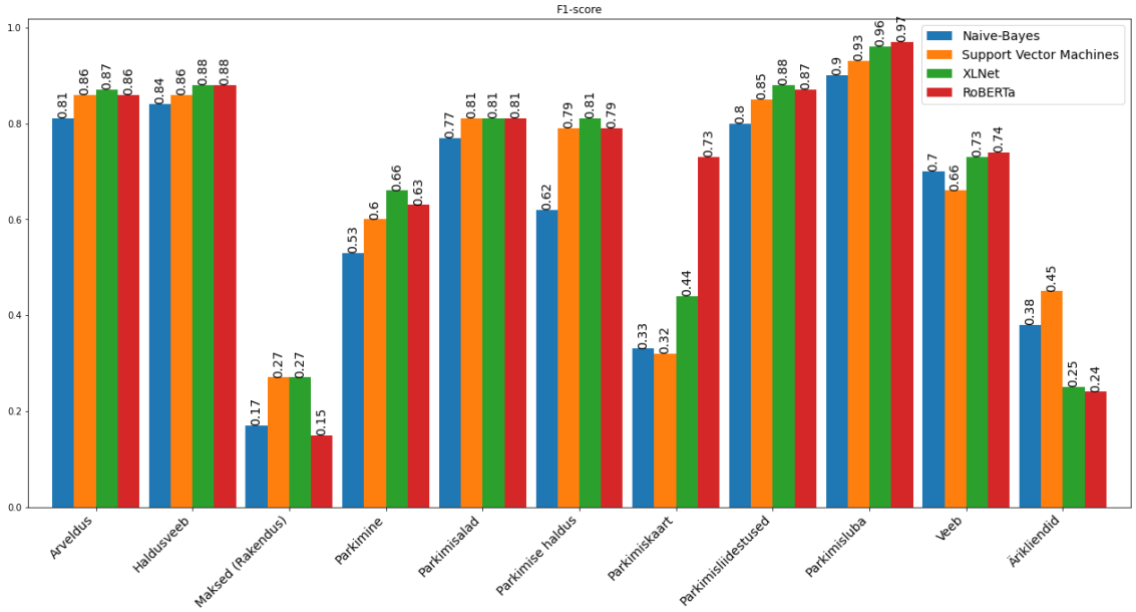
Tabel 6 Mudelite kordustäpsus, saagis ja F1-skoor makro keskmise alusel testandmestikul

Tabelis 7 on esitatud mudelite kordustäpsus, saagis ja F1-skoor kaalutud keskmise alusel testandmestikul. Seal on näha, et kõige parema kordustäpsuse ja saagise said RoBERTa ning XLNet mudelid (0.82) ning nende F1-skoor oli peaaegu identne (erinevus 0.01). SVM mudel jäi neile marginaalselt F1-skooriga alla (0.80), kuid kõige madalama F1-skooriga oli NB (0.75).

Mudel	Kordustäpsus	Saagis	F1-skoor
Naive Bayes	0.76	0.76	0.75
SVM	0.80	0.80	0.80
XLNet	**0.82**	**0.82**	**0.82**
RoBERTa	**0.82**	**0.82**	0.81

Tabel 7 Mudelite kordustäpsus, saagis ja F1-skoor kaalutud keskmise alusel testandmesikul

Joonisel 12 on näidatud erinevate mudelite F1-skoorid arendusmeeskondade lõikes. Jooniselt on näha, et üle tiimide on mudelite klassifitseerimisedukusel suur varieeruvus. Kolme tiimi puhul olid F1-skoorid oluliselt madalamad, milleks olid maksed (rakendus), parkimiskaart ja ärikliendid. Kõige kõrgemaid skoore sai tiim parkimisluba.



Joonis 12 Mudelite F1-skoorid arendusmeeskondade lõikes testandmestikul

Maksed (Rakendus) tiimi puhul oli skoor kõrgeim SVM ja XLNet mudelitel (0.27), seejärel Naive Bayes (0.17) ning kõige madalama skooriga oli RoBERTa (0.15). Seevastu meeskonna parkimiskaart puhul andis teistest mudelitest oluliselt kõrgema skoori RoBERTa mudel (0.73). Sellele lähim mudel oli XLNet (0.44). Tiimi ärikliendid puhul said nii XLNet (0.25) kui ka RoBERTa (0.24) madalaima skoorid. Kõrgeim skoor oli SVM mudelil (0.45) ja sellele lähedal oli NB (0.38).

Tiimidel parkimine ning parkimise haldus oli selgelt näha, et NB ennustusskoorid olid võrreldes teiste mudelitega oluliselt madalamad. Parkimise tiimi puhul oli NB skoor 0.53, kuid teistel mudelitel olid skoorid vahemikus 0.60 - 0.66. Parkimise halduse tiimil oli vahe suurem – NB sai skooriks 0.62, kuid teiste mudelite skoorid olid vahemikus 0.79 - 0.81.

Teise andmestiku, mis oli mõeldud täiendava analüüsi jaoks, sisendiks olid tootetoe uued probleemid, mis tekkisid pärast mudelite koostamist. Selle andmestiku puhul polnud väheoluliste meeskondade filtreerimist tehtud, mistõttu näitasid mõõdikud reaalelulisemaid tulemusi.

Lisaks oli selle andmestiku juures ka kaheksa arendusmeeskonda, mida algses filtreeritud andmestikus polnud. Nendest tiimidest neli oli uued tiimid, mida varem ei eksisteerinud ja ülejäänud neli tiimi olid väheolulisuse kriteeriumi alusel testandmestikust välja jäänud. Kuna uued valideerimisandmestiku arendustiimid ei olnud täiendava mudelite analüüsi mõttes olulised, siis on need nimetatud lihtsalt kujul „Uus tiim #X“.

Tabelis 8 on välja toodud mudelite makro keskmine võrdlus valideerimisandmestikul. Tabelist on näha, et valideerimisandmestikul on mudelitel oluliselt väiksem makro keskmine võrreldes testandmestikuga. See on peamiselt tingitud sellest, et testandmestikul lisandus juurde kaheksa meeskonda, mille jaoks mudelid ei olnud treenitud. Kordustäpsused on mudelitel väga sarnased, kuid kõige suurem kordustäpsus oli mudelitel SVM ja NB (mõlemad 0.39). Saagis oli kõige kõrgem RoBERTa ja SVM mudelitel (mõlemad 0.39). F1-skoor oli samuti kõige kõrgem RoBERTa ja SVM mudelitel (mõlemad 0.38). Kõige madalama F1-skooriga oli NB (0.31).

Mudel	**Kordustäpsus**	**Saagis**	**F1-skoor**
Naive Bayes	**0.39**	0.31	0.31
SVM	**0.39**	**0.39**	**0.38**
XLNet	0.37	0.36	0.34
RoBERTa	0.38	**0.39**	**0.38**

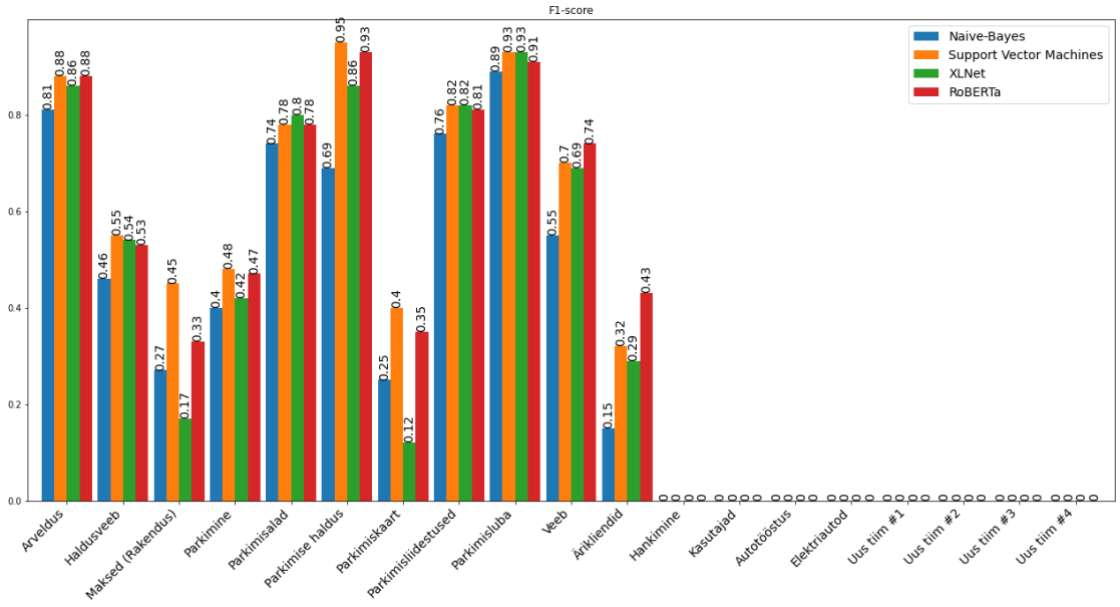
Tabel 8 Mudelite kordustäpsus, saagis ja F1-skoor makro keskmise alusel valideerimisandmestikul

Tabelis 9 on esitatud mudelite võrdlus kaalutud keskmise alusel valideerimisandmestikul. Tabelist on näha, et mudelid SVM ja RoBERTa andsid identseid tulemusi (F1-skoor 0.73). XLNet mudel (F1-skoor 0.71) jäi marginaalselt alla SVM ja RoBERTa mudelitele. Võrreldes teiste mudelitega sai NB mudel kõige madalamad tulemused (F1-skoor 0.65). Võrreldes testandmestikuga, said kõik mudelid valideerimisandmestikul oluliselt madalamaid skoore.

Mudel	**Kordustäpsus**	**Saagis**	**F1-skoor**
Naive Bayes	0.68	0.67	0.65
SVM	**0.73**	**0.74**	**0.73**
XLNet	0.72	0.72	0.71
RoBERTa	**0.73**	**0.74**	**0.73**

Tabel 9 Mudelite kordustäpsus, saagis ja F1-skoor kaalutud keskmise alusel valideerimisandmestikul

Joonisel 13 on näidatud mudelite F1-skoorid arendusmeeskondade lõikes valideerimisandmestikul. Antud joonisel said kaheksa tiimi null-skoorid, kuna mudelid ei olnud treenitud nendesse tiimidesse klassifitseerimiseks.



Joonis 13 Mudelite F1-skoorid arendusmeeskondade lõikes valideerimisandmestikul

Tiimi haldusveeb osas toimus oluline langus – testandmestikul oli keskmiseks F1-skooriks 0.87, kuid valideerimisandmestikul on keskmiseks F1-skooriks 0.52. Lisaks on näha, et kolm mudelit andsid sarnaseid tulemusi, kuid erandiks on Naive Bayes (0.46).

Tiimid maksed (rakendus), parkimiskaart ja ärikliendid on madalate skooridega nii test- kui ka valideerimisandmestikul. Arvestades, et antud tiimide puhul on treeningandmeid ja valideerimisandmeid vähe, siis suure tõenäosusega pole mudelid suutnud leida eristuvat sisu probleemidele. Selle uurimiseks on tehtud edasist analüüsi peatükis 5.3 tiimide maksed (rakendus) ja ärikliendid põhjal.

Sarnaselt testandmestikule, said valideerimisandmestikul mudelid kõrged skoorid tiimides arveldus, parkimise haldus ja parkimisluba. Oluline erand on NB mudel parkimise halduse tiimi puhul, kuna selle mudeli skoor 0.69 oli oluliselt madalam teiste mudelite keskmisest skoorist 0.91. Tiim parkimisluba on ainus, mis sai keskmise F1-skoori üle 0.9 mõlema andmestiku puhul.

Tabelis 10 on esitatud hüperparameetrite otsingul põhinev treenimisaeg mudelite lõikes. Kuigi täpsema võrdluse jaoks oleks vaja teha mitu erinevat treenimist, et leida keskmine treenimiseks kulunud aeg, saab selle tabeli põhjal siiski aru treeningaja erinevuse skaalast. Antud tabelist on näha, et Naive Bayes mudeli treenimiseks oli ajakulu nullilähedane, kuid XLNet mudeli parameetrite otsinguks kulus kõige kauem aega.

Mudel	**Naive Bayes**	**SVM**	**XLNet**	**RoBERTa**
Treenimise aeg	2s	7m 30s	1h 47m	1h 17m

Tabel 10 Treenimisaeg mudelite lõikes

Mudelite võrdluses oli kõige madalamate skooridega eristuvaks mudeliks Naive Bayes. Arvestades, et see mudel oli kõige lihtsamini treenitav, oli see siiski hea mudel, mida sai võtta baasjooneks, et näha, kuidas keerulisemad mudelid toimivad etteantud andmestikel. Lisaks oli see ka hea mudel eksperimenteerimise faasis, kuna see mudel oli arvutuslikult kõige kiirem, mistõttu oli tagasisidetsükkel oluliselt lühem.

Selgelt eristuvat parimat mudelit ei tekkinud, kuna XLNet, RoBERTa ja SVM mudelid andsid mõlema andmestiku puhul sarnaseid tulemusi. XLNet mudel jäi RoBERTa ja SVM mudelitele marginaalselt alla valideerimisandmestikul, kuid samas oli see võrreldes SVM mudeliga parem testandmestikul. Kuna XLNet mudel oli treenimisel ajakulu mõttes kõige mahukam ja ei andnud teistest mudelitest paremaid tulemusi, siis ei olnud see hea mudel antud ülesande jaoks.

Selle põhjal saab järeldada, et kõige paremad mudelid antud ülesande jaoks olid RoBERTa ja SVM. Kuigi RoBERTa treenimiseks kulus rohkem aega, on edasise analüüsi jaoks kasutatud seda mudelit, kuna see oli testandmestiku puhul SVM mudelist kõrgemate skooridega ja analüüsi teostamiseks ei olnud edasist mudeli treenimist vaja teha.

Lisades 2 ja 3 on välja toodud mudelite kordustäpsus ning saagis tiimide lõikes test- ja valideerimisandmestikul.

5.2 Parima mudeli võrdlus inimesega

Selleks, et saaks võrrelda parimat mudelit inimesega, on oluline defineerida seda, mille alusel inimesed klassifitseerimise mõttes vigu teevad. Antud töö raames on inimeste poolsed klassifitseerimise vead defineeritud selliselt, et iga probleem, mida on vähemalt ühe korra ümber suunatud teise arendustiimi, tähendab klassifitseerimise mõttes viga.

Ümbersuunamiste leidmiseks kasutati Jira probleemidel kaasas olnud ajalugu, kus oli ära näidatud, millistesse tiimidesse antud probleem on varasemalt kuulunud. Selle ajaloo põhjal arvutati ümbersuunamiste arv. Olulise erandina jäeti välja probleemide suunamine tootetoe agentide tiimi, kuna selle tiimi eesmärk on probleeme edasi suunata arendusmeeskondadesse.

Tabelis 11 on välja toodud inimeste poolt tehtud arendustiimide ümbersuunamised testandmestikul. Antud tabelis oli probleeme kokku 933, kuid nendest vähemalt ühe korra suunati ümber 124 probleemi. Selle alusel saab tuletada veamäära, milleks on 13%.

Ümbersuunamiste arv	**0**	**>= 1**
Probleemide arv	809	124

Tabel 11 Inimeste poolt tehtud ümbersuunamiste arv probleemide lõikes testandmestikul

Tabelis 12 on esitatud inimeste poolt tehtud ümbersuunamised valideerimisandmestikul. Selle andmestiku puhul on probleeme kokku 554 ning veamäär võrreldes testandmestikuga on suurem (17%).

Ümbersuunamiste arv	**0**	**>= 1**
Probleemide arv	460	94

Tabel 12 Inimeste poolt tehtud ümbersuunamiste arv probleemide lõikes valideerimisandmestikul

Tabelis 13 on välja toodud võrdlus RoBERTa mudeli ja inimese täpsuse vahel. Tabelist on näha, et mudel on inimese täpsusele lähedal, kuid jääb siiski alla. Testandmestiku puhul on erinevus 5%, kuid valideerimisandmestiku puhul on erinevus 9%.

Täpsused	**Testandmestik**	**Valideerimisandmestik**
Mudeli täpsus	82%	74%
Inimese täpsus	87%	83%

Tabel 13 RoBERTa mudeli ja inimese täpsuse võrdlus test- ja valideerimisandmestikul

Kuna mudeli täpsus jäi alla inimese täpsusele, siis ei saanud teha Jira pistikprogrammi lahendust automaatselt jaotavaks, vaid loodi soovitusi pakkuv lahendus. Seevastu on mudeli täpsus piisavalt hea, et aidata lahendada tootetoe tiimi skaleeritavust. Mudel saab aidata sellega, kui tootetoe tiimi lisanduvad uued agendid, siis sellele lahendusele tuginedes on neil võimalik jaotada uusi probleeme tiimidesse piisavalt hea täpsusega.

Tabelis 14 on näidatud RoBERTa mudeli ennustusvead (teistesse tiimidesse) ja varasemalt ümbersuunatud probleemid tiimide lõikes. Andmed pärinevad test- ja valideerimisandmestikult, kus on eemaldatud tiimid, mida mudeli treenimiseks ei kasutatud.

Arendustiim	**Ühiste probleemide osakaal ümbersuunatud probleemide suhtes**	**Ühised probleemid**	**Ümbersuunatud probleemid**	**Mudeli vead**
Veeb	38%	5	13	23
Parkimine	**77%**	20	26	58
Arveldus	55%	18	33	40
Ärikliendid	**88%**	15	17	29
Parkimisliidestused	46%	22	48	47
Maksed (rakendus)	**77%**	10	13	19
Parkimisalad	29%	7	24	41
Haldusveeb	23%	3	13	17
Parkimisluba	14%	2	14	4
Parkimiskaart	**100%**	6	6	10
Parkimise haldus	0%	0	2	8

Tabel 14 RoBERTa mudeli tegeliku tiimi ennustusvead ja ümbersuunatud probleemid tiimide lõikes test- ja valideerimisandmestikul

Tabelist 14 on näha, et tiimi parkimisluba puhul oli ühisosa väike, kuna mudel tegi ainult 4 ennustusviga, kuid ümber suunati 14 probleemi, mis viitab sellele, et mudel suudab ennustada antud tiimi suhtes paremini kui inimene.

Suurimad ühiste probleemide osakaalud ümbersuunatud probleemide suhtes on tiimidel parkimiskaart, ärikliendid, maksed (rakendus) ja parkimine. Suurt ühisosa valesti klassifitseeritud probleemide lõikes saab tõlgendada mitmeti.

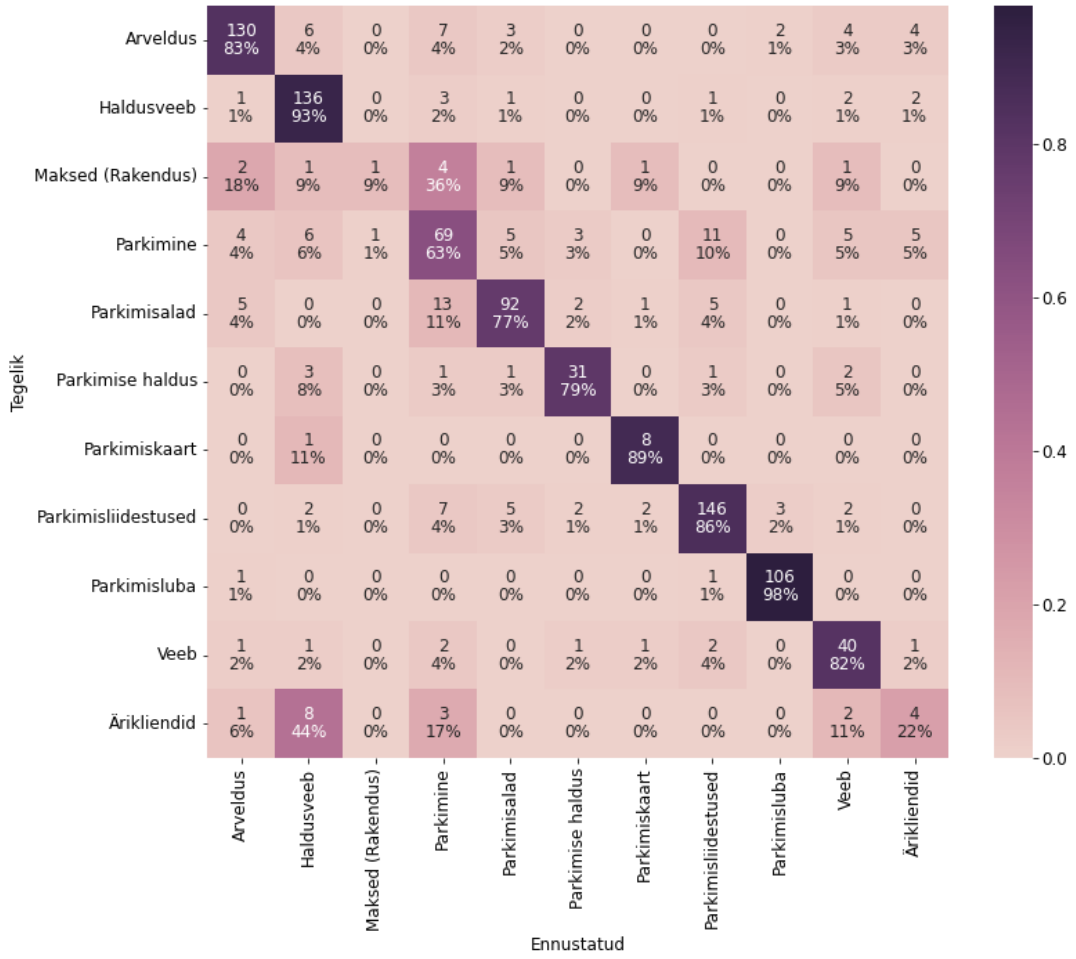
Üheks hüpoteesiks on, et antud valesti klassifitseeritud probleemid on loodud sellise sõnastusega, et need võiksid kuuluda mitmesse tiimi, mistõttu oli nii inimestel kui ka mudelil keeruline otsuseid teha. Sellised probleemid võivad tekkida juhul, kui tiimid on teatud valdkondades seotud, mistõttu võib neid probleeme olla raske klassifitseerida. Näiteks tiimid arveldus, maksed (rakendus) ja parkimine on oma valdkonna poolest tugevalt seotud.

Teiseks hüpoteesiks on, et teatud tüüpi probleemid võivad vajada mitmelt erinevalt tiimilt lahendust, mistõttu on probleeme suunatud mitme tiimi vahel. Hüpoteeside täiendavat uurimist on tehtud peatükis 5.3, et leida kinnitust, kas need vastavad tõele.

5.3 Parima mudeli klassifitseerimisedukus tiimide põhiselt

Käesolevas peatükis on uuritud lähemalt kahte madalaima klassifitseerimisedukusega tiimi, milleks olid maksed (rakendus) ja ärikliendid, et selgitada nende madalat klassifitseerimisedukust. Lisaks on võrdlusalusena uuritud ka kõige parema edukusega tiimi, milleks oli parkimisluba.

Joonisel 14 on näidatud RoBERTa mudeli veamaatriks testandmestikul, kus on näidatud klassifitseeritud probleemide arv koos reapõhiste normaliseeritud väärtustega.



Joonis 14 RoBERTa mudeli veamaatriks

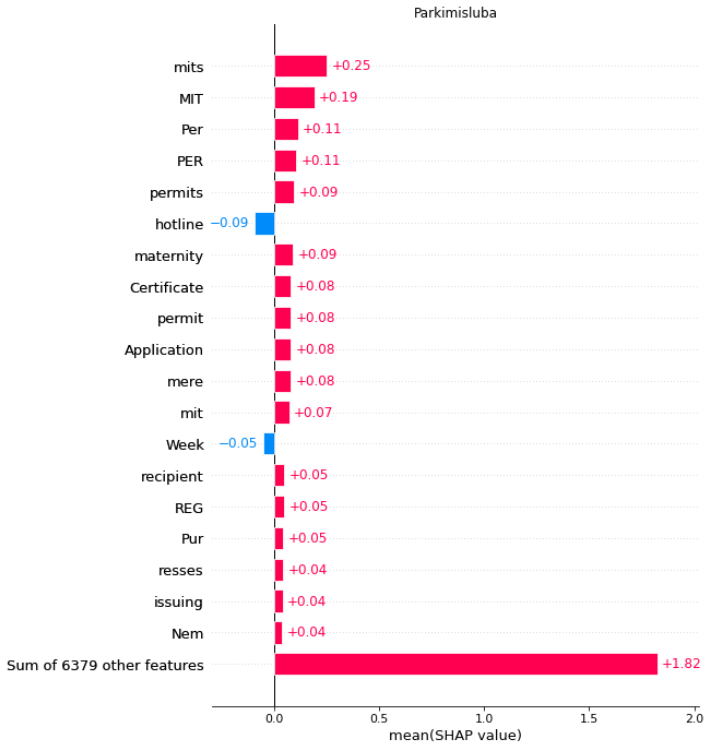
Tiimil parkimisluba oli tehtud mudeli poolt klassifitseerimisvigu tiimide parkimisliidestused ja arveldus suhtes. Kuna valesti klassifitseeritud probleeme oli kokku 7, siis oli võimalik neid probleeme käsitsi täpsemalt uurida. Selle käigus selgus, et tiimi arveldus puhul olid probleemid seotud tihedalt selle tiimi valdkonnaga, mistõttu kehtib peatükis 5.2 välja toodud esimene hüpotees, ehk probleemid on sõnastatud selliselt, et need võiksid kuuluda mitme tiimi valdkonda. Parkimisliidestuste tiimi suhtes tehtud mudeli vead olid ka varem inimeste poolt ümber suunatud, mistõttu ka selle tiimi suhtes peatükis 5.2 välja toodud esimene hüpotees kehtib.

Tiimil maksed (rakendus) oli klassifitseerimisvigu tehtud peaaegu kõikide tiimide suhtes, välja arvatud parkimisliidestused, parkimisluba ja ärikliendid. Õigesti klassifitseeriti ainult ühe korra ja valesti klassifitseeriti 11 korda. Valesti klassifitseeritud probleemide hulgast suunati neid ümber kuue probleemi puhul. Ka selle tiimi puhul olid kõik valesti klassifitseeritud probleemid tugevalt seotud teiste tiimide valdkondadega, mistõttu esimene hüpotees kehtib ka selle tiimi kohta.

Tiimi ärikliendid puhul oli valesti klassifitseerimisi tehtud tiimide veeb, parkimine, haldusveeb ja arveldus suhtes. Valesti klassifitseeritud probleeme oli kokku 26, millest 13 oli ka inimeste poolt ümber suunatud. Lähemalt uurides selgus, et 20 probleemi olid samuti teiste tiimide valdkondadega seotud, mistõttu kehtis ka selle tiimi puhul esimene hüpotees. Ülejäänud kuue probleemi puhul oli valdkond samuti seotud, kuid mitte nii tugevalt, mis viitab sellele, et mudelil polnud olnud piisavalt palju andmeid sisendiks, et luua tugevaid seoseid.

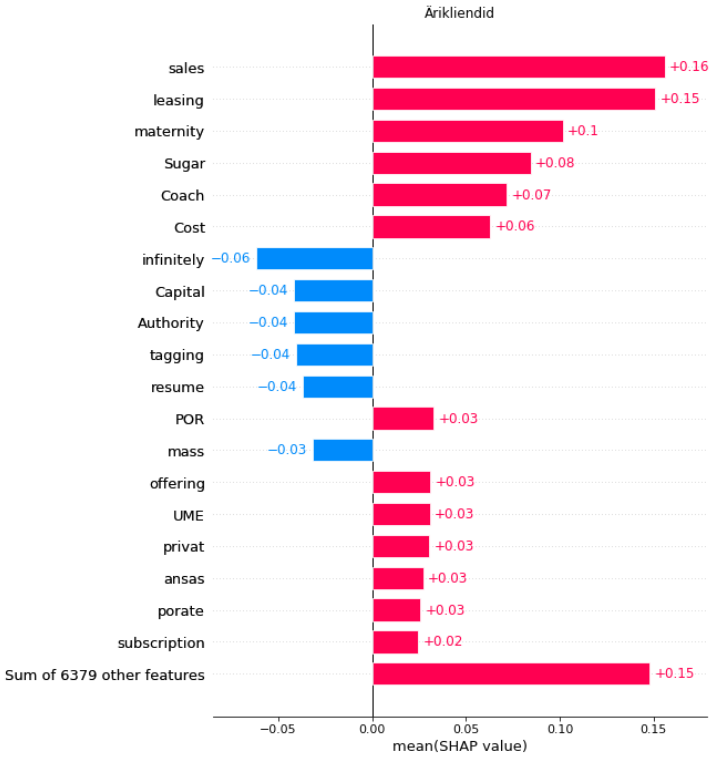
Täiendavaks analüüsimiseks on kasutatud SHAP teeki, mille abil oli võimalik analüüsida RoBERTa mudeli väljundit. Antud meetodi väljundid näitavad mudeli sisendite keskmist olulisust antud klassi suhtes. RoBERTa mudeli puhul olid sisendiks tekstide sõnad ja sõnaosad. Positiivsed väärtused suurendavad tõenäosust kuuluda antud klassi ning negatiivsed väärtused vähendavad tõenäosust kuuluda antud klassi. [37]

Joonisel 15 on välja toodud 20 kõige olulisemat sõnaosa tiimi parkimisluba kohta, mis tekkisid SHAP analüüsi abil. Antud jooniselt on näha, et esimesed viis sõnaosa on kõik seotud sõnaga *permits* ehk parkimisluba, mis viitab sellele, et antud tiimi puhul on väga selgelt eristatav domeen. Lisaks on mudelil ka palju muid sõnu, mis olid parkimisloa klassi ennustamisel summeerituna suure olulisusega (1.82).



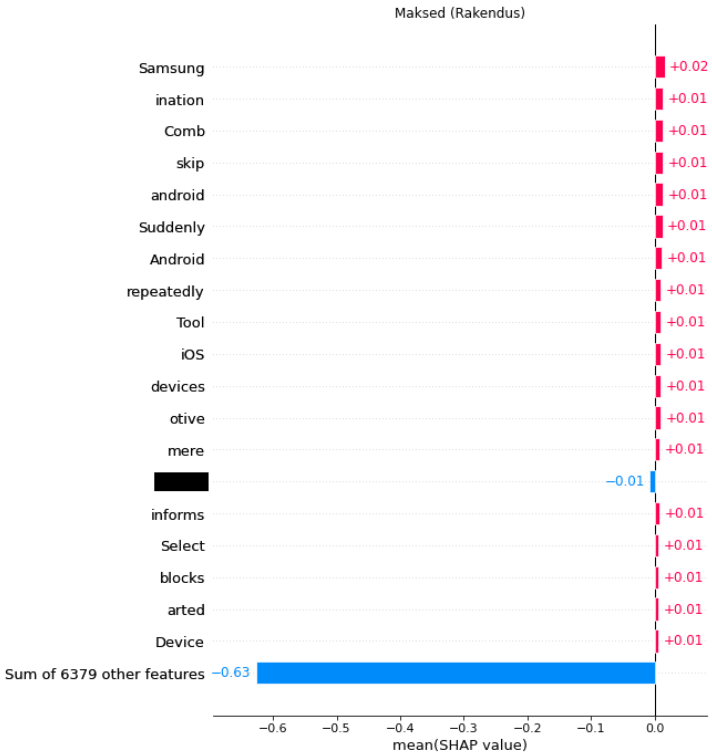
Joonis 15 Olulisemad 20 sõnaosa arendusmeeskonna parkimisluba ennustamise suhtes

Joonisel 16 on välja toodud 20 kõige olulisemat sõnaosa tiimi ärikliendid kohta. Seal on näha, et erinevalt tiimist parkimisluba, on sellel tiimil ainult üksikud sõnad, mis määravad antud tiimi kuuluvuse. Kaks kõige olulisemat sõna olid "sales" ja "leasing". Peale 20 olulisema sõnaosa mõjutasid ülejäänud sõnaosad summeerituna antud tiimi kuuluvust vähe (0.15).



Joonis 16 Olulisemad 20 sõnaosa arendusmeeskonna ärikliendid ennustamise suhtes

Joonisel 17 on näidatud meeskonna maksed (rakendus) 20 kõige olulisemat sõnaosa. Antud joonisel on näha, et ei olnud sõnu, mis mõjutaks olulisel määral antud tiimi kuuluvust. Lisaks on näha, et läbivalt on olulisemad sõnaosad seotud operatsioonisüsteemide ja nutitelefonidega nagu näiteks sõnad "Samsung", "Android" ja "iOS". Ülejäänud sõnaosad summeerituna osutasid negatiivset mõju antud tiimi klassifitseerimiseks (-0.63). Antud joonisel on üks sõna peidetud, kuna see sisaldas inimese nime.



Joonis 17 Olulisemad 20 sõnaosa arendusmeeskonna maksed (rakendus) ennustamise suhtes

Kõigi kolme tiimi lähemal uurimisel selgus, et peamiseks põhjuseks vigade tekkimisel oli see, et probleemide sisu oli seotud mitme tiimi valdkonnaga. See tähendab seda, et antud probleemide puhul ei olnud tekstide analüüs piisav, vaid oli vaja lisaks teada tiimide struktuuri ja tausta. Lahendusteks võiks olla kas muuta ettevõtte struktuuri, et tiimide valdkonnad oleksid paremini eristuvad või sõnastada sellist tüüpi probleeme täpsemalt, et oleks selgemini aru saada, millise tiimiga võiksid antud probleemid seotud olla.

5.4 Võrdlus varasemate töödega

Varasemate töödega võrdlusel on võetud aluseks kaks tööd, millest esimene uuris Istanbuli Tehnikaülikooli kasutajate toe probleemide klassifitseerimist vastavatesse ülikooli osakondadesse ja nende alamkategooriatesse. Antud töös oli kokku neli osakonda, mis omakorda jagunesid kokku 20ks alamkategooriaks. Töös kasutati SVM, Naive Bayes, KNN ja otsustuspuude meetodeid, millest parimaid tulemusi andsid SVM ja otsustuspuud. [1]

Teise töö puhul oli uuritud defektide raporteerimise klassifitseerimist vastavalt veatüübile. Veatüübi klasse oli 12 ning näideteks olid tarkvara viga, disainiviga, riistvaraline viga. Töö sisendiks oli ligikaudu 5500 teksti, mis pärinesid ABB ettevõtte

ühest meeskonnast. Antud töös kasutati SVM, Naive Bayes, KNN, otsustuspuude ja Random Forest meetodeid. Sarnaselt käesoleva tööga, selgus selle töö tulemusena, et kõige parema tulemuse andis SVM mudel F1-skooriga 60.76%. Kõige madalama F1-skoori sai Random Forest (39.85%), millele järgnes Naive Bayes mudel (41.69%). [3]

Käesoleva töö ning kahe varasema töö puhul oli peamiseks sarnasuseks see, et läbivalt andis häid tulemusi SVM mudel ning kõikide tööde puhul oli andmestik balansseerimata. Lisaks oli kõikides töödes välja toodud sarnased probleempüstitused, kus probleemide jaotamine ning nendele reageerimine peab olema kiire, mida on keeruline teostada inimeste poolt käsitsi jaotades. [1], [3]

5.5 Piirangud ja edasised arengud

Töö raames loodud mudel täitis seatud eesmärgi, kuid väheste probleemidega arendustiimide puhul oli klassifitseerimistäpsus siiski madal, mistõttu oleks vaja nende tiimide jaoks teha edasist tööd. Lisaks ei olnud võimalik luua sellist mudelit, mis suudaks klassifitseerida piisava täpsusega, et luua automaatset klassifitseerimise töövahendit, mis suudaks asendada inimest.

Selle parandamiseks aitaks edasine treeningandmete kogumine uutelt probleemidelt ja ka mudelite edasine optimeerimine väiksemate arendustiimide suhtes. Alternatiivselt saaks kasutada mudelit ka selliselt, et see klassifitseeriks ainult nende tiimide suhtes, mille puhul on mudelil suur tõenäosus õigesti klassifitseerida ja jätta ülejäänud probleemid inimese poolt eraldi üle vaatamiseks.

Peatükis 5.2 sai välja toodud erinevaid hüpoteese selgitamaks suurt ühisosa ümbersuunamiste ja mudeli klassifitseerimisvigade vahel. Antud töö raames leiti kinnitust esimesele hüpoteesile (segane probleemide sõnastus) kolme tiimi näitel. Teisele hüpoteesile, ehk mitme tiimi poolt lahendust vajavatele probleemidele, kinnitust ei leitud. Selle kinnitamiseks või ümber lükkamiseks aitaks samuti edasine andmete kogumine uutelt probleemidelt.

Peatükis 5.3 sai välja toodud mitme valdkonnaga tihedalt seotud probleemide üheks lahenduseks seda, et sõnastada probleeme täpsemaks. Töö raames aga ei tehtud edasist analüüsi selle osas, millist tüüpi probleemide puhul peaks sõnastust parandama. Selliste

probleemide tuvastamiseks saaks kasutada tekstikaeve meetodeid nagu näiteks teemade modelleerimine. [38]

Töö raames sai uuritud tootetoe probleemide jaotamist, kuid tootetoe efektiivsemaks tegemiseks on ka muid viise, mida töös ei uuritud. Üheks viisiks oleks luua sarnaste probleemide tuvastamine, et arendajal oleks lihtsam leida probleemile lahendust, mida on varasemalt sarnastes probleemides kasutatud. Kuigi Jira poolt on selline lahendus juba olemas [39], on see autori hinnangul algeline ning pole probleemide lahendamisel kasu andnud.

Töös pakutud lahenduse kasutamiseks teistes ettevõtetes on tarvis teha täiendusi eeltöötluse ja andmete alla laadimise protsessis. Kuigi andmete alla laadimine oli automatiseeritud, siis olemasolevate andmete uuendamise jaoks head protsessi ei olnud. Lisaks sai eeltöötluse osas loodud mitmeid reegleid, mis olid ettevõtte spetsiifilised. Kuna igal ettevõttel on suure tõenäosusega omad jaotamise ja filtreerimise reeglid, siis oleks vaja nende defineerimiseks paremat viisi, kuna töö raames olid need kirja pandud otse koodis.

6 Kokkuvõte

Käesoleva töö eesmärk oli luua EasyPark ettevõtte näitel lahendus, mis suudaks jaotada tootetoe probleeme vastavatesse arendustiimidesse. Uurimistöö käigus uuriti probleemide klassifitseerimist EasyPark ettevõtte tootetoe tekstide näitel, mis olid saadud Jira keskkonnast.

Eesmärgi täitmiseks loodi soovitusi andev Jira pistikprogramm koos taustteenusega, mis kasutab treenitud mudelit. Töö väljund võimaldab skaleerida tootetuge selliselt, et uutel inimestel oleks võimalik teha probleemide jaotamise osas otsuseid sarnase täpsusega nagu kogenud töötajad.

Töö käigus uuriti nelja erinevat meetodit: Naive Bayes, SVM, RoBERTa ja XLNet, mille hulgast andsid kõige paremaid tulemusi SVM ja RoBERTa. Kahe mudeli seast valiti edasiseks analüüsiks RoBERTa. Võrdluses inimesega oli RoBERTa mudeli klassifitseerimisedukus madalam, kuid oli piisavalt täpne, et luua soovitusi andev lahendus.

Kõige parem klassifitseerimisedukus oli tiimi parkimisluba puhul ning kõige madalam klassifitseerimisedukus oli tiimidel maksed (rakendus) ja ärikliendid. Peamine põhjus klassifitseerimisedukuse vähenemisel oli probleemide seotus mitme erineva tiimi valdkonnaga ning teatud tiimide puhul oli ka põhjuseks andmete vähesus.

Vastavalt töö tulemustele soovitab autor keerulisemate probleemide puhul neid selgemalt sõnastada või võimaluse korral struktureerida ettevõtte tiime selliselt, et oleks tiimide domeen selgemalt eristuv. Töö üheks edasiarenduseks on välja selgitada, millist tüüpi probleemide puhul peaks probleemi sõnastust selgemaks tegema.

Kasutatud kirjandus

[1] M. Altintas ja A. C. Tantug, „Machine Learning Based Ticket Classification in Issue Tracking Systems,“ %1 *The 2nd International Conference on Artificial Intelligence and Computer Science (AICS)*, Istanbul, 2014.

[2] P. K. Chilana, T. Grossman ja G. Fitzmaurice, „Modern Software Product Support Processes and the Usage of Multimedia Formats,“ %1 *CHI Conference on Human Factors in Computing Systems*, Vancouver, 2011.

[3] P. S. Parmar, P. K. Biju, M. Shankar ja N. Kadiresan, „Multiclass Text Classification and Analytics for Improving Customer Support Response through different Classifiers,“ %1 *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, 2018.

[4] Atlassian Corporation, „Jira automation basics,“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira/guides/expand-jira/automation. [Kasutatud 22 12 2022].

[5] EasyPark, „Our story,“ [Võrgumaterjal]. Available: https://easyparkgroup.com/our-story/. [Kasutatud 27 11 2022].

[6] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng ja C. Potts, „Learning Word Vectors for Sentiment Analysis,“ %1 *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Association for Computational Linguistics, 2011, pp. 142-150.

[7] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng ja C. Potts, „Recursive deep models for semantic compositionality over a sentiment treebank,“ *EMNLP*, kd. 1631, pp. 1631-1642, 2013.

[8] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu ja J. Gao, „Deep Learning Based Text Classification: A Comprehensive Review,“ *ACM Computing Surveys*, kd. 54, nr 3, pp. 2-27, 2020.

[9] S. Tripathi, R. Mehrotra, V. Bansal ja S. Upadhyay, „Analyzing Sentiment using IMDb Dataset,“ %1 *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, Bhimtal, 2020.

[10] D. S. Sachan, M. Zaheer ja R. Salakhutdinov, „Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function,“ *Proceedings of the AAAI Conference on Artificial Intelligence*, kd. 33, nr 1, p. 6943, 2019.

[11] G. Liachoudis, „Sentiment Analysis of Movie Reviews by Merging Comments from two Well-Known Platforms,“ Tilburg, 2020, pp. 18-19.

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer ja V. Stoyanov, „RoBERTa: A Robustly Optimized BERT Pretraining Approach,“ 26 07 2019. [Võrgumaterjal]. Available: https://arxiv.org/abs/1907.11692.

[13] B. P. a. L. Lee, „A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts,“ %1 *Proceedings of the ACL*, 2004.

[14] A. I. Kadhim, „Survey on supervised machine learning techniques for automatic text classification,“ *Artificial Intelligence Review*, p. 273–292, 1 6 2019.

[15] V. Narayanan, I. Arora ja A. Bhatia, „Fast and accurate sentiment classification using an enhanced Naive Bayes model.,“ %1 *Intelligent Data Engineering and Automated Learning - IDEAL 2013*, Springer Berlin Heidelberg, 2013, pp. 194-201.

[16] J. E. Ramos, „Using TF-IDF to Determine Word Relevance in Document Queries,“ 2003.

[17] C. D. Manning, P. Raghavan ja H. Schütze, „Scoring, term weighting and the vector space model,“ %1 *Introduction to Information Retrieval*, 2009, pp. 117-119.

[18] J. Devlin, M.-W. Chang, K. Lee ja K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,“ 2019. [Võrgumaterjal]. Available: https://arxiv.org/abs/1810.04805.

[19] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov ja Q. V. Le, „XLNet: Generalized Autoregressive Pretraining for Language Understanding,“ 2019. [Võrgumaterjal]. Available: https://arxiv.org/abs/1906.08237.

[20] Z. Yang ja Q. Le, „Transformer-XL: Unleashing the Potential of Attention Models,“ Google, 29 01 2019. [Võrgumaterjal]. Available: https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html. [Kasutatud 14 12 2022].

[21] S. Tong ja D. Koller, „Support Vector Machine Active Learning with Applications to Text Classification,“ *Journal of Machine Learning Research,* kd. 2, pp. 45-66, 2001.

[22] N. Kalcheva, M. Karova ja I. Penev, „Comparison of the accuracy of SVM kernel functions in text classification,“ %1 *2020 International Conference on Biomedical Innovations and Applications (BIA)*, 2020.

[23] A. Kumar, „SVM RBF Kernel Parameters with Code Examples,“ 18 06 2020. [Võrgumaterjal]. Available: https://vitalflux.com/svm-rbf-kernel-parameters-code-sample/. [Kasutatud 14 12 2022].

[24] scikit-learn, „RBF SVM parameters,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Kasutatud 14 12 2022].

[25] Y. Liu, R. Wang ja Y.-S. Zeng, „An Improvement of One-Against-One Method for Multi-Class Support Vector Machine,“ %1 *2007 International Conference on Machine Learning and Cybernetics*, Hong Kong, 2007.

[26] G. Singh, B. Kumar, L. Gaur ja A. Tyagi, „Comparison between Multinomial and Bernoulli Naive Bayes for Text Classification,“ %1 *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, 2019.

[27] A. M. Kibriya, E. Frank, B. Pfahringer ja G. Holmes, „Multinomial Naive Bayes for Text Categorization Revisited,“ %1 *AI 2004: Advances in Artificial Intelligence*, Hamilton, 2004, p. 491.

[28] S. Raschka, „Naive Bayes and Text Classification I - Introduction and Theory,“ 14 02 2017. [Võrgumaterjal]. Available: https://arxiv.org/abs/1410.5329.

[29] A. Tharwat, „Classification Assessment Methods,“ *Applied Computing and Informatics,* kd. 17, nr 1, pp. 168-192, 2020.

[30] M. Heydarian, T. E. Doyle ja R. Samavi, „MLCM: Multi-Label Confusion Matrix,“ *IEEE Access,* kd. 10, pp. 19086-19087, 2022.

[31] „Fernet Spec,“ [Võrgumaterjal]. Available: https://github.com/fernet/spec/blob/master/Spec.md. [Kasutatud 10 11 2022].

[32] Princeton University, „About WordNet.,“ 2010. [Võrgumaterjal]. Available: https://wordnet.princeton.edu/citing-wordnet. [Kasutatud 11 11 2022].

[33] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei ja S.-H. Deng, „Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization,“ *Journal of Electronic Science and Technology,* kd. 17, pp. 27-29, 2019.

[34] Amazon Web Services, Inc, „Amazon EC2 G5 Instances,“ [Võrgumaterjal]. Available: https://aws.amazon.com/ec2/instance-types/g5/. [Kasutatud 26 11 2022].

[35] „Flask,“ [Võrgumaterjal]. Available: https://github.com/pallets/flask. [Kasutatud 13 12 2022].

[36] „Part 1: Build a Jira hello world app,“ Atlassian, 07 09 2022. [Võrgumaterjal]. Available: https://developer.atlassian.com/platform/forge/build-a-hello-world-app-in-jira/. [Kasutatud 09 11 2022].

[37] S. Lundberg, „text plot,“ [Võrgumaterjal]. Available: https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/text.html. [Kasutatud 3 12 2022].

[38] M. Schweinberger, „Topic Modeling with R,“ The University of Queensland, Australia. School of Languages and Cultures, 13 09 2022. [Võrgumaterjal]. Available: https://slcladal.github.io/topicmodels.html. [Kasutatud 01 01 2023].

[39] Atlassian Corporation, „Find similar requests,“ [Võrgumaterjal]. Available: https://support.atlassian.com/jira-service-management-cloud/docs/what-are-similar-requests/. [Kasutatud 22 12 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Gunnar Joosep Hint

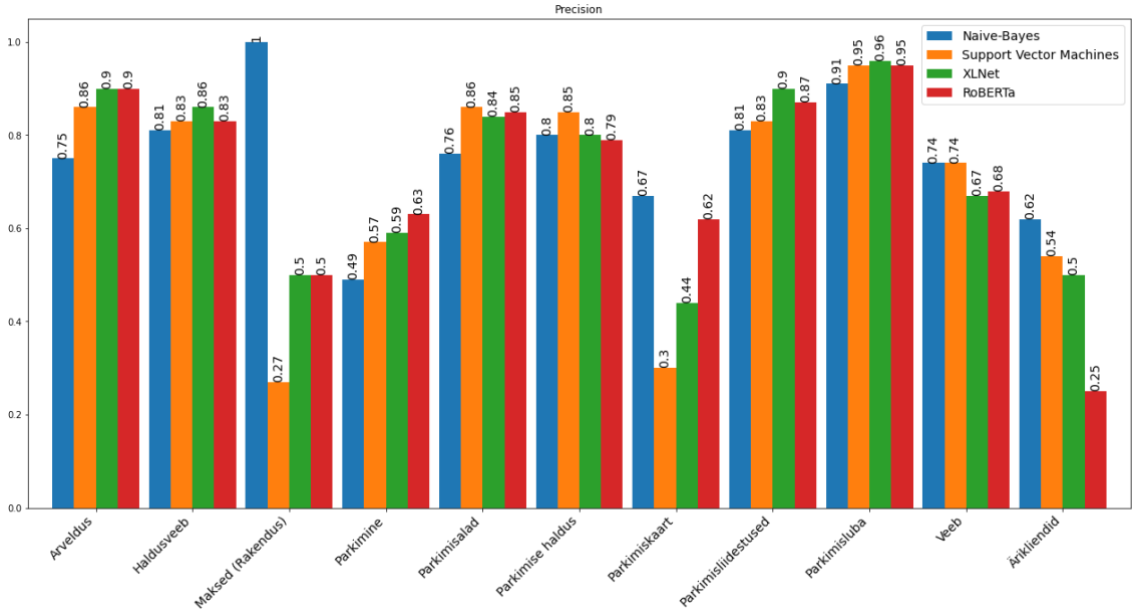
1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Tootetoe probleemide jaotamine arendusmeeskondadesse kasutades teksti klassifitseerimise meetodeid", mille juhendaja on Ahti Lohk
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

03.01.2023

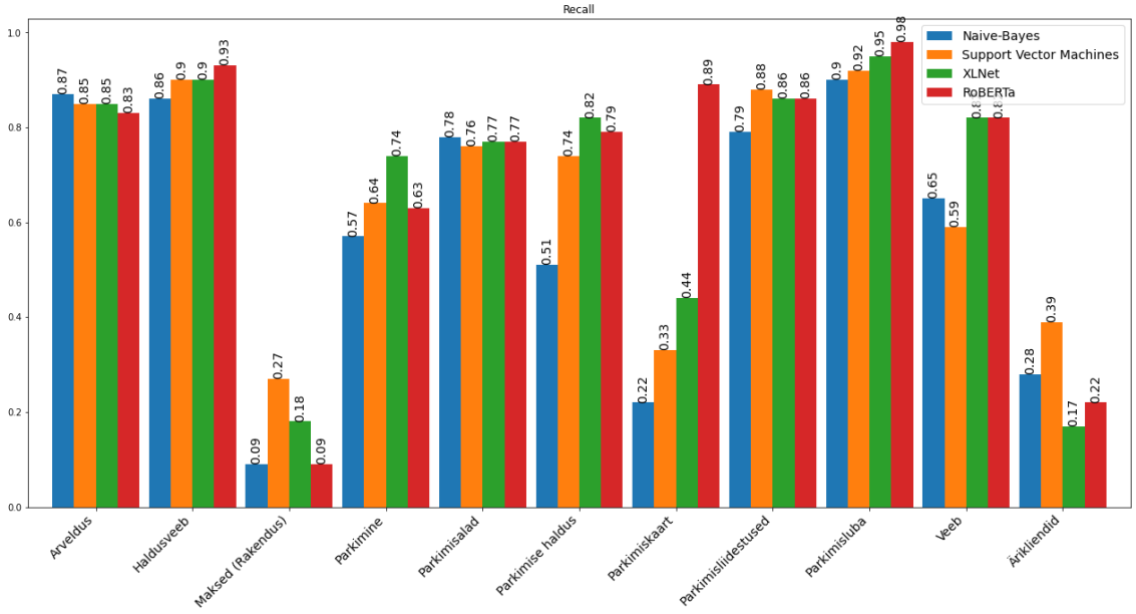
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Mudelite täpsus ja saagis testandmestiku põhjal

Mudelite kordustäpsus testandmestiku põhjal tiimide lõikes:

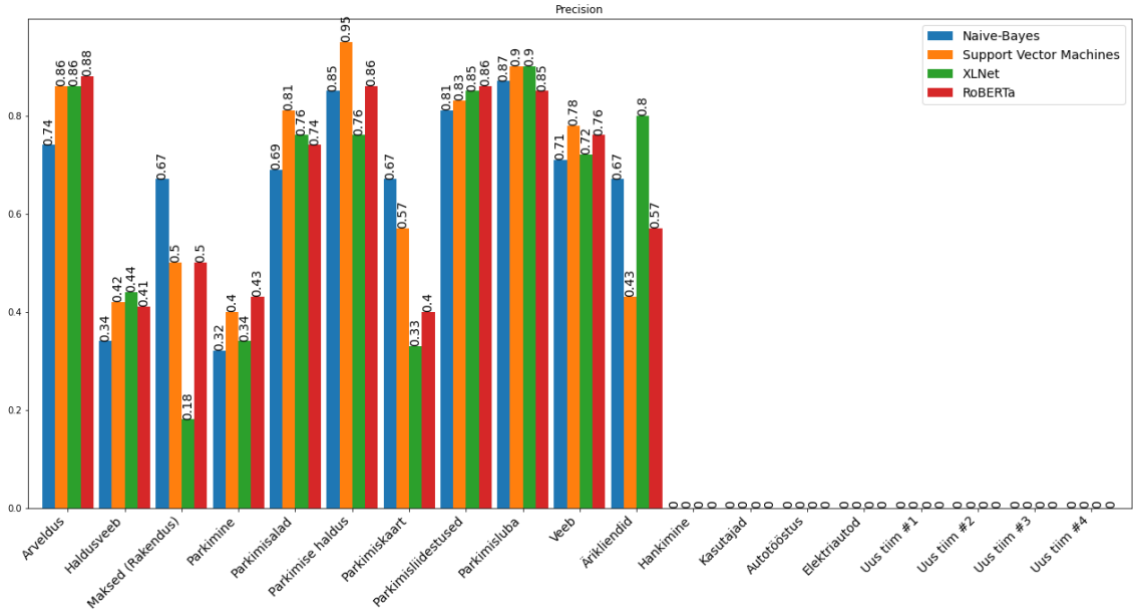


Mudelite saagis testandmestiku põhjal tiimide lõikes:

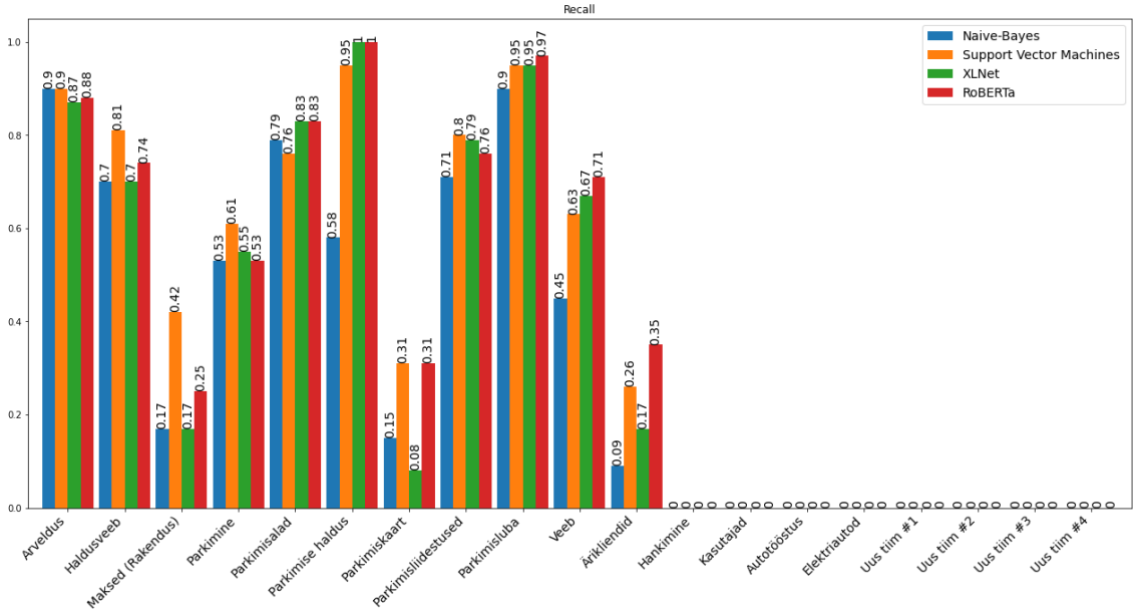


Lisa 3 – Mudelite täpsus ja saagis valideerimisandmestiku põhjal

Mudelite kordustäpsus valideerimisandmestiku põhjal tiimide lõikes:

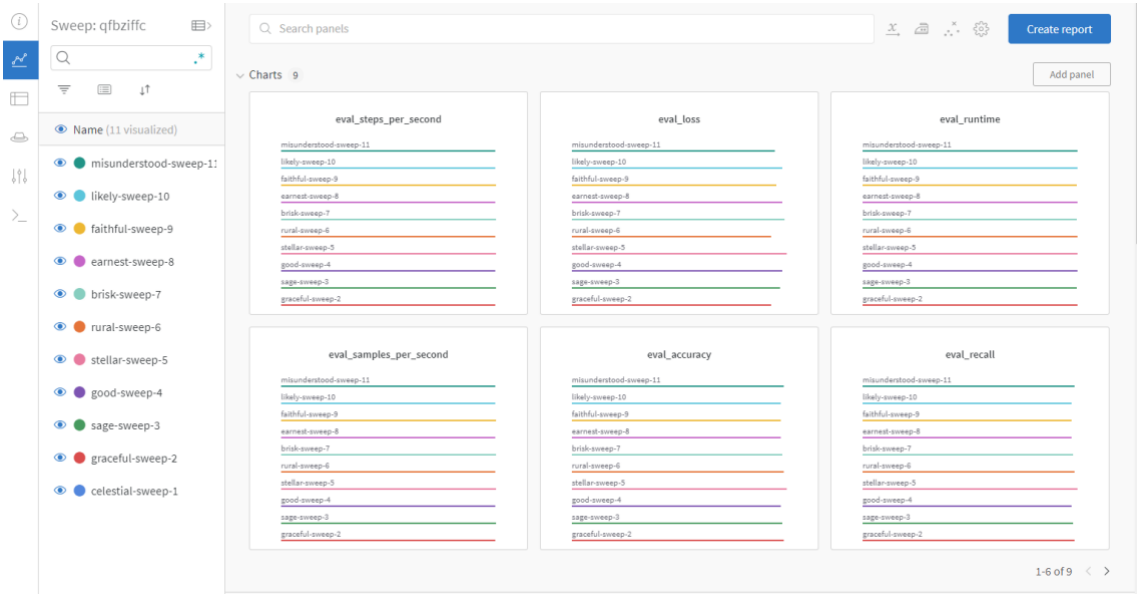


Mudelite saagis valideerimisandmestiku põhjal tiimide lõikes:



Lisa 4 – W&B keskkonna kuvatõmmised

Ülevaade hüperparameetrite otsingu tulemustest:



Ülevaade parameetrite mõjust ja klassifitseerimistulemustest tiimide põhiselt: