TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marten Tamme 193779IAIB

Finantsmeeskondade tööde planeerimine tööhaldustarkvaras Uku

Bakalaureusetöö

Juhendaja: Priit Järv
doktorikraad

Tallinn 2022

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: Marten Tamme

25.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on finantsmeeskondade tööde parem planeerimine tööhaldustarkvaras Uku. Antud funktsionaalsuse tellijaks on Artify OÜ.

Kasutajate poolt eelseadistatud igakuine tööplaan ehk ülesannete järjestus ei ole kooskõlas selle tegeliku täitmisega. Peamisteks põhjusteks on kasutaja enda harjumused ülesandeid teises järjekorras teha või tema klientide harjumused dokumentide esitamisel, mis on aluseks ülesande täitmisele.

Käesoleva töö raames on valminud algoritm, mis suudab kasutaja korduvaid ülesandeid paigutada tööplaani selliselt, et need läheksid paremini kokku tema harjumustega. Lisaks valmisid masinõppe mudelid ülesannete ajahinnangu ennustamiseks, mis on loodud, et tulevikus kasutajate töökoormust hinnata. Praeguseks ei ole töös loodud funktsionaalsused integreeritud põhirakendusega.

Algselt oli plaanis luua algoritm, mis koostaks kasutajale efektiivsema tööplaan ehk süsteem paigutaks ülesanded vastavalt nende kestustele ja tähtajale tööplaani, sõltumata sealjuures kasutaja varasematest harjumustest. Lõputöö käigus selgus, et selline funktsionaalsus ei ole kasutaja jaoks tegelikult vajalik ning sellest tulenevalt sai jooksavalt nii ärivajadusi kui ka algoritmi tööpõhimõõteid korrigeeritud.

Funktsionaalsus on loodud programmeerimiskeeles Python ning on mõeldud kasutamaks mikroteenusena põhirakenduse kõrval. Ajahinnangute ennustamiseks on kasutatud masinõppe teeki Scikit-Learn ning tekstitöötluse teeki Gensim.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 57 leheküljel, 9 peatükki, 26 joonist, 3 tabelit.

Abstract

Planning the Work of Financial Teams in Job Management Software Uku

The aim of this bachelor's thesis is to better plan the work of financial teams in the work management software Uku. The client of this functionality is Artify OÜ.

The monthly work plan preset by the users, i.e. the sequence of tasks, is not in line with its actual performance. The main reasons are the user's own habits of performing the tasks in a different order or the habits of his / her customers when submitting documents, which is the basis for completing the task.

Within the framework of this work, an algorithm has been developed that can place repetitive tasks in the work plan in a way that better matches their habits. In addition, machine learning models for predicting task time have been developed to estimate user workload in the future. Currently, the functionalities created in this work are not integrated with the main application.

Initially, the plan was to create an algorithm that would create a more efficient work plan for the user, i.e. the system would place tasks according to their duration and deadline in the work plan, regardless of the user's previous habits. During the dissertation, it became clear that such functionality is not necessary for the user, and as a result, both the business needs and the operating principles of the algorithm were adjusted on an ongoing basis.

The functionality is written in the Python programming language and is intended to be used as a microservice alongside the main application. The machine learning library Scikit-Learn and the word processing library Gensim have been used to predict time estimates.

The thesis is in Estonian and contains 57 pages of text, 9 chapters, 26 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	Application Programming Interface; rakendusliides
CSV	Comma-separated values; Komaga eraldatud väärtuste fail
Eesrakendus	Rakendus, mida kasutab lõppkasutaja
Hüperparameeter	Tehisnärvivõrgu arhitektuuri või treenimisalgoritmi parameeter, mille väärtus määratakse enne treenimisprotsessi alustamist
JSON	JavaScript Object Notation; andmevahetusvorming
ORM	Object Relational Mapping; tehnoloogia, mis seob objektorienteeritud programmeerimiskeeli andmebaasidega
SQL	Structured Query Language; andmebaasi päringukeel, mis on loodud relatsioonbaasihaldurite jaoks
Tagarakendus	Rakendus, mida lõppkasutaja ei näe ja mis vastutab andmetega manipulatsiooni eest
UI/UX	user interface/user experience; kasutajaliides ja kasutuskogemus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tööhaldustarkvara Uku[1] on mõeldud finantsmeeskondade ja raamatupidamisbüroode töö efektiivsuse ning kasumlikkuse tõstmiseks. Uku abiga on raamatupidajal lihtsam enda ülesandeid planeerida ja tööle kuluvat aega mõõta. Selleks seadistatakse iga kliendi kohta kindla intervalli järel korduvad ülesanded, mis moodustavad raamatupidaja tööplaani.

Kasutajate poolt eelseadistatud tööplaan ehk ülesannete järjestus ei ole aga kooskõlas selle tegeliku täitmisega. Kõigist 550 tuhandest korduvast ülesandest 96% ülesande planeeritud alguskuupäev ei lange kokku tegeliku ülesande alustamise kuupäevaga. Peamisteks põhjusteks on kasutaja enda harjumused ülesandeid teises järjekorras teha või tema klientide harjumused dokumentide esitamisel, mis on aluseks ülesande täitmisele. Sellest tingituna, ei ole kasutajal selget ülevaadet, mis päeval ta mis ülesandeid peaks tegema ning see omakorda võib põhjustada tähtaegade ületusi (esines 21% ülesannetes), mis võivad lõppeda rahalise kahjuga nii kliendile kui raamatupidamisbüroole.

Käesoleva töö eesmärgiks on luua algoritm, mis suudab koostada kasutaja harjumustele vastava tööplaani ehk paigutada ülesanded tööplaani sellistele päevale, kus kasutaja nendega kõige suurema tõenäosusega alustab. Selle tulemusel on kasutajal silme ees ainult need ülesanded, millega ta tegelema peaks. Selline algoritm aitab vähendada raamatupidajate tähtaegade ületusi ning tõstab raamatupidajate produktiivsust.

Selleks, et tulevikus prognoosida kasutaja töökoormust, on tarvis teada kui kaua üks või teine ülesanne aega võtab. Käesolevas töös on loodud kaks masinõppemudelit ülesande ajahinnangu ennustamiseks. Põhjus, miks ajahinnang tuleb süsteemi poolt luua, on kasutajate käitumismustrid, mis on näidanud, et nad kas ei oska hinnata ülesande kestust või ei soovi ajahinnanguid ise sisestada. Ilma ajahinnanguteta pole aga võimalik tulevikus kasutaja töökoormuseid prognoosida ning ülekoormusohte tuvastada.

[1] https://www.getuku.ee/

2 Projekti ülesehitus

Selles peatükis kirjeldatakse projekti ülesehitust. Osas 2.1 tuuakse välja käesolevas töös loodav funktsionaalsus, osas 2.2 kirjeldatakse arhitektuuri ning osas 2.3 töös kasutatud tehnoloogiaid, osas 2.4 põhjendatakse masinõppeteegi valik ning osas 2.5 kirjeldatakse projekti seadistuses kasutatavaid tehnoloogiaid.

2.1 Funktsionaalsus

Käesoleva töö funktsionaalsusteks on tööplaanis olevate korduvate ülesannete kordumisreeglite korrigeerimine ning nii korduvate kui ka juhuslike ülesannete ajahinnangute ennustamine.

Kordumisreegli korrigeerimine algoritmi poolt tähendab seda, et muudetakse reeglit, mille abil süsteem genereerib ülesandeid teatud intervalli tagant kasutaja tööplaani. Korrigeerimise eesmärk on paigutada ülesanne kasutaja tööplaani sellisele päevale, kus tavaliselt kasutaja selle ülesandega tegelema hakkab. Kuna lõputöö raames optimeeriti kuise kordumisega ülesandeid, siis ei ole mõistlik teha korrigeerimist reaalajas, kuna reegli muutumiseks ehk uue info tulekuks läheb aega minimaalselt üks kuu, seega on efektiivsem reegel eelgenereerida ning salvestada põhirakenduse andmebaasi. Reegli uuendamine toimuks algoritmi jooksutamisel igakuise intervalli järel.

Ajahinnangute mudelite treenimisel on samuti mõistlik kasutada igakuist intervalli, kuna lühema perioodi korral ei teki piisavalt palju uusi andmeid, et need tulemusi muudaks. Korduvate ülesannete ajahinnanguid saab samuti ette arvutada ning salvestada andmebaasi koos korrigeeritud reeglitega. Juhuslike ülesannete puhul on tarvis põhirakendusel küsida asünkroonselt ülesandele ajahinnangut kui kasutaja seda ise määranud ei ole.

2.1.1 Korduva ülesande korrigeeritud reegel koos ajahinnanguga

Korduva ülesande korrigeeritud reegli ja ülesande ajahinnangu leidmiseks on tarvis sisendina kasutada kasutaja identifikaatorit. Selle tulemusel leitakse kõik kasutajaga

seotud baasülesannetele korrigeeritud kordumisreeglid ning ajahinnangud. Vastus tagastatakse JSON objekti kujul, kus objekti võtmeks on baasülesande identifikaator ning väärtuseks korrigeeritud kordumisreegli JSON objekt, mille sisse on lisatud ülesande ajahinnang.

2.1.2 Juhusliku ülesande ajahinnang

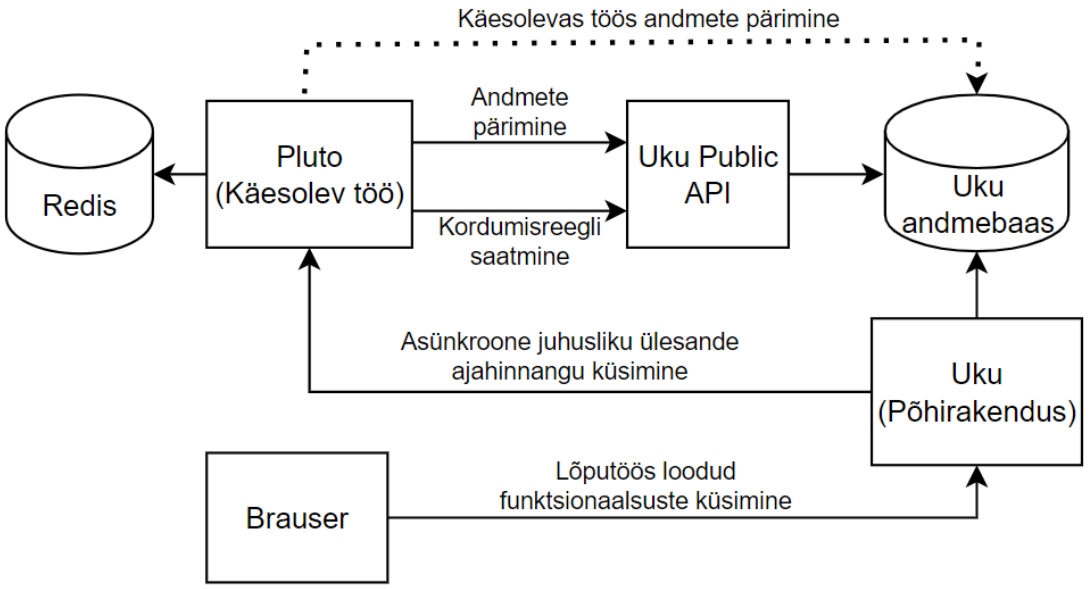
Juhusliku ülesande ajahinnangu leidmiseks on vaja sisendina kasutada järgmisi atribuute:

1. Ülesande pealkiri
2. Ettevõtte identifikaator
3. Loomiskuupäeva kuu number
4. Kliendi identifikaator
5. Kliendi registrikood
6. Ülesande looja/täitja identifikaator

Nende sisendatribuutide kaudu arvutatakse treenitud mudeli põhjal ajahinnang, mille ühikuks on sekund. Eelpool nimetatud atribuudid kujunesid välja andmeanalüüsi käigus ning mudel treeniti nende atribuutide alusel. Selleks, et mudelilt ülesandele ajahinnangut küsida, on tarvis teada samu atribuute, mis treenimisel kasutati, et andmed sobilikule vektorkujule viia.

2.2 Arhitektuur

Käesoleva töö funktsionaalsus on loodud eraldi tarkvara projektina ning kannab nime Pluto. Ettevõtte soov on kasutada Pluto funktsionaalsust mikroteenusena põhirakenduse kõrval, kus Pluto kasutaks põhirakendusega suhtlemiseks avalikku rakendusliidest (*public* API). Käesolev töö veel rakendusliidest ei kasuta, kuna see on alles tegemisel, seetõttu kasutab töös loodud funktsionaalsus andmete küsimiseks otse põhirakenduse andmebaasi (vt joonis 1).



Joonis 1. Arhitektuur

Mikroteenuse puhul on tegemist arhitektuuristiiliga, mis struktureerib rakenduse teenuste kogumina, mis on paremini hooldatav ning testitav, kuna ei ole läbipõimunud teiste teenustega [1]. Pluto realiseerimiseks mikroteenusena luuakse pilveteenuse server, milles on nii käesolevas töös loodud funktsionaalsus kui ka Redis andmebaas, mida kasutatakse esialgu ajahinnangute mudelite treenimisprotsessis.

Igakuise intervalli tagant käivitatakse manuaalselt ülesannete ajahinnangu mudelite treenimine ning tööplaani algoritmi arvutused. Hiljem on plaanis vastav protsess automatiseerida. Manuaalseks rakendamiseks lisatakse Plutosse CSV fail, milles on kõigi nende ettevõtete avaliku rakendusliidese võtmed, kes on lubanud oma andmeid töödelda. Seejärel käivitatakse mikroteenuses olevad funktsionaalsused.

Andmete pärimine toimub ettevõtte kaupa ning selleks kasutatakse vastava ettevõtte avaliku rakendusliidese võtit. Selle tulemusel saadakse ettevõtte andmed, mida on tarvis ajahinnangute mudelite treenimiseks ning tööplaani algoritmi rakendamiseks. Andmeid päritakse osade kaupa, et vähendada koormust põhirakenduse andmebaasile. Hiljem viiakse kogutud andmed Plutos vastavale kujule, et neid treenimiseks ja arvutamiseks kasutada.

Peale korrigeeritud kordumisreeglite ning korduvate ülesannete ajahinnangute arvutamist salvestatakse need Uku andmebaasi, kasutades selleks Uku avaliku rakendusliidest.

Hiljem saab Uku vastavaid reegleid otse enda andmebaasist kasutada. Kui kasutaja loob juhusliku ülesande ja jätab sellele ajahinnangu määramata, siis toimub asünkroonne päring, mille tulemusel küsitakse Plutolt ülesandele ajahinnang, mis ülesandele lisatakse.

2.3 Kasutatud tehnoloogiad

Käesolevas töös kasutatakse programmeerimiskeelt Python 3, arenduskeskkonda PyCharm, masinõppeteeki Scikit-Learn, andmemassiivide haldamisteeki NumPy, graafikuteeki Matplotlib ja tekstitöötlus teeki Gensim. Projekti seadistuses on kasutatud tarkvara Docker, relatsioonilist andmebaasi MySQL, ORM-i SQLAlchemy ning võti-väärtus andmebaasi Redis.

Python 3 on platvormiülene kõrgetasemelise andmestruktuuriga, interpreteeritud, objektorienteeritud programmeerimiskeel. Laialdane valik masinõppe- ja andmetöötlus-teeke teeb programmeerimiskeelest Python hea valiku masinõppe mudelite ning algoritmide arendamiseks [2].

PyCharm on JetBrains ettevõtte poolt loodud integreeritud arenduskeskkond, mis on mõeldud programmeerimiskeele Python jaoks. PyCharm sisaldab palju sisseehitatud tööriistu, mis aitavad arendajal leida vigu koodist, luua ühendust andmebaasiga ning testida koodi [3].

Scikit-Learn on vabavaraline masinõppeteek, mis toetab nii juhendatud kui juhendamata õpet. Lisaks sisaldab Scikit-Learn erinevaid tööriistu mudeli sobitamiseks, andmete eeltöötluseks ja mudeli hindamiseks [4].

NumPy on programmeerimiskeele Python teadusliku andmetöötluse põhipakett. Tegemist on teegiga, mis aitab andmemassiive hallata ja töödelda. NumPy on kirjutatud programmeerimiskeeles C, milles on suurte andmestike töötlemine kiirem kui programmeerimiskeeles Python [5].

Matplotlib on programmeerimiskeele Python ja selle numbrilise matemaatika laienduse NumPy graafikuteek. Matplotlib on terviklik teek staatiliste, animeeritud ja interaktiivsete visualisatsioonide loomiseks [6].

Gensim on avatud lähtekoodiga teek, mis võimaldab tõhusalt esitada dokumente semantiliste vektoritena. Gensim algoritmid avastavad automaatselt dokumentide

semantilise struktuuri, uurides nende statistilisi koosesinemise mustreid [7]. Käesolevas lõputöös on kasutatud Gensim Doc2Vec mudelit, mis esitab igat dokumenti vektorina ja on meetodi Word2Vec üldistus [8].

Docker on vabavaraline tarkvara, mis teostab operatsioonisüsteemi tasemel virtualiseerimist ja on mõeldud rakenduste arendamiseks, tarnimiseks ja käitamiseks [9].

MySQL on avatud lähtekoodiga relatsiooniline andmebaasihaldussüsteem, millesse saab päringud esitada standardiseeritud ehk SQL kujul [10].

SQLAlchemy on programmeerimiskeele Python SQL tööriistakomplekt ja objekt-relatsiooniline kaardistaja (ORM), mis annab rakenduse arendajale täieliku SQL võimsuse ja paindlikkuse [11].

Redis on avatud lähtekoodiga mälusisene võti-väärtus salvestusruum, mida kasutatakse põhiliselt andmebaasi ning vahemäluna [12].

2.4 Masinõppeteegi valik

Selles peatükis põhjendatakse masinõppe teegi Scikit-Learn sobivust ülesannete ajahinnangu mudeli loomiseks ning võrreldakse teeki kahe teise tuntud masinõppe teegi PyTorch ning TensorFlow teekidega.

Scikit-Learn puhul on tegemist ühe populaarseima masinõppeteegiga, mida on kerge projekti üles seada ning millel on suur kasutajaskond ja aktiivne tugi, mis muudab teegi usaldusväärseks ning jätkusuutlikuks. Nõrkuseks võib välja tuua, et Scikit-Learn on ühe arhitektuuripõhine, seega ei ole võimalik luua sügavaid närvivõrgu mudeleid.

Nii Scikit-Learn kui TensorFlow on mõlemad loodud selleks, et aidata arendajal uusi mudeleid luua ning neid võrrelda, seega on tegemist kahe üpris sarnase teegiga. Erinevused tulenevad sellest, et TensorFlow puhul on võimalus vaadata mudeli kapoti alla ning optimeerida ja muuta mudelit tõhusamaks. Seevastu on Scikit-Learn laiema mudelivalikuga ning rohkem üldistatud teek kui TensorFlow, mis annab Scikit-Learn teegile eelise erinevaid mudeleid probleemi lahendamiseks katsetada ning selle põhjal parim välja valida, mida hiljem täiustada saab. Kuna lõputöös lahendatav ajahinnangute ennustamise probleem sisaldab andmestikus palju juhuslikkust ning sobiliku mudeli leidmine võib osutuda keeruliseks, siis on mõistlik eelistada teeki Scikit-Learn [13].

Scikit-Learn puhul on tegemist kõrgetasemelise masinõppe teegiga, kus on võimalik mõne koodi reaga mudel luua. Seevastu PyTorch puhul on tegemist madalatasemelise süvaõppe teegiga, kus mudeli loomisel on võimalik minna rohkem detailidesse ning kohandada mudelit vastavalt vajadusele [14]. Kuna lõputöö eesmärgiks on katsetada ajahinnangute ennustamist läbi masinõppe mudeli, siis mõistlikum on alustada lihtsama masinõppe teegiga ehk Scikit-Learn teegiga ning tulevikus proovida keerukamaid ning rohkem võimalust pakkuvaid teeke.

2.5 Projekti seadistuses kasutatavad tehnoloogiad

Bakalaureusetöös loodav funktsionaalsus on loodud eraldiseisva projektina, millele on valitud vastavad tehnoloogiad, et projekti mugavalt üles seada ja hallata.

Projekti ülesehituses on kasutatud tarkvara Docker, mis toestab operatsioonitasandil virtualiseerimist. Lõputöös muudab see arendusprotsesse lihtsamaks tänu sellele, et projekti on võimalik kiirelt ja mugavalt ükskõik millises masinas üles seada. Tarkvara Docker kasuks sai otsustatud, kuna põhirakenduses Uku kasutatakse sama tarkvara ning mõistlik on hoida ettevõttesiseselt tehnoloogia valikutes ühtset joont.

Kuna lõputööks vajalik andmestik pärineb põhirakenduse andmebaasist MySQL ning kasutatakse ORM-i SQLAlchemy, siis sai sama valik tehtud ka käesolevas projektis.

Vahemälu andmebaasina sai kaalutud nii võti-väärtus andmebaasi Redis kui mitte-realtsioonilse andmebaasi MongoDB kasutamist. Andmeid, mida on tarvis salvestada, tuleneb ajahinnangute mudeli treenimisel tekkivatest optimeerimistest. Ühe näitena võib tuua JSON objekti, mis hoiab teadmist selle kohta, mis vektori indeksile, mis ettevõtte identifikaator vastab. Neid andmeid on oluline salvestada, et hiljem oleks võimalik treenitud mudelit kasutada. Andmete salvestamisel tuleb arvestada nende kättesaadavusega. Kuna juhusliku ülesande ajahinnangu saamine toimub reaalajas, siis peab mudeliga seotud andmete lugemine toimuma kiiresti.

MongoDB puhul on tegemist mitte-relatsioonilise ehk mitte-SQL andmebaasiga, kus andmeid hoitakse paindlikes JSON-dokumentides. Just paindlik andmete hoidmine muudab MongoDB kiiremaks kui relatsioonilised andmebaasid, kuid kettale salvestamise tõttu on MongoDB aeglasem kui mälusisene andmebaas Redis.

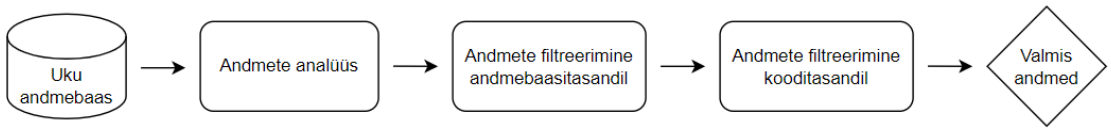
Käesolevas töös on kasulikum kasutada vahemäluna võti-väärtus andmebaasi Redis, kuna salvestatavaid andmeid on samuti mugavam hoida võti-väärtus paarides. Andmete muutmälus hoidmine annab parima tulemuse andmete kiireks lugemiseks, mis aitab kiirendada ülesannete ajahinnangu mudelite treenimise ja nende rakendamise protsesse.

3 Andmestik

Lõputöös käsitletavaks andmestikuks on põhirakenduse Uku andmebaas, kuhu on andmeid kogutud alates aastast 2018. Sellest andmebaasist valitakse välja sobilikud andmed nii mudelite treenimiseks kui ka algoritmi tööks.

3.1 Andmestiku ettevalmistamine

Allolev joonis kirjeldab üldist protsessi, kuidas andmebaasis olevatest andmetest saadakse „valmis andmed“, mida käesolevas töös kasutatakse.



Joonis 2. Andmestiku ettevalmistamine

Põhirakenduse andmebaasi analüüsides leitakse tabelid ning sobilikud atribuudid funktsionaalsuse realiseerimiseks. Valitud tabelitest on täpsemalt juttu osas 3.2 Andmebaasi tabelid. Põhiline andmete filtreerimine toimub andmebaasitasandil, kus tingimused sõltuvad vastavalt funktsionaalsuse loomisele, millest on juttu peatükkides 4 ja 5. Keerukamaid filtreerimisi on mõistlikum ja efektiivsem teostada kooditasandil. Kooditasandil filtreerimise tingimusteks on:

1. Ülesande kestuse põhjal ekstreemsuste ning müra eemaldamine
2. Kliendi andmebaasiväliste lisaandmete olemasolu kontrollimine
3. Ülesannete arvu piiramine kliendi lõikes

Täpsemalt tuleb juttu kooditasandil filtreerimisest peatükis 4 ja 5. Peale filtreerimist on saadud „valmis andmed“, mida saab kasutada funktsionaalsuste realiseerimiseks.

3.2 Andmebaasi tabelid

Põhirakenduse andmebaasi analüüsides leiti sobilikud atribuudid nii ajahinnangu mudelite treenimiseks kui ka algoritmi loomiseks. Lõputöö andmestiku moodustavad järgmised andmebaasi tabelid: konto, ülesanne, klient, ajasisestus ning määratud ülesanne (vt Lisa 2 – Käesolevas töös kasutatavad põhirakenduse andmebaasi tabelid).

3.2.1 Konto

Kontosid on kahte liiki: ettevõte ning isik. Neid eristab konto tabelis parameeter tüüp. Loetavuse mõttes on käesolevas töös asendatud konto tabelile viited vastavalt tüübile ehk konto identifikaator on asendatud isiku puhul isiku identifikaator ning ettevõtte puhul ettevõtte identifikaator. Isik on seotud ülesandega läbi tabeli määratud ülesanne.

3.2.2 Ülesanne

Lõputöös kasutatakse kolme liiki ülesandeid: baasülesanne, korduv ülesanne ning juhuslik ülesanne. Kõik ülesande liigid pärinevad ühest tabelist.

Ülesande üldised atribuudid

Ülesande üldised atribuudid kuuluvad kõikidele eelpool mainitud ülesannete liikidele. Nendeks atribuutideks on:

1. Pealkiri
2. Ettevõtte identifikaator
3. Kliendi identifikaator
4. Staatus
5. Olek
6. Loomiskuupäev

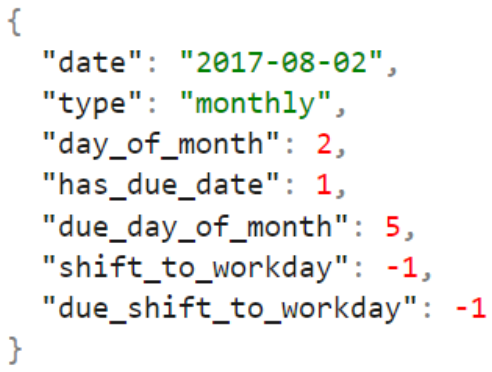
Ülesande pealkiri on tekstiväärtus, mis näitab kasutajale ülesande sisu. Ülesande loomiskuupäev näitab, mis kuupäeval ülesanne süsteemi loodi. Ettevõtte identifikaator on seos ülesande ning ettevõtte vahel, kelle töötaja ülesannet teostab, ning kliendi

identifikaator on seos ülesande ning kliendi ehk ettevõtte vahel, kellele ülesannet teostatakse.

Ülesande staatuseid on kolm: „uus“, „tegemisel“ ja „lõpetatud“. Staatuse „uus“ puhul on tegemist ülesandega, mis on loodud, kuid millega ei ole veel alustatud, staatus „tegemisel“ näitab, et ülesandega on alustatud, kuid see on alles pooleli ning staatus „lõpetatud“ tähendab, et ülesanne on valmis. Olekuid on ülesandel kaks: „kustutatud“ ja „mitte kustutatud“. Lõputöö raames vaadeldakse ainult neid ülesandeid, mis on staatuses „lõpetatud“ ning olekus „mitte kustutatud“.

Baasülesanne

Korduvate ülesannete genereerimiseks kasutatakse baasülesannet, mis on kasutaja poolt varasemalt defineeritud. Lisaks üldistele ülesande parameetritele, on baasülesandel kordumisreegel. Tegemist on JSON objektiga, millel on järgmised parameetrid:



Joonis 3. Baasülesande kordumisreegel

Ülaltoodud reegli puhul on tegemist baasülesande parameetriga, mille abil genereeritakse ülesanne iga kuu 2. kuupäevaks ning mille tähtaeg on kuu 5. kuupäev. Mõlema kuupäeva nihked näitavad, kuidas tuleb kuupäeva korrigeerida vastavalt nädalavahetusele. Joonisel 3 vastavad sellele *shift_to_workday* ja *due_shift_to_workday*. Täpsem nihke loogika on välja toodud allolevas tabelis (vt Lisa 3 – Kordumisreegli nihked).

Tabel 1. Kuupäevade nihete seletus

Nihe	Selgitus
-1	Kui kuupäev langeb nädalavahetusele, siis korrigeeritakse kuupäev tööpäevale, mis eelnes sellele nädalavahetusele. Kui kuupäev langeb nädala sisse, siis nihkel mõju ei ole.
0	Kuupäev jäetakse sellisele kujule, nagu ta on olenemata, kas kuupäev langeb nädalavahetusele või mitte.
1	Kui kuupäev langeb nädalavahetusele, siis korrigeeritakse kuupäev tööpäevale, mis järgneb sellele nädalavahetusele. Kui kuupäev langeb nädala sisse, siis nihkel mõju ei ole.

Korduv ülesanne

Korduv ülesanne on süsteemi poolt kindlale ajale, kindlate atribuutidega ning kindla kliendiga seotud genereeritud ülesanne. Baasülesandest eristab teda kordumisreegli puudumine. Lisaks üldistele ülesande atribuutidele, on korduval ülesandel:

1. Kordumise identifikaator
2. Kordumise kuupäev
3. Ülesande kestus

Kordumise identifikaator on seos korduva ülesande ning baasülesande vahel. Kordumise kuupäev on kuupäev, kuhu baasülesande põhjal ülesanne on ajaliselt paigutatud ning ülesande kestus näitab, mitu sekundit seda ülesannet on tehtud. Lõputöös kasutatakse ainult neid ülesandeid, mis on olekus „lõpetatud“ ning mille kestus on suurem nullist.

Juhuslik ülesanne

Juhuslik ülesanne on kasutaja poolt eesrakenduses loodud ülesanne. Lisaks üldistele ülesande atribuutidele on juhuslikul ülesandel kestus, mis näitab, mitu sekundit seda ülesannet on tehtud. Lõputöös kasutatakse ainult neid ülesandeid, mis on olekus „lõpetatud“ ning mille kestus on suurem nullist. Juhuslikku ülesannet kirjeldavad järgmiste atribuutide puudumine:

1. Kordumise identifikaator

2. Kordumisreegel

3. Kordumise kuupäev

3.2.3 Klient

Kliendiks nimetatakse ettevõtet, kellele Uku kasutajad ülesandeid teevad. Kliendi tabelis on olulisteks atribuutideks:

1. Ettevõtte identifikaator

2. Registrikood

3. Asukoha riik

4. Nimi

5. Staatus

6. Olek

Ettevõtte identifikaator on ettevõtte seos kliendiga, kellele ülesannet teostatakse. Registrikood on kood, mille järgi on võimalik ettevõtet identifitseerida näiteks riiklikes infosüsteemides. Asukohariik kirjeldab, mis riigis ettevõte loodi ning kliendi nime kasutatakse kasutajaga valideerimisprotsessis. Kliendi staatuseid on kaks: „aktiivne“ ja „mitteaktiivne“ ning olekuid samuti kaks: „kustutatud“ ning „mitte kustutatud“. Lõputöös vaadeldakse ainult sellised kliente, kes on staatuses „aktiivne“ ning olekus „mitte kustutatud“.

3.2.4 Ajasisestus

Ajasisestus on tabel, kuhu lisatakse kõik ülesandega seotud ajamõõtmised. Ülesande kogukestus saadakse kõigi temaga seotud ajasisestuste summeerimisel.

Ajasisestuse peamised atribuudid on:

1. Ülesande identifikaator

2. Isiku identifikaator

3. Algus

4. Lõpp
5. Kestus
6. Olek

Ülesande identifikaator on ajasisestuse seos ülesandega, millele aega sisestati ning isiku identifikaator seos isikuga, kes ülesandele aega sisestas. Ajasisestuse algus ning lõpp on kuupäeva ning kellaaega sisaldavad väärtused, mis näitavad ajasisestuse algust ning lõppu. Ajasisestuse kestus näitab sekundites kui pikalt kestis kirjeldatav ajasisestus. Olekuid on ajasisestusel kaks: „kustutatud“ või „mitte kustutatud“. Lõputöös vaadeldakse ainult neid ajasisestusi, mis on olekus „mitte kustutatud“.

3.2.5 Määratud ülesanne

Määratud ülesanne on tabel, mis seob omavahel kasutaja ehk ülesande tegija ning ülesande. Tabeli atribuudiks on ülesande identifikaator ning isiku identifikaator.

3.3 Täiendandmete otsimine

Kuna põhirakenduse andmebaasis ei olnud piisavalt häid andmeid, mille põhjal tuvastada sarnaseid kliente ehk ettevõtteid, siis otsiti käesolevas töös nende kohta täiendavaid lisaandmeid. Andmeid koguti kõigile ligipääsetavast avalikust infosüsteemist e-krediidiinfo [15] ning kasutati ärilisel eesmärgil ehk rakendati masinõppemudeli treenimisel, et ülesannetele ajahinnanguid määrata.

3.3.1 Eelprotsess

Täiendandmete kasutamine lõputöös on eksperiment, et teha kindlaks, kas sarnaste omadustega klientidel ehk ettevõtete vahel leidub mustreid ka raamatupidamis-protsessides. Selleks, et mitte liigselt keskenduda andmete kogumisele, on tehtud kitsendus lisaandmete leidmise protsessis. Selle tulemusel leitakse lisaandmed ainult Eestis loodud ettevõtete kohta, kuna neid on süsteemis kõige rohkem.

Täiendandmetena koguti:

1. Töötajate arv
2. Põhitegevusalad

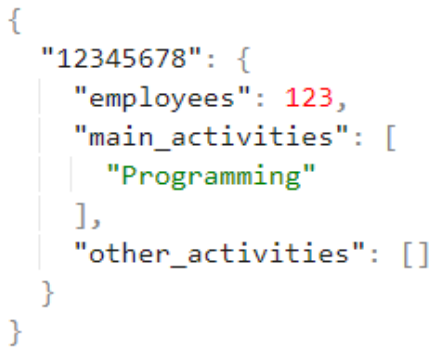
3. Kõrvaltegevusalad

Täiendandmete valikult lähtuti sellest, millised andmed on avalikult kättesaadavad ning millised atribuudid aitaksid tuvastada sarnaseid kliente. Sellest tulevalt kujunes üheks atribuudiks töötajate arv, mis kirjeldab ettevõtte suurust, ning teiseks põhitegevusala, mis kirjeldab ettevõtte tegevusvaldkonda. Hüpoteesiks oli, et sama suurte ettevõtete või sama tegevusvaldkonnaga ettevõtetel on ühiseid jooni ka raamatupidamises, mis aitavad lõputöö kontekstis paremini ülesannete ajahinnanguid ennustada.

3.3.2 Andmete kogumisprotsess

Andmete kogumisprotsessi esimese sammuna leiti põhirakenduse andmebaasist kõik sellised registrikoodid, mis vastasid klientidele, mille asukoha riik oli Eesti. Seejärel salvestati need faili, et neid hiljem kasutada.

Teise sammuna kasutati veebikraapimist [16], ehk andmete kogumist veebilehelt. Loodud funktsionaalsus sai sisendiks kõik registrikoodid ning ühekaupa leidis igale koodile vastavad lisaandmed. Seejärel salvestati leitud andmed JSON formaadis faili, kus võtmeks määrati registrikood ning väärtuseks lisaandmeid sisaldav JSON objekt (vt joonis 4).



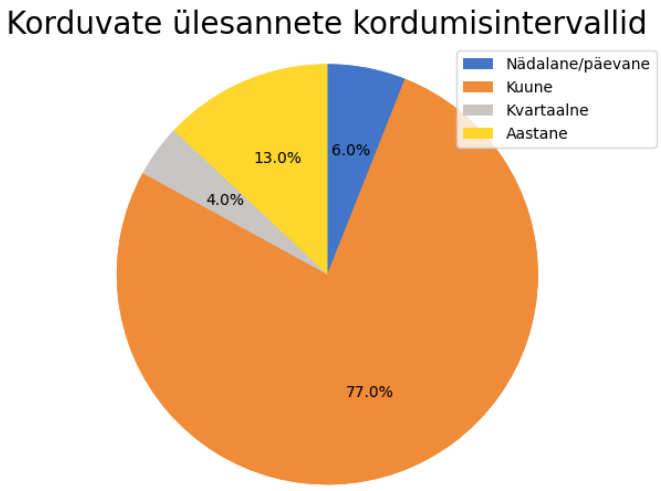
Joonis 4. Ettevõtte lisaandmete objekt

Andmete kogumise käigus selgus, et ettevõtetel on enamasti üks põhitegevusala ning kõrvaltegevusalad puuduvad. Sellest tulenevalt rakendati käesolevas töös lisaandmetest ainult töötajate arvu ning põhitegevusala.

3.4 Statistika

Andmemahtude paremaks mõistmiseks on käesolevas osas välja toodud olulisemad statistilised näitajad. Korduvate ülesannete statistika põhineb ajaperioodil *2020-06-01* kuni *2022-03-01*. Ajaperioodi algus on valitud Uku suurkliendi CH Konsultatsioonid OÜ järgi, kes sel perioodil platvormi kasutamise tõsisemalt ette võtsid. Andmeid on kasutatud käesolevas lõputöös kõigi sel perioodil tegutsevate kasutajate ning ettevõtete kohta.

Uku platvormil on valitud ajaperioodil 882 kasutajat, kellel on vähemalt üks korduv ülesanne. Baasülesandeid on kokku 43 000 ning nendest on määratud ajaperioodiks põhirakenduse poolt genereeritud 550 000 korduvat ülesannet, millest 530 000 ülesande planeeritud algusaeg ei lange kokku tegeliku alustamisajaga ning 117 000 ülesannet on läinud üle tähtaja. Erinevaid ülesande kordumisintervalle on neli: nädalane/päevane, kuine, kvartaalne ning aastane. Statistiliselt jaotuvad baasülesanded nelja kordumisintervalli vahel järgnevalt:



Joonis 5. Korduvate ülesannete jaotus intervallide järgi

Juhuslikke ülesandeid on samal perioodil 480 000, mida kasutatakse eraldi juhuslike ülesannete ajahinnangute mudelis.

4 Tööplaani optimeerimise algoritm

Selles peatükis antakse ülevaade tööplaani optimeerimise algoritmist. Osas 4.1 tuuakse välja ärivajadus, osas 4.2 kirjeldatakse algoritmi tööks kasutatavat andmestikku, osas 4.3 kirjeldatakse tööplaani korrigeerimist vastavalt kasutaja harjumustele, osas 4.4 kirjeldatakse tööplaani ülesannete korrigeerimist tähtaja järgi ning osas 4.5 hinnatakse algoritmi headust testandmete pealt.

4.1 Ärivajadus

Tööhaldustarkvara Uku kasutajate poolt eelseadistatud igakuine tööplaan (ülesannete järjestus) ei ole kooskõlas selle tegeliku täitmisega. Peamisteks põhjusteks on kasutaja enda harjumused ülesandeid teises järjekorras teha või tema klientide harjumused dokumentide esitamisel, mis on aluseks ülesande täitmisele. Sellest tingituna, ei ole kasutajal selget ülevaadet, mis päeval ta mis ülesannetega tegelema peaks.

Hetkel töötab korduva kuupõhise ülesande loomine põhirakenduses nii, et kasutajale pakutakse ülesande paigutamiseks tööplaani automaatselt kuu esimest kuupäeva ning kui kasutaja seda ise ei muuda, siis genereeritaksegi alati ülesanne kuu esimesele kuupäevale. Andmeanalüüsist selgus, et selliseid ülesandeid on päris palju (25%) ning nende tegelik alustusaeg ei lange enamasti kokku süsteemi poolt määratud alustusajaga. Sellest saab järeldada, et nendel juhtudel ei ole kasutaja korduvate ülesannete algusaega oma harjumustele vastavalt seadistanud.

Funktsionaalsuse eesmärgiks on kõigi korduvate ülesannete genereerimiskuupäeva korrigeerimine vastavalt kasutaja harjumustele ehk teisisõnu kordumisreegli korrigeerimine selliselt, et kasutajale kuvatakse ülesannet rakenduses päeval, millal ta kõige suurema tõenäosusega sellega alustab. Lisaks teostatakse teine korrigeerimine lõpptähtaja suhtes, kus vaadeldakse mitmel erineval päeval kasutaja ülesannet tavaliselt teeb ning korrigeeritakse algusaeg selliselt, et kasutaja jõuaks ülesande valmis.

4.2 Andmestik

Käesolevas lõputöös on algoritmi väljatöötamiseks jagatud andmete pärimine eraldi päringuteks. Esmalt päritakse kõik kasutaja identifikaatorid, kellel on vähemalt üks korduv ülesanne, seejärel päritakse kasutaja identifikaatori alusel kõik tema baasülesanded ning siis iga tema baasülesande abil põhirakenduse poolt genereeritud korduvad ülesanded. Selliste eraldi päringute kasutamine kiirendab arendusprotsessi, kuid hiljem algoritmi kasutusele võttes, tuleb andmebaasipäringuid optimeerida.

Käesolevas lõputöös vaadeldakse ainult kuupõhise kordumisega ülesandeid. Põhjus, miks selline kitsendus valiti, tuleneb asjaolust, et nädalase/päevase kordumisega ülesannete optimeerimine ei annaks piisavalt suurt efekti, kvartaalse kordumisega ülesannete osakaal on liiga madal (4%) ning aastapõhiste korduvustega ülesannete optimeerimiseks on andmeid liiga vähe ehk iga baasülesande kohta kõigest üks genereeritud ülesanne. Kuupõhise kordumisega ülesannete osakaal on andmestikus seevastu päris suur (77%) ning igast baasülesandest on määratud perioodiks genereeritud kuni 18 ülesannet, mis on piisav, et kasutaja harjumusi tuvastada.

4.2.1 Kasutajate leidmine

Selleks, et leida andmestikust üles kõik kasutajad, kellel on vähemalt üks korduv ülesanne, tuleb kasutada kolme tabelit: Määratud ülesanne, ülesanne ning konto. Täpsemalt on kirjeldatud nende tabelite sisusid osas 3.2.

Andmestiku kriteeriumid on järgnevad:

1. Kasutajal on seos ülesandega
2. Ülesanne peab omama kordumise identifikaatorit
3. Ülesanne peab olema staatuses „lõpetatud“ ning olekus „mitte kustutatud“
4. Ülesanne ei tohi omada kordumisreeglit
5. Ülesande kordumise kuupäev jääb vahemikku *2020-06-01* kuni *2022-03-01*
6. Ülesanne peab omama ajamõõdet

Ülesandega seotud kriteeriumid garanteerivad korduva ülesande tunnused ning kasutaja seos ülesandega näitab, et selline korduv ülesanne kuulub kasutajale.

4.2.2 Baasülesannete leidmine

Baasülesanne on kasutaja poolt eeldefineeritud ülesandemall, millest kordumisreegli alusel genereeritakse kasutajale süsteemi poolt vastavalt intervallile ülesanne. Andmestiku moodustamiseks kasutatakse kolme tabelit: ülesanne, klient ning määratud ülesanne. Täpsemalt on kirjeldatud nende tabelite sisusid osas 3.2.

Andmestiku kriteeriumid on järgnevad:

1. Ülesanne peab omama kordumisreeglit
2. Ülesande ja kasutaja vahel peab olema seos
3. Ülesanne on olekus „mitte kustutatud“

Ülesande kordumisreegli omamine aitab eristada baasülesandeid teistest ülesannetest. Seos kasutaja ning ülesanded vahel leiab ainult konkreetse kasutaja kohta käivad baasülesanded ning olek filtreerib välja ainult asjakohased baasülesanded.

4.2.3 Korduvate ülesannete leidmine

Korduv ülesanne on baasülesande abil kindla intervalli järel süsteemi poolt loodud ülesanne. Andmestiku moodustamiseks kasutatakse kolme tabelit: ülesanne, klient ning ajasisestus. Täpsemalt on kirjeldatud nende tabelite sisusid osas 3.2.

Andmestiku kriteeriumid on järgnevad:

1. Ülesanne peab omama kordumise identifikaatorit
2. Ülesanne peab olema staatuses „lõpetatud“ ning olekus „mitte kustutatud“
3. Ülesanne ei tohi omada kordumisreeglit
4. Ülesande kordumise kuupäev jääb vahemikku *2020-06-01* kuni *2022-03-01*
5. Ülesanne peab omama ajamõõdet
6. Ajamõõde peab kuuluma kasutajale ning peab olema olekus „mitte kustutatud“

7. Klient peab olema staatuses „aktiivne“ ning olekus „mitte kustutatud“

kordumisreegli puudumine ning kordumise identifikaatori olemasolu aitab eristada korduvaid ülesanded nii baas- kui ka juhuslikest ülesannetest. Ülesande staatuse ning oleku abil leitakse kõik asjakohased ja aktiivsed ülesanded. Ajamõõte olemasolu ja kuuluvus kasutajale abil leitakse kõik sellised ülesanded, mille alguaeg, lõpp, kestus ning tegija on teada. Kordumise kuupäev näitab, mis kuupäevale süsteem ülesande genereeris ning kliendi filtreerimine staatuse ja oleku järgi garanteerib, et leitakse ülesanded, mille klient on aktiivne.

Andmestiku ajaline piiramine tuleneb asjaolus, et määratud perioodil alustas Uku suurklient CH Konsultatsioonid OÜ tõsisemalt platvormi kasutamisega ning kuna nad on ettevõtte koostööpartnerid ka valideerimise protsessis, siis on nende järgi valitud ajaperiood, et nende andmed oleksid võimalikult kvaliteetsed.

4.3 Kasutaja harjumustest lähtuv tööplaan

Selleks, et luua kasutaja harjumustest lähtuv tööplaan, tuleb vastavalt kasutaja identifikaatorile leida kõik tema baasülesanded. Seejärel leitakse kõik baasülesandest süsteemi poolt genereeritud korduvad ülesanded, mille genereerimisel kasutatakse baasülesande küljes olevat kordumisreeglit. Selles on info ülesande algukuupäeva ning tähtaja kohta (vt Lisa 4 – Korrigeerimisprotsess).

Kordumisreegli korrigeerimiseks on seatud mõned kriteeriumid, mille abil saab kindlaks teha, millist reeglit korrigeerida ja millist mitte. Korrigeeritud kordumisreegel leitakse ainult siis, kui sellest reeglist on genereeritud rohkem kui kolm ülesannet. Vähemate ülesannete korral on oht, et kasutaja harjumus ei ole veel välja kujunenud ning reegel korrigeeritakse juhuslikult. Korduvate ülesannete puhul filtreeritakse ka müra välja ehk kui korduva ülesandega on alustatud rohkem kui 30 päeva varem või 30 päeva hiljem, siis loetakse sellist korduvat ülesannet müraks, mida ei arvestata kasutaja harjumuse tuvastamisel.

4.3.1 Uue reegli loomine vastavalt harjumustele

Uue reegli loomise puhul on tegemist lähenemisega, kus olemasolevat reeglit ei kasutata ning uus reegel luuakse puhtalt kasutaja ajasisestuste põhjal. Kuna ülesandega

alustamisajad olid kuude lõikes väga varieeruvad, siis kerkisid õhku mitmed probleemid, mida kas ei olnud võimalik või oli keeruline lahendada.

Algusaja ning ülesande sisu probleem

Esimeseks probleemiks oli see, et kuupäev ei peegeldanud, mis ülesannet tegelikult tehti. See tähendab, et kui näiteks ülesandega alustati 5. jaanuaril, siis ei olnud teada, kas tegemist on detsembrikuu ülesandega, mis tehti hiljem, jaanuarikuu ülesandega, mis tehti õigel ajal, või veebruarikuu ülesandega, mis tehti varem. Ehk teisisõnu ülesande alustamiskuu ning ülesande sisu vahel ei olnud võimalik tuvastada seost.

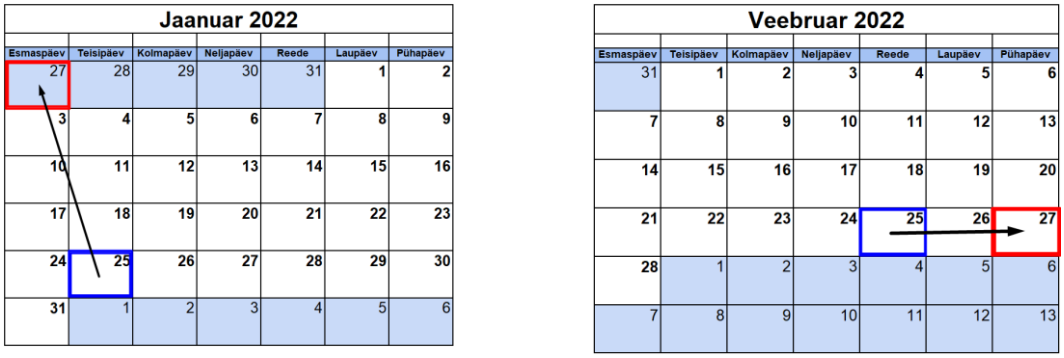
Selleks et probleemi lahendada, tuli kasutusele võtta ülesande küljes olev atribuut kordumise kuupäev, mis näitab, kuhu paigutati ülesanne süsteemi poolt. Seega on võimalik selle järgi kindlaks teha, mis kuu ülesandega on tegemist. Oluline on siinkohal mainida, et selleks, et uue reegli loomine ei oleks seotud olemasoleva reegliga, on kordumise kuupäevast kasutada ainult kuud. See aitab vältida olemasoleva reegli kasutamist uue kordumisreegli loomisel.

Sobivama kuupäeva valimise probleem

Teine probleem, mis õhku kerkis, oli sobivama kuupäeva valimise etapis. Selleks, et leida statistiliselt parim kuupäev ülesandega alustamiseks, tuli koostada nö statistika ühe baasülesande poolt loodud kõigi korduvate ülesannete algusaja kohta.

Selline lähenemine peidab endas tegelikult probleemi, mis kallutab tublisti algoritmi poolt väljapakutavat kuupäeva. Nimelt sama alguskuupäevaga ülesanded ei pruugi kirjeldada harjumust. Ütleme, et teatud korduvat ülesannet genereeritakse süsteemi poolt iga kuu 25-daks kuupäevaks ning selle ülesandega alustati kahel korral 27-ndal. Selle põhjal võiks järeldada, et uus reegel peaks paigutama ülesande 27-ndale kuupäevale.

Joonisel 6 on näha, et jaanuari ülesandega alustati tegelikult 29 päeva varem ehk 27-ndal detsembril ning veebruari ülesannet tehti 2 päeva hiljem ehk 27-ndal veebruaril. Kuupäevad langevad küll kokku, aga tegelikult puudub kahe ülesande tegemise vahel kindel harjumus. Oluline on veel siinkohal mainida, et kui olemasolevat reeglit mitte kasutada, siis tegelikult ei ole teada, et ülesanne genereeriti süsteemi poolt 25-ndale kuupäevale. Sellisel juhul ei ole võimalik probleemi olemust tuvastada.



Joonis 6. Jaanuari- ja veebruarikuu ülesande alustamise päeva võrdlus genereerimise kuupäevaga

Tegemist on probleemiga, millele ei ole head lahendust, kuna ülesanne ei ole piiratud ühe kuu lõikes ehk jaanuarikuu ülesannet võidakse teha nii detsembris, jaanuaris kui ka veebruaris, seega ühe ülesande kontekstis tuleb vaadata suisa kolme kuist ajaperioodi.

Nädalavahetusele langeva ülesande probleem

Kolmandaks probleemiks on ülesande nihutamine nädalavahetuste suhtes. Kuna baasülesandes olevas reeglis on defineeritud, kuidas tuleb nihutada ülesannet vaadeldavas kuus, kui see genereerimise tulemusel langeb nädalavahetusele. Täpsemalt on nihetest juttu alamosas 3.2.2. Nihete teadmise puudumine ehk selle teadmise mittekasutamine olemasolevast reeglist, tõstab omakorda algoritmi keerukust, kuna nende tuvastamine puhtalt alustamiskuupäevade põhjal on keeruline, sest andmed on väga volatiilsed ning sellise detailsuse kindlaks tegemiseks on keskmiselt ühe baasülesande kohta korduvaid ülesandeid liiga vähe.

Kokkuvõte

Kuna uue reegli loomisel tekkisid mitmed probleemid, millest osasid ei olnud võimalik, kas andmete vähesuse või volatiilsuse tõttu lahendada, siis sai otsustatud selle lahendusviisiga mitte jätkata ning töötada välja uus lahendusviis, millega on võimalik eespool kirjeldatud probleeme vältida.

4.3.2 Olemasoleva reegli korrigeerimine vastavalt harjumustele

Teiseks lähenemisviisiks on olemasoleva kordumisreegli korrigeerimine, mis tähendab, et kasutatakse kõiki varasemalt kasutaja poolt määratud kordumisreegli parameetreid ning leitakse olemasoleva reegli kõrvale korrigeeritud reegel.

Selleks, et leida seos ülesande algusaja ning sisu vahel ehk kas 5. jaanuaril tehakse detsembri, jaanuari või veebruari ülesannet, kasutatakse kordumise kuupäevas olevat kuud, mille alusel saab eeldada, mis kuu ülesandega on parasjagu tegemist. Ülesande nihutamiseks nädalavahetuse suhtes, kasutatakse olemasolevas kordumisreeglis defineeritud nihet, mis näitab, kuidas tuleb ülesanne ümber paigutada, kui see langeb nädalavahetusele. Täpsemalt juttu alamosas 3.2.2.

Uue reegli loomisel tekkis probleem parema ülesande alustuskuupäeva leidmisega, kus kaks ühte ja sama kuupäeva ei pruukinud peegeldada tegelikku harjumust. Olemasoleva reegli korrigeerimisel, on võimalik alustuskuupäev leida nihete teel. See tähendab, et korduvate ülesannete algusaega ei kasutata mitte kuupäevana, vaid leitakse päevades nihe tegeliku algusaja ning süsteemi poolt välja pakutava algusaja vahel. Täpsemalt sobiliku nihke leidmisest on juttu alamosas 4.3.3 Sobiva nihke leidmine.

Olemasoleva kordumisreegli korrigeerimise lähenemist kasutades, ei tekkinud ületamatuid probleeme ning seda lahendusviisi kasutatakse lõputöös, et kasutajale vastavalt harjumustele ülesandeid paremini tööplaani paigutada.

4.3.3 Sobiva nihke leidmine

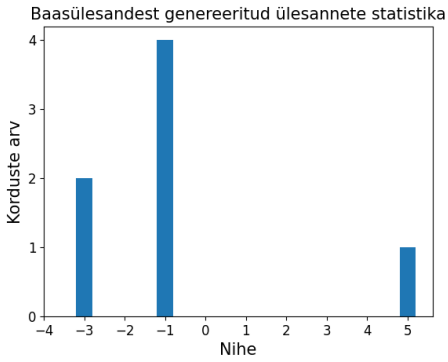
Igale baasülesandele vastab teatud hulk kindla intervalliga, kindlale kliendile korduvaid ülesandeid. Ühe baasülesande poolt genereeritud korduvad ülesanded on oma sisult samad, kuid teostatud erinevatel ajaperioodidel. Kordumisreegli alusel paigutab süsteem ülesande kasutaja tööplaani ehk määrab algusaja. Selleks, et leida kasutaja harjumustele vastav sobilikum algusaeg, millal kasutaja tegelikud selle ülesandega alustab, kasutatakse nihke meetodit.

Nihete kaudu sobiliku kuupäeva leidmise eeliseks on see, et nihetest moodustub arvtelg, kus negatiivne väärtus ehk negatiivne nihe näitab mitu päeva varem ning positiivne nihe mitu päeva hiljem, alustati tegelikult ülesandega. Nihe leitakse järgneva valemi abil:

$$nihe = \text{ü}lesandega\_alustamise\_kuup\text{ä}ev\ - genereerimise\_kuup\text{ä}ev$$

Näiteks, kui süsteemi poolt genereeritud ülesande alustamiskuupäevaks on *25.01* ning tegelikult alustati selle ülesandega *22.01*, siis vastav nihe on -3 ehk ülesandega alustati 3 päeva varem. Nii leitakse kõigi baasülesandest genereeritud korduvate ülesannete nihked

tegeliku algusaja suhtes ja saadakse statistika, mis kirjeldab, mitu korda vastavat nihet baasülesande poolt loodud korduvate ülesannete ja tegeliku alguaja vahel tekkis.



Joonis 7. Baasülesandest genereeritud korduvate ülesannete statistika

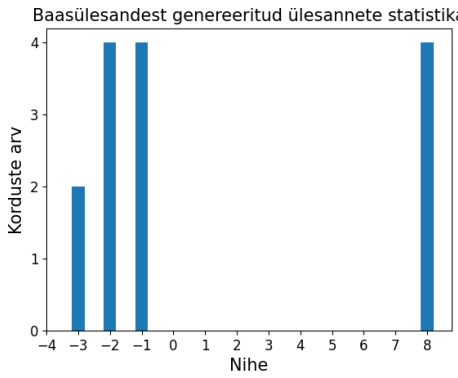
Joonis 7 näitab, et nihkeid väärtusega -3 esineb 2 korda, -1-te esineb 4 ning 5-te esineb 1 kord. Selleks, et teha kindlaks sobivaim nihe, mille alusel kordumisreeglit korrigeerida, on loodud kolm erinevat algoritmi: maksimaalse nihke meetod, akna meetod ning kaaludega akna meetod.

Maksimaalse nihke meetod

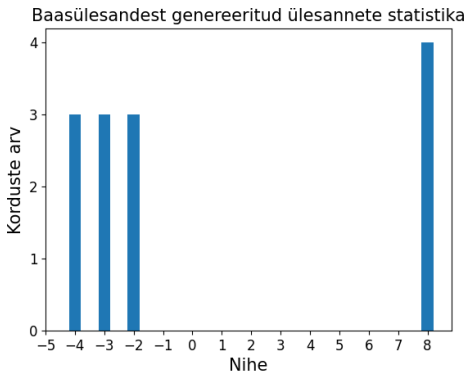
Tegemist on kõige lihtsama meetodiga, kus vaadeldakse milline nihe esines statistikas kõige rohkem ning see ongi kõige parem nihe kordumisreegli korrigeerimiseks. Sellise lähenemise suurimaks miinuseks on olukord, kus maksimaalseid nihkeid on mitu, sest siis on keeruline parimat nendest välja valida.

Joonis 8 puhul on keeruline otsustada, kas parim nihe on -2, -1 või hoopis 8, kuna neil kõigil on korduste arv 4. Saab küll otsustada, et alati valitakse väikseim nihe maksimaalsete hulgast ehk -2, kuna kindlam on paigutada ülesanne ajaliselt ettepoole kui viia kaugemale.

Teiseks probleemiks on olukord, kus nihked on väga ebaühtlased, mida kirjeldab joonis 9. Selle näite puhul otsustaks algoritm, et parim nihe on +8, kuid kui vaadata andmetesse, siis nihked -4, -3 ja -2 on üksteisele väga lähestikku ning liites nende kordused kokku saadakse 9, mis tähendab, et 9-l juhul on ülesanne tehtud keskmiselt 3 päeva varem ning 4-jal juhul 8 päeva hiljem. See tähendab, et ülesannet tuleks tegelikult paigutada ajaliselt hoopis ettepoole, mida aga selline meetod ei suuda tuvastada.



Joonis 8. Maksimaalse nihke meetodi sobiliku nihke valiku probleem



Joonis 9. Maksimaalse nihke meetodi ebaühtlaste ülesannete probleem

Akna meetod

Tegemist on meetodiga, kus mitte ei vaadelda ainult kindla nihke väärtust, vaid vaadeldakse ka selle läheduses olevate nihete väärtuseid ning iga nihke kohta leitakse vastav skoor. Parimaks nihkeks osutub see, millele vastav skoor on kõige suurem. Skoori leidmiseks kasutatakse järgnevat valemit, kus x tähistab vaadeldava nihke väärtust ning *f(x)* on funktsioon, mis leiab vastavalt nihke väärtusele korduste arvu.

$$skoor = \sum_{i=x-2}^{x+2} f(i)$$

Skoori leidmiseks liidetakse vadeldava nihke kordustele juurde kaks nihet vasakul ning kaks nihet paremal olevate nihete kordused, mida kirjeldab välja toodud skoori valem. Teisisõnu, kui vaadeldavaks nihkeks on -1 ehk $x = -1$, siis summeritakse korduste arvud, kus x väärtused kuuluvad lõikku [-3;1]. Skoori leitakse ainult selliste nihete jaoks, millel leidub korduste arv statistikas.

Joonisel 10 arvutatakse skoor viiele nihkele: -4, -2, -1, 0 ja 8, millest siis parim välja valitakse. Nihke -4 skoori arvutamisel vaadeldakse kaks nihet temast vasakul olevaid (-5 ja -6) ning kaks nihet paremal olevaid (-3 ja -2) ning summeeritakse vastavad kordused kokku. Analoogselt leitakse ka -2, -1, 0 ning 8 skoorid

$$skoor\_neg\_4 = 0 + 0 + 2 + 0 + 4 = 6$$

$$skoor\_neg\_2 = 2 + 0 + 4 + 4 + 2 = 12$$

$$skoor\_neg\_1 = 0 + 4 + 4 + 2 + 0 = 10$$

$$skoor\_0 = 4 + 4 + 2 + 0 + 0 = 10$$

$$skoor\_8 = 0 + 0 + 4 + 0 + 0 = 4$$

Leitud skooridest parim nihe on -2, kuna selle skoor on kõige suurim.

Selle meetodi plussiks on see, et vaadeldakse üksteisele lähedal olevaid nihkeid koos. Lisaks muutub väiksemaks tõenäosus, et tekib sama maksimaalse skooriga nihkeid. Ometi ei ole see välistatud, kuid nende tekkimisel, saab eelistada nihet, mille omaväärtus on suurim ning kui ka need langevad kokku tehakse valik väikseima nihe kasuks, kuna kindlam on tuua paigutada ülesanne ajaliselt ettepoole kui viia kaugemale.

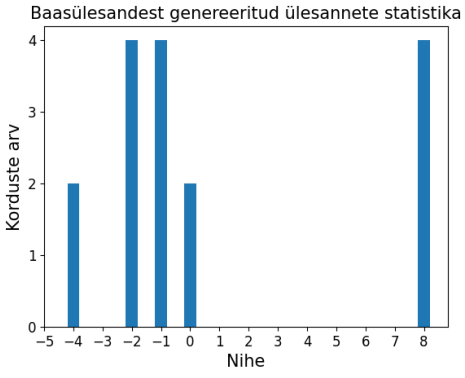
Teiseks probleemiks on üksteisele distantsilt lähedaste nihete kogumiku puhul keskmiste nihete ebaaus eelistamine äärtele joonisel 11. Peale vaadates tundub, et parim nihe on -4, kuna selle omaväärtus on suurim, kuid kui arvutada iga nihkele vastav skoor, siis selgub, et tegelikult selle meetodi järgi on suurima skooriga nihked -3 ja -2. Põhjuseks on see, et äärmistel nihetel (-4 ja -1) on skoori leidmisel alati vähem nullist suuremaid liidetavaid, kui keskmistel nihetel (-3 ja -2), seega on keskmistel nihetel alati suurem skoor kui äärtel.

$$skoor\_neg\_4 = 0 + 0 + 10 + 2 + 2 = 14$$

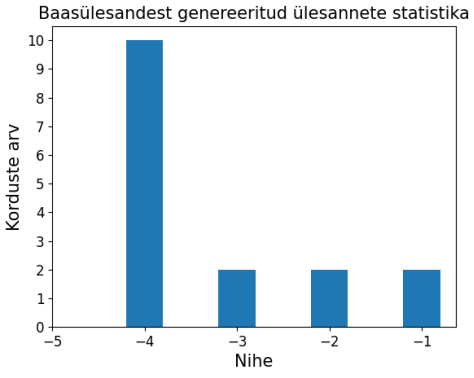
$$skoor\_neg\_3 = 0 + 10 + 2 + 2 + 2 = 16$$

$$skoor\_neg\_2 = 10 + 2 + 2 + 2 + 0 = 16$$

$$skoor\_neg\_1 = 2 + 2 + 2 + 0 + 0 = 6$$



Joonis 10. Akna meetodiga mitme maksimaalse kordusega sobiliku nihke leidmine

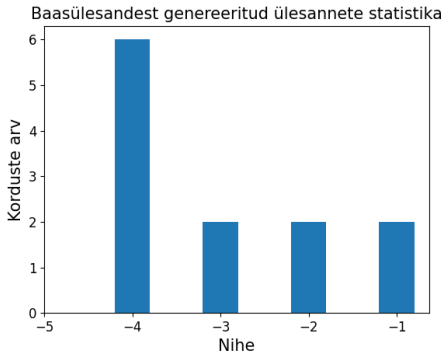


Joonis 11. Akna meetodi ebaaus keskmiste väärtuste eelistamine

Kaaludega akna meetod

Tegemist on Aknaga meetodi edasiarendusega, milleks on võetud kasutusele kaalud, mis aitavad eelistada skoori arvutamisel vaadeldava nihke korduste arvu ning vähem eelistada vaadeldava nihke läheduses olevate nihete korduseid.

Valitud kaaludeks on määratud vaadeldava nihke jaoks 1.25 ning läheduses olevate nihete jaoks 0.8. Eesmärk, on lahendada probleem, kus keskmiseid nihkeid eelistatakse äärmistele. Kaalud on valitud katseeksitusmeetodil, kus prooviti lahendada sarnaseid olukordi päris andmetel, mis esines joonisel 11 ning selle tulemusel kujunesid välja kaalud 1.25 ning 0.8.



Joonis 12. Kaaludega akna meetodi abil äärte probleemi lahendamine

Joonisel 12 statistika põhjal saab kaaludega akna meetodi abil leida skoorid alloleva valemi alusel, kus muutuja *f(x)* on mingi vaadeldav funktsioon, mis saab sisendiks nihke ning tagastab sellele nihkele vastava korduste arvu. Argument x-i puhul on tegemist vaadeldava nihkega.

$$skoor = 0.8 * \sum_{i=x-2}^{x-1} f(i) + 1.25 * f(x) + \ 0.8 * \sum_{i=x+1}^{x+2} f(i)$$

Seega ülal kirjeldatud valemi põhjal saab leida järgmised skoorid:

$$skoor\_neg\_4 = 0.8 * (0 + 0) + 1.25 * 6 + 0.8 * (2 + 2) = 0 + 7.5 + 3.2 = 10.7$$

$$skoor\_neg\_3 = 0.8 * (0 + 6) + 1.25 * 2 + 0.8 * (2 + 2) = 4.8 + 2.5 + 3.2 = 10.5$$

$$skoor\_neg\_2 = 0.8 * (6 + 2) + 1.25 * 2 + 0.8 * (2 + 0) = 6.4 + 2.5 + 1.6 = 10.5$$

$$skoor\_neg\_1 = 0.8 * (2 + 2) + 1.25 * 2 + 0.8 * (0 + 0) = 3.2 + 2.5 + 0 = 5.7$$

Skooridest järeldub, et parimaks nihkeks on -4. Kui leidub statistikas rohkem kui üks nihe, mille skoor on maksimaalne, siis parimaks osutub see nihe, mille enda korduste arv on suurim. Kui ka see on sama, valitakse nende hulgast arvuliselt väikseim nihe.

4.4 Lõputähtajaga korrigeeritud tööplaan

Tegemist on korrigeeritud reegli üle korrigeerimisega lõputähtaja suhtes. Kui esimese sammuna korrigeeriti reeglit üksnes kasutaja harjumuste põhjal, siis nüüd korrigeeritakse leitud reeglit vastavalt lõpptähtajale. Lõpptähtaja järgi korrigeerimise idee seisneb selles, et kui kasutaja harjumuse kohaselt on ülesandega alustamise kuupäev selline, et ta ei jõua tähtajaks ülesannet valmis, siis tuuakse alustamiskuupäeva ettepoole. Põhjus, miks lõputähtaja järgi korrigeerimine toimub eraldi sammuna, tuleneb sellest, et vajadusel kui tekib korduvaid ülesandeid, kus pole lõpptähtaeg oluline, saaks lõputähtaja järgi korrigeerimise välja jätta.

Korrigeerimiseks kasutatakse kasutaja harjumust selle kohta, mitmel erineval päeval vaadeldavat ülesannet keskmiselt tehakse, sellest tulenevalt saab arvutada kuupäeva, millal hiljemalt ülesandega alustada tuleks, et sellega õigeks ajaks valmis jõuda. Tõeväärtus, mille abil saab leida, kas ülesannet tuleks üle korrigeerida on järgmine:

$$uue\_reegli\_alguskuupäev + erinevate\_päevade\_arv - 1 > lõpptähtaeg$$

Lahutustehe -1 tuleneb sellest, et ka lõpptähtaja päeval on võimalik kasutajal ülesannet teha, sest lõpptähtaeg on alati määratud kellaajaliselt päeva lõppu. Kuupäeva korrigeerimine käib analoogselt eelmisele tõeväärtusele ja on kirjeldatud järgmise valemiga:

$$uus\_alguskuupäev = lõpptähtaeg - erinevate\_päevade\_arv + 1$$

Lõpptähtajale 1 juurde liitmine tähendab, et ka lõpptähtaja kuupäeval on võimalik ülesandega tegeleda. Näiteks oletame, et harjumuse järgi paigutaks süsteem uue reegli järgi ülesande 15-ndale kuupäevale ning lõpptähtaeg on 17-ndal kuupäeval. Kuna keskmiselt tegeleb kasutaja selle ülesandega viiel erineval päeval, siis $15 + 5 \geq 17$, seega sellist ülesannet tuleb korrigeerida ning uueks alguskuupäevaks saadakse $17 - 5 + 1 = 13$.

Oluliseks piiranguks, mis lõputöö raames on seatud, on korrigeerimine kuu lõikes ehk korrigeerimine ei tohi minna teise kuusse võrreldes ülesande sisuga ehk jaanuarikuu ülesannet ei tohi korrigeerida detsembrisse ega veebruarisse. Selle piirangu eesmärk on vältida segaduse tekitamist, kus kasutajal ei ole ülevaadet kuulõikes vajalikest töödest.

Näiteks harjumuse järgi on ülesanne paigutatud veebruarikuu 3-ndale kuupäevale ning tähtaeg on 4-ndal. Kasutaja on harjunud ülesannet tegema viiel erineval päeval. Valemi järgi tuleks selline ülesanne paigutada eelmise kuu ehk jaanuarikuu viimasele päevale, kuid selle tulemusel liiguks veebruari ülesanne eelmisesse kuusse, mis ei ole lubatud. Seega paigutatakse ülesanne veebruari kuu 1-sele kuupäevale.

4.5 Algoritmi headuse hindamine testandmete pealt

Peatükis 4 on olnud juttu kahest kordumisreegli korrigeerimise algoritmist. Esimene neist on kirjeldatud osas 4.3 Kasutaja harjumustest lähtuv tööplaan ning teine osas 4.4 Lõputähtajaga korrigeeritud tööplaan. Testandmete pealt on hinnatud ainult kasutaja harjumustest lähtuvat algoritmi, kuna tegemist on põhialgoritmiga tööplaani optimeerimises. Lisaalgoritmi ehk lõpptähtaja järgi korrigeerimist ei hinnatud testandmete pealt, kuna selleks puudus andmestikus vastav info ehk ei olnud täpselt teada, kas hilinemine oli tingitud sellest, et raamatupidaja tegi ise ülesannet liiga hilja või tema kliendist, kes hilines dokumentide esitamisega, mistõttu oli raamatupidaja sunnitud alustama ülesandega hiljem.

Algoritmi loomisel on kasutatud andmeid perioodil *2020-06-01* kuni *2021-12-01* ning perioodi *2021-12-01* kuni *2022-03-01* on kasutatud testimiseks. Algoritmi headuse leidmiseks arvutatakse nii vana reegli kui korrigeeritud reegli distants ehk päevade arv tegelikust esimesest ülesande ajasisestusest. Distantsi leidmine on kirjeldatud valemiga:

$$distants = |planeeritud\_kuupäev - tegelik\_kuupäev|$$

Seejärel leitakse kui suur viga ehk distants oli keskmiselt ülesandel nii enne kui ka pärast algoritmi rakendamist. Enne algoritmi rakendamist oli ülesande keskmine distants planeeritud alustusaja ning tegeliku alustusaja vahel 9.1 päeva ning peale algoritmi rakendamist 4.4 päeva. See näitab, et bakalaureusetöös valminud algoritm suudab tuvastada kasutaja harjumusi ning nendest lähtuvalt planeerida ülesandeid tööplaani paremini.

5 Ülesande ajahinnangu ennustamine

Käesolev peatükk räägib ülesande ajahinnangu ennustamise ärivajadusest, andmestikust, korduvate ning juhuslike ülesannete mudelitest ning Gensim tekstimudelist ja mudeli täpsuse hindamisest.

5.1 Ärivajadus

Algselt oli plaanis rakendada ülesannete ajahinnanguid tööplaani optimeerimise protsessis, kuid kuna selline lähenemine ei läinud Uku kasutajate vajadustega kokku, siis sai ajahinnangute ennustamine loodud selleks, et tulevikus hakata prognoosima kasutajate töökoormust.

Ettevõttes ei ole varem ajahinnangute ennustamise funktsionaalsust analüüsitud, veel vähem realiseeritud, seega puudub ettevõttel nõue mudeli täpsusele. Põhiliseks eesmärgiks on lõputöö raames analüüsida andmeid, katsetada ning astuda esimesi samme selles suunas, et iga ülesanne saaks endale ajahinnangu.

Põhjus, miks ajahinnanguid tuleks süsteemi poolt luua, on kasutajate käitumismustrid, mis on näidanud, et nad kas ei oska hinnata ülesande kestust või ei soovi ajahinnanguid ise sisestada. Ilma ajahinnanguteta pole aga võimalik tulevikus kasutaja töökoormust prognoosida.

5.2 Juhusliku ja korduva ülesande ajahinnangu ennustamine

Alguses loodi ainult üks ülesande ajahinnangumudel, mis ennustas nii juhuliku kui ka korduva ülesande ajahinnangut. Analüüsides korduvaid ülesandeid selgus, et ülesande kestusel on ühe baasülesande lõikes teatuid mustreid, millest kasvas välja hüpotees, et korduvate ülesannete kestust peab saama täpsemalt ennustada kui juhuslikke ülesandeid. Sellest tulenevalt sai hüpoteesi kontrollitud ning loodud kaks ajahinnangumudelit. Üks juhuslike ning teine korduvate ülesannete jaoks.

Korduvate ülesannete puhul on oluline, et mudel suudaks leida tugevaid seoseid baasülesande põhjal genereeritud korduvate ülesannete vahel ning nõrgemaid seoseid teiste atribuutidega. Juhuslike ülesannete puhul on oluline, et mudel leiaks seoseid ülesande pealkirjade ning erinevate klientide omaduste vahel. See tähendab, et juhuslike ülesannete puhul peab üldistusvõime olema suurem, kui korduvate ülesannete puhul. Seega annab mõlema ülesande tüübi jaoks eraldi mudeli loomine võimaluse valida atribuute vastavalt mudelile ning seeläbi mõjutada mudeli üldistusvõimet.

5.3 Andmestiku vektoriks teisendamise põhimõtted

Selleks, et masinõppemudel saaks andmetest aru ning suudaks leida mustreid ja seoseid nende vahel, tuleb iga andmeobjekti atribuut teisendada kas üheks või mitmeks vektorielemendiks. Seejärel saab nendest elementidest moodustada tervikvektori. Vektori paremaks mõistmiseks, saab seda seletada kui andmete konverteerimist masinõppe jaoks arusaadavasse keelde.

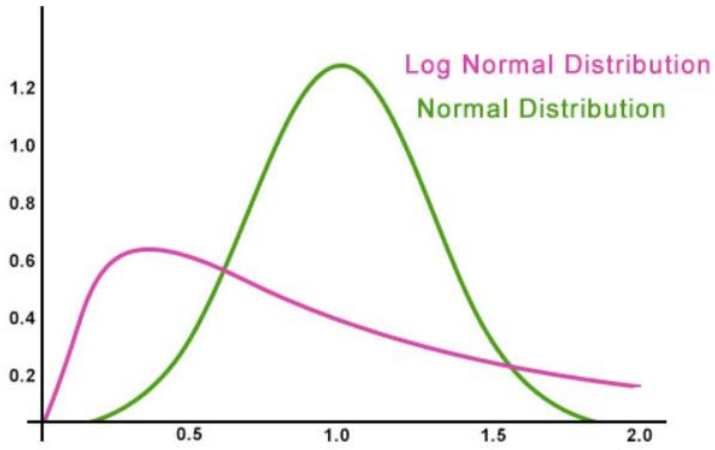
5.3.1 Andmeliigi järgi vektoriks konverteerimine

Andmete konverteerimisel mängib olulist rolli andmeliik, mille põhjal kujuneb sellest kas ühe või mitme vektori elemendiga massiiv. Andmeid on kahte liiki: nominaal- ja ordinaalandmed. Nominaalandmete puhul puudub andmete vahel järjestus. Näiteks klienti kirjeldav identifikaator, kus identifikaator väärtusega 1 ning väärtusega 2 vahel puudub sisuline seos, kuna identifikaator on arv, mis kirjeldab konkreetset objekti. Ordinaalandmed on sellised andmed, millel on kindel järjestus, kuid astmete vahe selles järjestuses võib olla ebaühtlane. Näiteks oletame, et leidub kaks ülesannet, mille kestused on vastavalt 5 minuti ning 10 minutit. Sellisel juhul nende ülesannete vahel on sisuline seos, kuna 7.5 minutit jääb 5 ja 10 minuti vahele.

5.3.2 Andmete standardiseerimine koos logaritmiga

Andmete standardiseerimine on arvuliste väärtuste viimine vahemikku -1 kuni 1-ni. See on vajalik, et treenida regressioonimudelit, sest vastasel korral võivad standardiseerimata andmed tekitada mudelisse müra.

Logaritmi kasutus tuleneb sellest, et ajakulu jaotus andmestikus on lähedasem logaritmilisele normaaljaotusele kui tavalisele normaaljaotusele



Joonis 13. Logaritmiline normaaljaotus ning normaaljaotus [17]

Kogu standardiseerimise protsess koos logaritmiga näeb välja järgmine: olgu meil atribuut, mille minimaalne väärtus on 100 ja maksimaalne väärtus 10000. Esmalt võetakse mõlemast äärmusest logaritm ning saadakse 10 ja 100. nüüd viiakse mõlemad väärtused vahemikku -1 kuni 1-ni, see tähendab, et -1-le vastab 10 ning 1-le vastab 100. Seega kui soovime atribuuti standardiseerida, siis omistab see edaspidi väärtuse -1 ja 1 vahel. Valemina näeb see välja järgmine:

$$standardiseeritud\_väärtus = 2 * \frac{(\log otsitav\_väärtus - \log minimaalne\_väärtus)}{(\log maksimaalne\_väärtus - \log minimaalne\_väärtus)} - 1$$

Minimaalsed ning maksimaalsed väärtused salvestatakse andmebaasi Redis, et neid oleks võimalik kasutada vektori koostamisel ennustamisprotsessis.

5.4 Modelleerimine

Modelleerimise etapis treenitakse nii juhuslike kui korduvate ülesannete jaoks viis regressiooni mudelit. Nendeks mudeliteks on: LassoLars [18], Lasso [19], Ridge [20], KNeighborsRegressor [21] ning Lineaarne regressioon [22].

Lineaarse regressiooni puhul kasutati hüperparameetritena vaikeväärtuseid ning LassoLars, Lasso ning Ridge puhul määrati mudelite hüperparameetriks: $alpha = 0.00001$

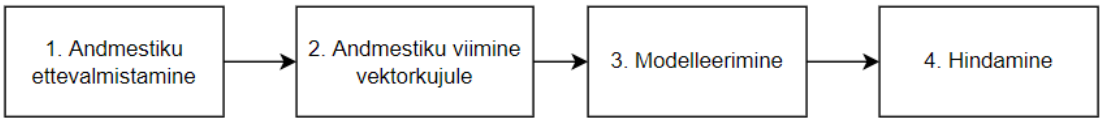
Alpha on reguleerimistermini parameeter ehk karistustermin. Sobilik alpha väärtus leiti katseeksitusmeetodi teel, mil prooviti väärtuseid 1-st kuni 0.00001-ni. Mida suurem oli

alpha, seda paremini ennustati lühikesi kestusega ülesandeid, kuid seda halvemini pika kestusega ülesandeid. Valitud alpha on kesktee lühikeste ning pikkade kestustega ülesannete vahel [23].

KNeighborsRegressor puhul on kasutatud hüperparameetrit $n\_neighbours = 7$, mis tähendab, et ennustatav väärtus leitakse seitsme lähima naabervektori põhjal [21].

5.5 Juhuslike ülesannete ajahinnangu mudel

Juhuslike ülesannete ajahinnangu mudeli loomine toimub neljas etapis.



Joonis 14. Juhuslike ülesannete ajahinnangu mudeli loomise etapid

Kõigepealt valmistatakse ette andmestik, seejärel viiakse andmed vektorkujule. Sellele järgneb andmete modelleerimine ehk mudelite loomine ning siis hinnatakse mudelite headust ning valitakse välja parim.

5.5.1 Andmestiku ettevalmistamine

Juhuslike ülesannete ajahinnangu mudeli treenimiseks kasutatav andmestik moodustatakse kahes etapis. Esmalt filtreeritakse andmed põhirakenduse andmebaasitasandil ning seejärel kooditasandil. Seejärel toimub andmete jagamine treening- ning testandmeteks.

5.5.1.1 Andmestiku moodustamine andmebaasi päringuga

Andmestiku moodustamiseks kasutatakse kolme tabelit: ülesanne, klient ning määratud ülesanne. Täpsemalt on kirjeldatud nende tabelite sisusid osas 3.2.

Ülesande kriteeriumid

Ülesanne peab vastama järgmistele kriteeriumitele:

1. Peab puuduma kordumine
2. Peab olema ajavahemikus *2021-01-01* kuni *2022-03-01*

3. Peab olema staatuses „lõpetatud“ ning olekus „mitte kustutatud“

4. Peab omama kestust

Kordumise puudumine tähendab seda, et ülesandel puudub nii kordumise identifikaator kui ka kordumisreegel. Andmestikku on piiratud ajaliselt, kuna andmemahtude suurendamisel puudub efekt mudeli täpsusele, millest on täpsemalt kirjutatud alamosas 5.5.4 Hindamine. Ülesande staatuse, oleku ning ülesande kestuse kriteeriumid on vaikimisi määratud filtreerimistingimused, mis igas päringus peavad eksisteerima, et asjakohaseid andmed saada.

Kliendi kriteeriumid

Klient peab vastama järgmistele kriteeriumitele:

1. Peab omama registrikoodi

2. Peab asuma Eestis

3. Peab olema staatuses „aktiivne“ ning olekus „mitte kustutatud“

Andmestikku on kitsendatud registrikoodi olemasoluga, kuna mudeli treenimiseks on vajalikud kliendi kohta käivad lisaandmed, mida on võimalik leida ainult registrikoodi alusel. Lisaandmetest on pikemalt juttu osas 3.3 Täiendandmete otsimine. Asukohariigi järgi filtreerimine tuleneb asjaolust, et lisaandmed on hetkel ainult Eestis loodud klientide ehk ettevõtete kohta, seega pole mõtet teisi andmeid andmebaasist pärida. Kliendi staatuse ning oleku kriteeriumid on vaikimisi määratud filtreerimistingimused, mis igas päringus peavad eksisteerima, et asjakohased andmed saada.

Määratud ülesande kriteeriumid

Määratud ülesande kriteeriumiks on seos kasutaja ehk ülesande täitja ja ülesande vahel ehk filtreeritakse välja ainult sellised ülesanded, mis on vähemalt ühe kasutaja nimel. Tabeli määratud ülesande kohta saab täpsemalt lugeda osas 3.2.

5.5.1.2 Andmestiku filtreerimine kooditasandil

Selleks et hoida andmete filtreerimisel ning andmestiku moodustamisel efektiivsust ning kiirust, ei ole mõistlik kogu filtreerimise osa teha andmebaasi tasandil. Seetõttu on keerukamad filtreerimised teostatud kooditasandil.

Kooditasandil filtreerimise kriteeriumid on:

1. Kliendi kohta leidub andmebaasiväliseid lisaandmeid
2. Ülesande kestused jäävad vahemikku 2 minutit kuni 16 tundi
3. Iga kliendi kohta lubatud maksimaalsete ülesannete arv on 50

Selleks, et oleks võimalik mudelit treenida, on tarvis kliendi kohta täpsemaid andmeid nagu ettevõtte tegevusala ning töötajate arv. Kui kliendil lisaandmeid ei leidu, tuleb selline ülesande objekt andmestikust eemaldada, et vältida puudulike andmetega objektide kasutamist mudelite treenimisel. Selline kriteerium vähendab tublisti andmemahtu, kuid see-eest annab indikatsiooni, kas tulevikus oleks tarvis kõigi klientide kohta käivat lisainfot koguda või mitte.

Ülesande kestuse ülemise ja alumise piiri määramise põhjuseks on ekstreemsuste eemaldamine andmetest. Kui ülesanne kestab alla alumise lubatud piiri ehk alla 2 minuti, siis on suure tõenäosusega tegemist müraga ehk aega ei ole kas korralikult mõõdetud või on tegemist nii väikese ülesandega, mida ei ole mõistlik mudeli treenimisel kasutada. Ülemise piiri puhul on eesmärk filtreerida ekstreemselt pikad ülesanded ehk ülesanded, mille pikkus on 16 tundi või teisisõnu 2 tööpäeva. Võrdluseks võib tuua, et keskmine juhuslikülesanne võtab aega 50 minutit.

Kliendipõhine ülesannete piiramine aitab paremini ühtlustada andmeid erinevate klientide vahel ning aitab kaasa üldistatud mudeli loomisele. Näiteks kui ühele kliendil tehakse ettemääratud perioodi jooksul 30 ülesannet ning teisel 500, siis ülesande lubatud ülempiir aitab kahte klienti ühtlustada, et ei tekiks üksikute klientide domineerimist treenitavas mudelis. Ülemiseks piiriks on valitud 50 ülesannet kliendi kohta ning kui ülesandeid on kliendil rohkem, siis valitakse ülesandeid juhuslikkuse alusel.

5.5.1.3 Treening- ja testandmeteks jagamine

Peale filtreerimist moodustab andmestiku 67 000 juhuslikku ülesannet, mis hõlmas endas 87 erinevat finantsmeeskonda, 4698 klienti ning 344 raamatupidajat. Seejärel toimub andmestiku jagamine treening- ning testandmeteks. Treeningandmed on mõeldud mudeli treenimiseks ning nende osakaal moodustab 75% filtreeritud andmestikust ehk umbes 50 000 ülesannet. Testandmed on mõeldud mudeli valideerimiseks ja täpsuse ennustamiseks

ning nende andmete osakaal on 25% ehk umbes 17 000 ülesannet. Oluline on, et treeningandmed ning testandmed oleksid rangelt lahus, vastasel korral on mudeli täpsus petlik.

5.5.2 Andmete viimine vektorkujule

Selles alamosas tuleb juttu vektori moodustamisest, mida masinõppemudel treenimisel ja valideerimisel kasutada saab. Andmete viimine vektorkujule tähendab andmete teisendamist masinõppemudeli keelde. Täpsemalt teisenduste põhimõtetest saab lugeda osas 5.2.

Mudeli atribuudid

Mudeli treenimiseks sobivate atribuutide valimisel lähtuti andmestiku analüüsist, mille tulemusel leiti kõik atribuudid, mis aitavad tuvastada sarnaseid ülesandeid. Mudeli treenimisel kasutati neid atribuute, millel oli suurim kaal ehk mõju mudeli ennustusvõimele.

Valitud atribuutideks on:

1. Ülesande pealkiri (tekstiväärtus)
2. Ettevõtte identifikaator (täisarvuline väärtus)
3. Ülesande tegemise kuu (täisarvuline väärtus)
4. Kliendi tegevusala (tekstiväärtus)
5. Kliendi identifikaator (täisarvuline väärtus)
6. Kliendi (ettevõtte) töötajate arv (täisarvuline väärtus)
7. Ülesande kestus (täisarvuline väärtus, mille ühikuks on sekund)

Ülesande pealkiri on tekstiväärtus, mis kirjeldab kõige paremini olemasolevatest andmetest ülesande sisu ning aitab leida seoseid ülesannetega, millel on sarnane pealkiri. Ülesande tegemise kuu annab võimaluse leida mustreid ülesannete vahel, mis on teostatud samas kuus ning ettevõtte identifikaator üldistab andmeid üle raamatupidamisbüroo.

Kliendi tegevusala on samuti tekstiväärtus nagu ülesande pealkiri ning see kirjeldab kõige paremini, mis ettevõttega on tegemist. Nii tegevusala kui kliendi ettevõttes olevate töötajate arv aitavad leida seoseid sarnase suure ja tegevusalaga klientide vahel. Kliendi identifikaator kirjeldab, millisele kliendile ülesanne tehti ning aitab leida mustreid samale kliendile tehtavate ülesannete vahel.

Vektorkujule viimine

Juhuslike ülesannete ajahinnangu mudeli andmestik jaguneb nominaal- ning ordinaalandmeteks.

Nominaalandmeteks on: ettevõtte identifikaator ning kliendi identifikaator. Mõlema puhul luuakse n kahendatribuudiga vektor ehk vektor, kus igale identifikaatorile vastab üks kindel element vektoris. Selleks et vektorit optimeerida, on lõputöös loodud eraldi andmestruktuur, kus hoitakse teadmist, mis indeksile vektoris, mis identifikaator vastab. Selline teadmine salvestatakse peale mudeli treenimist andmebaasi Redis, et hiljem treenitud mudeli kasutamisel oleks võimalik sisendandmeid samadel alustel vektorkujule viia.

Ülesande tegemise kuu kuulub samuti nominaalandmete hulka, kuid võrreldes ettevõtte ning kliendi identifikaatoritega, on tegemist lihtsama olukorraga, kus tuleb luua 12-elemendiline vektor, kus iga element on üks kuu aastas. Kuna kuid saab loogiliselt järjestada, siis ei ole tarvis eraldi teadmist Redisesse salvestada, vaid alati on teada, mis kuu, mis indeksile vektoris vastab.

Nii kliendi töötajate arv kui ka ülesande kestus kuuluvad ordinaalandmete hulka. See tähendab, et igale väärtusele luuakse üheelemendiline standardiseeritud vektor. Täpsemalt saab lugeda andmete standardiseerimisest alamosas 5.3.2. Selle eesmärgiks on vähendada mudeli treenimisel tekkivat müra, kuna masinõppel on lihtsam töötada andmetega, mis on standardiseeritud. Standardiseerimisprotsessis leitud atribuudi minimaalne ning maksimaalne väärtus salvestatakse peale mudeli treenimist andmebaasi Redis, et hiljem treenitud mudeli kasutamisel, saaks vektori samadel tingimustel kokku panna.

Tekstiväärtuste vektorkujule viimiseks treenitakse esmalt kaks Gensim-i tekstimudel. Esimene mudel on ülesande pealkirja ning teine kliendi põhitegevusala jaoks.

Treeningprotsessi tulemusel moodustub tekstiväärtusest vektor, mida saab mudeli treenimisel kasutada. Vektori pikkuseks ülesande pealkirjale on määratud 500 elementi ning kliendi põhitegevusalale 200 elementi. Vastavad elementide arvud on leitud katseksitus meetodil. Täpsemalt saab Gensim-i tööpõhimõtetest lugeda osas 5.5.

Tervikvektori moodustamine

Tervikvektori moodustamiseks liidetakse kõik atribuutidest moodustatud vektorid üksteisele otsa ning moodustatakse nii üks suur vektor, mis kirjeldabki konkreetset juhuslikku ülesannet.

Juhuslik ülesanne koosneb järgmiste pikkustega alamvektoritest:

1. Ülesande pealkiri – 500 elementi
2. Ettevõtte identifikaator – 87 elementi
3. Ülesande tegemise kuu – 12 elementi
4. Kliendi tegevusala – 200 elementi
5. Kliendi identifikaator – 4598 elementi
6. Kliendi (ettevõtte) töötajate arv – 1 element

Koguvektori pikkuseks kujunes 5698 elementi. Oluline on jälgida vektori kogupikkust, kuna sellest sõltub muutmälu kasutamise vajadus treeningprotsessis ning treenimise kiirus.

5.5.3 Modelleerimine

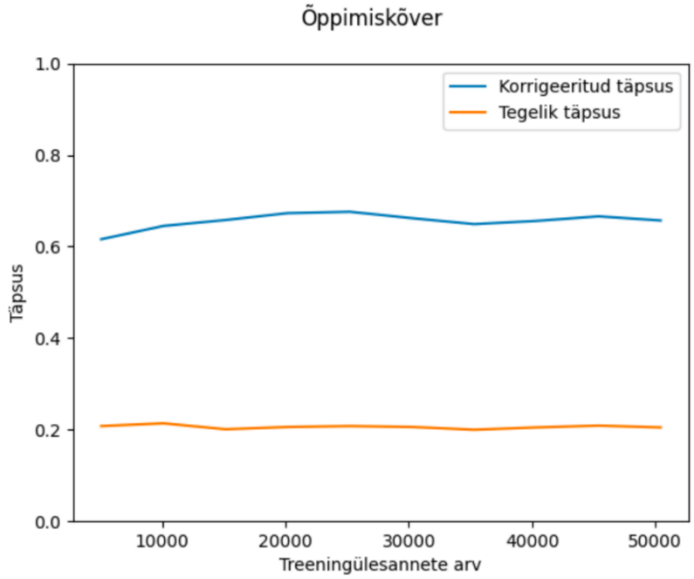
Modelleerimise etapis loodi 5 regressiooni mudelit, mida treenimiseks kasutati. Nendeks mudeliteks olid: LassoLars [18], Lasso [19], Ridge [20], KNeighborsRegressor [21] ning Lineaarne regressioon [22]. Täpsemalt saab nendest lugeda osas 4.4 Modelleerimine.

5.5.4 Hindamine

Selles alamosas tuleb juttu treenitud mudelite täpsusest ning õppimiskõverast.

Õppimiskõver

Õppimiskõver näitab treening- ning testandmete sõltuvust andmestiku suurusest. Teisisõnu, kui fikseerida testandmete suurus ning muuta treeningandmete suurust, siis õppimiskõver näitab, kuidas treeningandmestiku suurus mõjutab mudeli täpsust.



Joonis 15. Juhuslike ülesannete ajahinnangu ennustamise mudeli õppimiskõver

Joonisel 15 tähistab x-telge treeningandmete suurus ning y-telge mudeli täpsus. Joonisel on näha kahte funktsiooni, millest sinine näitab korrigeeritud vahemiku täpsust ning oranž tegelikku täpsus. Mõlema täpsuse leidmiseks esmalt konverteeritakse ülesande kestus eeldefineeritud minutitesse vastavalt sellele, mis väärtus on antud ülesande puhul lähim (vt joonis 19). Korrigeeritud vahemiku täpsuse puhul leitakse tõenäosus, et konverteerid ülesande kestus langeb ettemääratud vahemikku ning tegelik täpsuse puhul leitakse tõenäosus, et konverteeritud ülesande kestus langeb kokku konverteeritud tegeliku kestusega. Täpsemalt on kogu konverteerimise ja hindamise protsessist juttu osas 5.7.

Joonise 15 põhjal saab järeldada, et treeningandmete suurenemisel täpsus ei parane ning seega on mõistlik hoopis kasutada hinnanguliselt 50% kõigist treeningandmetest, et muuta mudeli treenimist kiiremaks ning optimeerida seeläbi ka muutmälu kasutust.

Ennustustulemused

Treenitud mudelite ennustustäpsused on näha tabelis 2, kus on eraldi leitud korrigeeritud vahemiku täpsus ning tegelik täpsus ja lisatud juurde standardhälve, mis näitab täpsuse

varieerumist. Standardhälve on leitud mitmekordsel mudeli treenimisel, kus treening- ning testandmed on igal iteratsiooni korral erinevad.

Tabel 2. Juhuslike ülesannete ajahinnangu mudelite täpsused

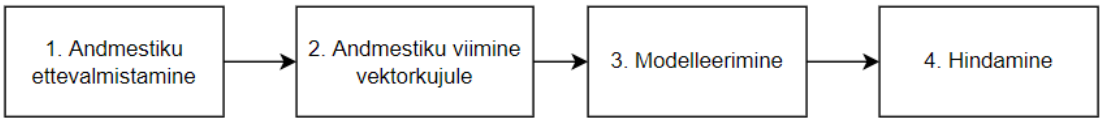
Mudel	**Korrigeeritud täpsus**	**Tegelik täpsus**
LassoLars	0.68 +- 0.01	0.17 +- 0.00
Lasso	0.67 +- 0.00	0.21 +- 0.01
KNeighborsRegressor	0.66 +- 0.01	0.20 +- 0.01
Ridge	0.63 +- 0.01	0.21 +- 0.00
LinearRegression	0.62 +- 0.01	0.21 +- 0.01

Nagu tabelist 2 näha, siis üldiselt on mudelitel varieerumine väike ning parimateks mudeliteks osutuvad korrigeeritud vahemiku täpsuse põhjal nii Lasso, LassoLars kui ka KNeighborsRegressor, kuid võrreldes tegelikku täpsust, siis parim mudel, mida kasutada on Lasso.

Lisaks sai mudelite treenimise käigus hinnatud, kas lisaandmed, mis kliendi kohta osas 3.3 leiti, aitavad leida seoseid sarnaste klientide vahel ning paranda mudeli täpsust. Selgus, et lisaandmed lisasid keskmiselt 2 protsendipunkti täpsust mudeli täpsusele juurde ning seega aitasid vähesel määral mudeli täpsust paremaks muuta.

5.6 Korduvate ülesannete ajahinnangu mudel

Korduvate ülesannete ajahinnangu mudeli loomine toimub neljas etapis.



Joonis 16. Korduvate ülesannete ajahinnangu mudeli loomise etapid

Kõigepealt valmistatakse ette andmestik, seejärel viiakse andmed vektorkujule. Sellele järgneb andmete modelleerimine ehk mudelite loomine ning siis hinnatakse mudelite headust ning valitakse välja parim.

5.6.1 Andmestiku ettevalmistamine

Korduvate ülesannete ajahinnangu mudeli treenimiseks kasutatav andmestik moodustatakse kahes etapis. Esmalt filtreeritakse andmed põhirakenduse andmebaasist ning seejärel kooditasandil. Selle eesmärgiks on optimaalselt müra ning ekstreemsete andmete välja filtreerimine.

5.6.1.1 Andmestiku moodustamine andmebaasi päringuga

Andmestiku moodustamiseks kasutatakse kolme tabelit: ülesanne, klient ning määratud ülesanne. Täpsemalt on kirjeldatud nende tabelite sisusid osas 3.2.

Ülesande kriteeriumid

Ülesanne peab vastama järgmistele kriteeriumitele:

1. Peab omama viidet baasülesandele
2. Peab olema ajavahemikus *2020-06-01* kuni *2022-03-01*
3. Peab olema staatuses „lõpetatud“ ning ei tohi olla olekus „kustutatud“
4. Peab omama ajamõõdet/ajamõõtmeid (ülesande kestus)

Selleks, et tagada ainult korduvate ülesannete olemasolu andmestikus, on andmestikku filtreeritud baasülesande viite järgi. Andmestik on piiratud ajaliselt, mis tuleneb Uku suurkliendi järgi, kes sel perioodil platvormi kasutamise tõsisemalt ette võttis. Staatuse, oleku ning ülesande kestuse kriteeriumid on vaikimisi määratud filtreerimistingimused, mis igas päringus peavad eksisteerima, et asjakohaseid andmeid saada.

Kliendi kriteeriumid

Klient peab vastama järgmistele kriteeriumitele: olema staatuses „aktiivne“ ning olekus „mitte kustutatud“. Eespool kirjeldatud kliendi kriteeriumid, on vaikimisi määratud filtreerimise tingimused, asjakohaste andmete saamiseks.

Määratud ülesande kriteeriumid

Määratud ülesande kriteeriumiks on seos kasutaja ehk ülesande täitja ja ülesande vahel ehk filtreeritakse välja ainult sellised ülesanded, mis on vähemalt ühe kasutaja nimel.

5.6.1.2 Andmestiku filtreerimine kooditasandil

Selleks et hoida andmete filtreerimisel ning andmestiku moodustamisel efektiivsust ning kiirust, ei ole mõistlik kogu filtreerimise osa teha andmebaasi tasandil. Seetõttu on keerukamad filtreerimised teostatud kooditasandil.

Filtreerimise kriteeriumid:

1. Ülesande kestus jäävad vahemikku 2 minuti kuni 16 tundi
2. Iga kliendi kohta lubatud maksimaalsete ülesannete arv on 20

Ülesande kestuse ülemise ja alumise piiri määramise põhjuseks on ekstreemsuste eemaldamine andmetest. Kui ülesanne kestab alla alumise lubatud piiri ehk alla 2 minuti, siis on suure tõenäosus, et tegemist müraga ehk aega ei ole kas korralikult mõõdetud või on tegemist nii väikese ülesandega, mida ei ole mõistlik mudeli treenimisel kasutada. Ülemise piiri puhul on eesmärk filtreerida ekstreemselt pikad ülesanded ehk ülesanded, mille pikkus on 16 tundi või teisisõnu 2 tööpäeva. Võrdluseks võib tuua, et keskmine korduvülesanne võtab aega 48 minutit.

Kuna baasülesandeid on väga palju, siis arvutiressursi nappuse tõttu ei ole võimalik kliendipõhiselt 50 ülesannet kasutada nagu juhuslike ajahinnangute mudelis, vaid tuleb piirduda 20-nega. Kliendi üleselt ülesannete piiramine aitab paremini ühtlustada andmeid erinevate klientide vahel ning optimeerida muutmälukasutust. Kui ülesandeid on kliendil rohkem kui 20, siis tehakse valik juhuslikkuse alusel.

5.6.1.3 Treening- ja testandmeteks jagamine

Peale filtreerimist moodustab andmestiku 71 000 korduvat ülesannet, mis hõlmas endas 210 erinevat finantsmeeskonda, 5323 klienti ning 664 raamatupidajat. Ülesande korduste arv andmestikus kujuneb juhuslikkuse alusel, kuna treenimiseks on piiratud ressurss. Seejärel toimub andmestiku jagamine treening- ning testandmeteks. Treeningandmed on mõeldud mudeli treenimiseks ning nende osakaal moodustab 75% filtreeritud andmestikust ehk umbes 53 000 ülesannet. Testandmed on mõeldud mudeli valideerimiseks ja täpsuse ennustamiseks ning nende andmete osakaal on 25% ehk 18 000 ülesannet. Oluline on, et treeningandmed ning testandmed peavad olema rangelt lahus, vastasel korral on mudeli täpsus petlik.

5.6.2 Andmestiku viimine vektorkujule

Selles alamosas tuleb juttu vektori moodustamisest, mida masinõppemudeli treenimisel ja valideerimisel kasutada saab. Andmete viimine vektorkujule tähendab andmete teisendamist masinõppemudeli keelde. Täpsemalt teisenduste põhimõtetest saab lugeda osas 5.2.

Mudeli atribuudid

Mudeli treenimiseks sobivate atribuutide valimisel lähtuti andmestiku analüüsist, mille tulemusel leiti kõik atribuudid, mis aitavad tuvastada sarnaseid ülesandeid. Mudeli treenimisel kasutati neid atribuute, millel oli suurim kaal ehk mõju mudeli ennustusvõimele.

Valitud atribuutideks on:

1. Ülesande pealkiri (tekstiväärtus)
2. Kordumise identifikaator (täisarvuline väärtus)
3. Ettevõtte identifikaator (täisarvuline väärtus)
4. Kliendi identifikaator (täisarvuline väärtus)
5. Ülesande täideviija identifikaator (täisarvuline väärtus)
6. Ülesande kestus (täisarvuline väärtus)

Ülesande pealkiri on tekstiväärtus, mis kirjeldab kõige paremini olemasolevatest andmetest ülesande sisu ning aitab leida seoseid ülesannetega, millel on sarnane pealkiri. Ettevõtte identifikaator aitab üldistada andmeid üle raamatupidamisbüroo, kliendi identifikaator aitab üldistada andmeid kliendipõhiselt ning kasutaja identifikaator kasutajapõhiselt. Kordumise identifikaatori abil on mudelil võimalik leida seoseid ühisest baasülesandest genereeritud ülesannete vahel.

Mudeli atribuutidest puudub registrikood, mille abil juhuslike ülesannete ajahinnangu ennustamise mudeli jaoks kliendi lisaandmeid leiti. Põhjuseks on vektori pikkuse optimeerimine lisaandmete arvelt, kuna nende mõju täpsusele on üpris madal.

Vektorkujule viimine

Korduvate ülesannete ajahinnangu mudeli treenimiseks, tuleb atribuudid esmalt teisendada vektorkujule. Põhimõtted, mida selleks järgida tuleb on kirjutatud osas 5.2.

Korduvate ülesannete ajahinnangu mudeli treenimisel on nominaalandmeteks: ettevõtte identifikaator, kliendi identifikaator, kasutaja identifikaator ning kordumise identifikaator. Kõigi nende puhul loodi n kahendatribuudiga vektor ehk vektor, kus igale identifikaatorile vastab kindel element. Selleks et vektorit optimeerida on lõputöös loodud eraldi andmestruktuur, kus hoitakse teadmist, mis indeksile vektoris, mis identifikaator vastab. Selline teadmine salvestatakse peale mudeli treenimist andmebaasi Redis, et hiljem treenitud mudelit kasutusele võttes oleks võimalik andmeid samadel alustel vektorkujule viia.

Ülesande kestus kuuluvad ordinaalandmete hulka. See tähendab, et igale väärtusele luuakse üheelemendiline standardiseeritud vektor. Täpsemalt saab lugeda andmete standardiseerimisest alamosas 5.3.2. Selle eesmärgiks on vähendada mudeli treenimisel tekkivat müra, kuna masinõppemudelil on lihtsam töötada andmetega, mis on standardiseeritud. Standardiseerimis protsessis leitud atribuudi minimaalne ning maksimaalne väärtus salvestatakse peale mudeli treenimist andmebaasi Redis, et hiljem treenitud mudeli kasutamisel, saaks vektori samadel tingimustel luua.

Esimene probleem, mis üles kerkis oli seotud kordumise identifikaatoriga. Nimelt leidus andmestikus ligi 15 000 erinevat kordumise identifikaatorit, mis tähendas, et üksnes ühe atribuudi vektori pikkus oli 15 000 elementi. Sellest tulenevalt oli koguvektori pikkus liiga suur, et treeningprotsessi läbi viia, kuna arvutiressurss sai treeningprotsessil otsa. Probleemi prooviti lahendada dimensionaalsuse vähendamise tehnikat kasutades [24]. Selleks lisati kordumise identifikaator ülesande pealkirjale sõnana juurde ning prooviti, kas Gensim suudab seda kasutada oma mudelis, et ühe baasülesande põhjal loodud ülesannete vahel seos luua.

Tekstiväärtuste vektorkujule viimiseks treenitakse ülesande pealkirjade jaoks Gensim-i tekstimudel. Ülesande pealkirjale on sõnana lisatud ülesande korduse identifikaator. Treeningprotsessi tulemusel moodustub tekstiväärtusest vektor, mida saab mudeli treenimisel kasutada. Vektori pikkuseks on määratud 1000 elementi, mis osutus katsetades parimaks. Täpsemalt saab Gensim-i tööpõhimõtetest lugeda osas 5.5.

Tervikvektori moodustamine

Tervikvektori moodustamiseks liidetakse kõik atribuutidest moodustatud vektorid üksteisele otsa ning moodustatakse nii üks suur vektor, mis kirjeldabki konkreetset korduvat ülesannet.

Korduv ülesanne koosneb järgmiste pikkustega vektorosadest:

1. Ülesande pealkiri – 1000 elementi
2. Ettevõtte identifikaator – 210 elementi
3. Kliendi identifikaator – 5323 elementi
4. Kasutaja identifikaator – 664 elementi
5. Ülesande tegemise kuu – 12 elementi

Koguvektori pikkuseks kujunes 7209 elementi. Oluline on jälgida vektori kogupikkust, kuna sellest sõltub muutmälu kasutamise vajadus treeningprotsessis ning treenimise kiirus.

5.6.3 Modelleerimine

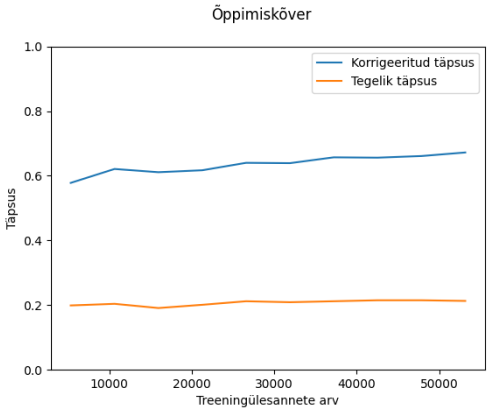
Modelleerimise etapis loodi viis regressiooni mudelit nagu ka juhuslike ülesannete puhul. Nendeks mudeliteks olid: LassoLars [18], Lasso [19], Ridge [20], KNeighborsRegressor [21] ning Lineaarne regressioon [22]. Täpsemalt saab modelleerimisest ja erinevatest mudelitest lugeda osas 4.4 Modelleerimine.

5.6.4 Hindamine

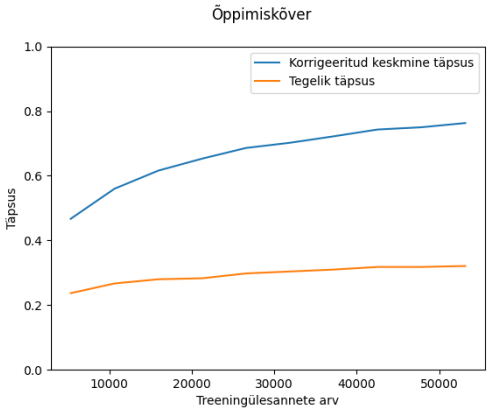
Selles alamosas tuleb juttu treenitud mudelite täpsusest ning õppimiskõverast.

Õppimiskõver

Õppimiskõver näitab treening- ning testandmete sõltuvust andmestiku suurusest. Teisisõnu, kui fikseerida testandmete suurus ning muuta treeningandmete suurust, siis õppimiskõver näitab, kuidas treeningandmestiku suurus mõjutab mudeli täpsust.



Joonis 17. Korrigeeritud täpsuse õppimiskõver



Joonis 18. Korrigeeritud keskmise täpsuse õppimiskõver

Joonisel 17 ja 18 tähistab x-telge treeningandmete suurus ning y-telge mudeli täpsus. Mõlemal joonisel on nähe kahte funktsiooni, mis kirjeldavad joonisel nimetatud täpsuseid. Kõigi täpsuste leidmiseks esmalt konverteeritakse ülesande tegelik kestus ning ennustatav kestus eeldefineeritud minutitesse, vastavalt sellele, mis väärtus on antud kestuse puhul lähim. Täpsemalt saab konverteerimise protsessist lugeda osas 5.7.

Joonisel 17 kirjeldatakse masinõppe mudeli poolt ennustatavaid ajahinnangu täpsuseid. Korrigeeritud vahemiku täpsuse puhul leitakse tõenäosus, et konverteeritud ülesande kestus langeb ettemääratud vahemikku ning tegelik täpsuse puhul leitakse tõenäosus, et konverteeritud ülesande kestus langeb kokku konverteeritud tegeliku kestusega.

Joonisel 18 kirjeldatakse masinõppe mudeli treeningandmete pealt baasülesande lõikes arvutatud keskmist korduva ülesande kestust. See tähendab, et leitakse keskmine kestus kõigi nende ülesannete vahel, millel on sama kordumise identifikaator ehk viide ühisele baasülesandele. Korrigeeritud keskmise vahemiku täpsuse puhul leitakse tõenäosus, et konverteeritud keskmise ülesande kestus langeb ettemääratud vahemikku sõltuvalt tegelikust kestusest ning tegelik täpsuse puhul leitakse tõenäosus, et konverteeritud keskmise ülesande kestus langeb kokku konverteeritud tegeliku kestusega.

Jooniste 17 ja 18 põhjal võib välja lugeda, et vastavalt kordumise identifikaatorile leitud keskmine ülesande kestus on tunduvalt täpsem kui masinõppe mudeli pool ennustatud kestus. Põhjuseks võib välja tuua, et mudeli treenimisel tuli teha mitmeid kitsendusi, mis olid tingitud arvutivõimsusest. Selle tulemusel ei olnud võimalik ühe baasülesande poolt

genereeritud korduvate ülesannete vahel tugevaid seoseid leida. See on ka üheks põhjuseks, miks korduvate ülesannete mudel ei tulnud parem kui juhuslike ülesannete mudel. Vaadates joonist 17 ja 18, siis mõlema puhul on näha, et sinine funktsioon on kasvavas trendis. See tähendab, et treeningandmete lisamisel on võimalik saavutada parem täpsus, kuid ilmselt tuleb parema täpsuse nimel uurida masinõppe teeke ja mudeleid, mida on võimalik treenida osade kaupa ning seeläbi hoida muutmälukasutust treeningprotsessis madalamana.

Ennustustulemused

Treenitud mudelite ennustuse täpsused on näha allolevas tabelis, kus on eraldi leitud korrigeeritud täpsus ning tegelik täpsus ja lisatud juurde standardhälve, mis näitab täpsuse varieerumist. Standardhälve on leitud mitmekordsel mudeli treenimisel, kus treening- ning testandmed on igal iteratsiooni korral erinevad.

Tabel 3. Korduvate ülesannete ajahinnangu mudelite täpsused

Mudel	**Korrigeeritud täpsus**	**Tegelik täpsus**
Keskmine ülesande kestus	0.76 +- 0.01	0.32 +- 0.01
KNeighborsRegressor	0.67 +- 0.01	0.23 +- 0.01
Lasso	0.66 +- 0.00	0.21 +- 0.01
Ridge	0.65 +- 0.00	0.22 +- 0.00
LinearRegression	0.64 +- 0.01	0.22 +- 0.01
LassoLars	0.57 +- 0.01	0.15 +- 0.00

Nagu tabelis 3 näha, siis üldiselt on mudelitel varieerumine väike ning parima tulemuse saab kui iga baasülesande poolt genereeritud korduvate ülesannete kestuste keskmine leida. Keskmine kestus edestab parimat treenitud mudelit KNeighborsRegressor-i täpsust ligi 10 protsendipunktiga. Masinõppe mudelite poolt välja pakutud ülesande ajahinnangud on ebatäpsemad kui keskmine ülesande kestus, kuna ülesande malle ehk baasülesandeid on palju ning ei olnud võimalik mudeli treenimisel kasutada vastavat identifikaatorit, mis omavahel samast baasülesandest loodud ülesandeid seostaks.

5.7 Gensim tekstimudel

Tekstimudeli loomisel on kasutatud tekstitöötlus teeki Gensim, mis võimaldab esitada dokumente semantiliste vektoritena ning tema algoritmid avastavad automaatselt dokumentide semantilise struktuuri uurides statistilisi koosesinemise mustreid. Lõputöös on kasutatud Gensim *Doc2Vec* mudelit, mis esitab igat dokumenti vektorina ja on meetodi *Word2Vec* üldistus [8].

Esimese sammuna viiakse treeningandmetes olev tekstiparameeter dokumendi kujule. Selleks tuleb tekstiväärtus lammutada laiali sõnadeks ning sõnade kogumit kasutada koos dokumendi järjekorra indeksiga *Doc2Vec* meetodi *TaggedDocument()* sisendina, mis tagastab indekseeritud dokumendi [8].

Pärast tekstide dokumentide kujule viimist, on võimalik luua mudel, mille abil on hiljem võimalik tekstile vastav vektor leida. Mudeli loomiseks tuleb defineerida vektori suurus ning minimaalse sõna pikkus, milleks on projekti üleselt määratud 3 tähemärki. Vektori suurus varieerub vastavalt lõputöös loodud mudelitele, eesmärgiga hoida koguvektori pikkust optimaalsena, kuna see avaldab mõju arvutiressursi kasutusele mudelite treeningprotsessides.

Mudeli loomise järel toimub sõnavara ehitamine, mille käigus koostatakse vastavalt eelpool loodud dokumentide põhjal statistika ning seejärel treenitakse mudel. Pärast seda salvestatakse mudel binaarsel kujul faili ning selle kasutamiseks loetakse mudel failist mällu. Lõpptulemuseks on treenitud mudel, mis suudab sisendiks antud teksti viia vektorkujule, mille väärtused teksti kõige paremini kirjeldavad.

5.8 Mudelite täpsuse hindamine

Mudeli ennustustäpsuse hindamiseks ei ole mõistlik kasutada üksühele võrdlust ennustuse ja tegelikkuse vahel, kuna ülesande kestuse ühikuks on sekund, siis ärivajadusest tuleneva funktsionaalsuse realiseerimiseks ei ole sekundite võrdlemisel saadav täpsus vajalik.

Selleks, et mudelitäpsust paremini ja mõistlikumalt hinnata, on loonud eraldi struktuur arvudest, mille iga element tähistab ühte kindlat ajalist väärtus minutites. Sellise struktuuri eesmärk on defineerida kindlad väärtused, milleks kõik ülesannete

ajahinnangud hiljem konverteeritakse ehk teisisõnu pidevad väärtused (ülesande kestused) teisendatakse diskreetseteks väärtusteks, mis on kirjeldatud joonisel 19. Konverteerimisel lähtutakse põhimõttest, et iga ajahinnang teisendatakse lähima väärtuseni, mis paikneb defineeritud struktuuris.

$$\begin{bmatrix} 5, 10, 15, 30, 45, 60, 90, 120, 180, 240, 480, 720, \\ 960, 1200, 1440, 1680, 1920, 4000, 8000 \end{bmatrix}$$

Joonis 19. Eeldefineeritud ülesande kestused minutites

Oletame, et ülesande ajahinnang on 700 sekundit, ümardatult on see 12 minutit ja kuna struktuuris on lähim väärtus sellele 10 ehk 10 minutit, siis ülesande uueks ajahinnanguks saab 10 minutit.

Struktuur on välja töötatud meetodil, kus iga järgmine element struktuuris kasvab proportsionaalselt oma väärtusega ehk mida väiksem väärtus, seda väiksem on erinevus tema ning järgmise elemendi vahel. Samuti on arvestatud, kui suurt ajahinnangu täpsust on tarvis saavutada. Suures pildis pole vahet kas ülesanne võtab aega 3.5 tundi või 4 tundi, kuna selles kontekstis võib arvestada, et ülesanne võtab aega umbes pool päeva.

Ülesannete ja ajahinnangute juures mängib suurt rolli ka korrapäratus. Näiteks leidub palju ülesandeid, mis võivad ajaliselt varieeruda 10 minuti ja 4 tunni vahel. Seega ei ole mõistlik piirduda ainult struktuuriväärtusteks konverteeritud väärtuste üksühele võrdlusega, vaid kasutada ka vahemikke, millesse jäädes loetakse ennustust täpseks.

Lõputöös on kasutatud vahemikke, kus ennustust loetakse täpseks kui tegeliku ülesande kestus on kas 1 või 2 elementi defineeritud struktuuris eespool või 1 element taga pool. See tähendab, et kui ennustus on 30 minutit, siis loetakse see täpseks kui tegeliku ülesande kestus on kas 10, 15, 30 või 45 minutit. Põhjus, miks on eelistatud pigem suuremat ennustust kui tegelikus, tuleneb asjaolust, et tulevikus on plaan kasutada ajahinnanguid töökoormuste hindamiseks, mis tähendab seda, et liiga optimistlikud ajahinnangud annavad valesid signaale ettevõttele, kes oma töötajate töökoormusi jälgivad. Ülesande kestuse suuremaks ennustamine annab aga varakult märku potentsiaalsest ülekoormuse tekkimisest töötajal.

Selline julge ümardamine ning üpris suure vahemiku järgi ennustustäpsuse hindamine tuleneb asjaolust, et ülesannete ajaline kestus on samade ülesande lõikes suuresti

varieeruv ning täpseid tulemusi on seega keeruline saavutada. Seda väidet kinnitab andmestikust pärinev näide, kus samast baasülesandest genereeritud korduvate ülesannete lõikes keskmise ülesande kestus minutites ning standardhälve tulid järgnevad: 19.67 ± 24.47. Mida suurem on standardhälve, seda varieeruvam on ülesande kestus ehk antud näites on tegemist väga suure standardhälbega. Näite täpsustusena võib lisada, et tegemist ei ole kõige äärmuslikuma olukorraga, vaid pigem keskmise varieeruvusega korduvate ülesannete puhul.

6 Tulemuste valideerimine

Tööplaani tulemuste valideerimine viidi läbi CH Konsultatsioonid OÜ[1] raamatupidajaga, kes andis tagasisidet lõputöös loodud funktsionaalsusele läbi kasutaja pilgu. Lisaks testiti algoritmi numbriliste mõõdikute järgi osas 4.5.

Kuna käesolev töö ei jõudnud põhirakendusega integreerimise faasi, siis raamatupidajaga valideerimiseks loodi talle eraldi raport tema klientidest ning nendele tehtavatest korduvatest ülesannetest. Joonisel 20 on näidis raportist ühe kliendi põhjal.

	Kliendi nimi	Ülesande pealkiri	Vana / uus kordumisreegel	Alustusaeg	Tähtaeg	Ajahinnangu ennustus / tegelik
0	TEST KLIENT	Aruanne 1	1 -> 7 (+6)	9 | 14:14:30 (1)	14 | 23:59:59	0 | 0
1	TEST KLIENT	Aruanne 2	1 -> 4 (+3)	7 | 11:31:34 (1)	14 | 23:59:59	0 | 0
2	TEST KLIENT	Aruanne 3	1 -> 18 (+17)	18 | 11:17:38 (1)	14 | 23:59:59	0 | 0
3	TEST KLIENT	Raamatupidamine	1 -> 3 (+2)	3 | 10:38:30 (6)	28 | 23:59:59	65 | 49
4	TEST KLIENT	Dokumentide sisestamine	1 -> 13 (+12)	15 | 13:43:40 (2)	28 | 23:59:59	32 | 23
5	TEST KLIENT	Pankade sisestamine	1 -> 10 (+9)	22 | 14:31:13 (1)	28 | 23:59:59	27 | 47

Joonis 20. Raamatupidajaga valideerimiseks loodud raporti näidis

Andmed on ühe kuu põhised ning esimene veerg „Kliendi nimi“ näitab, millisele kliendile kasutaja ülesannet tegi ning veerg „Ülesande pealkiri“ kirjeldab kliendile tehtava ülesande sisu. Kolmas veerg „Vana/uus kordumisreegel“ näitab, kuidas muutub ülesande kordumisreegel ehk kuupäev, kuhu ülesanne paigutatakse algoritmi rakendamise tulemusel ning sulgudes on välja toodud päevade erinevus. Veeru „Alustusaeg“ esimene element on päev, teine kellaaeg ning sulgudes mitmel erineval päeval keskmiselt kasutaja selle ülesandega tegeleb. Veerg „Tähtaeg“ on analoogne veerule „Algusaeg“, kus esimene element on päev ja teine kellaaeg. Viimane veerg „Ajahinnangu ennustus / tegelik“ võrdleb ajahinnangu ennustust (antud juhul ülesande keskmine kestus) tegeliku ülesandele kulunud ajaga, mille ühikuks on minut.

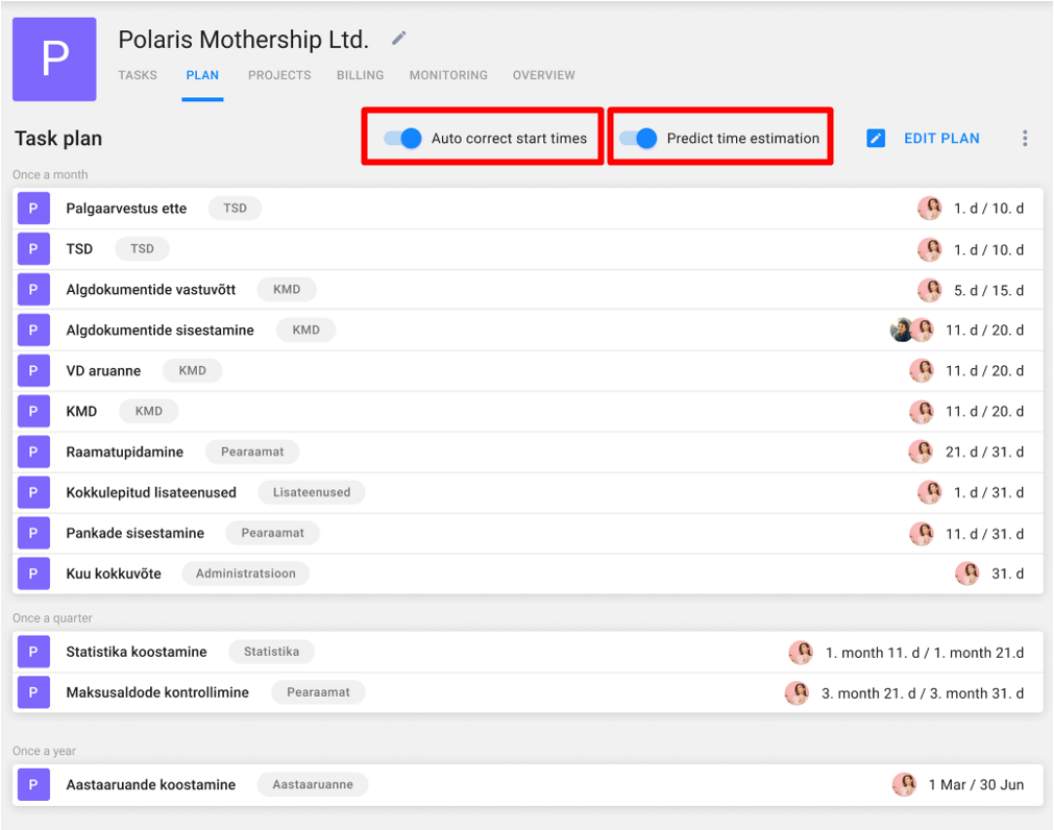
[1] CH Konsultatsioonid OÜ puhul on tegemist ühe suurima raamatupidamisbürooga Eestis (üle 50-ne raamatupidaja) ning suurima kliendiga, kes Uku platvormi kasutab.

Valideerimise käigus selgus, et need ülesanded, mille kestus on 0 minutit, kasutatakse tegelikult loendamise eesmärgil ning neid kasutab isik, kes koostab arveid. Reaalne ülesande täitmise kestus mõõdetakse teiste samale kliendile tehtavate ülesannete all. Seega algoritmi kontekstis ei tasu 0 minutiga (tegelik kestus mõni sekund) ülesandeid optimeerida. Lisaks tuli valideerimisest välja, et osad ülesanded on tihedalt omavahel seotud. Näiteks panagaarvestusi saab raamatupidaja teha alles siis, kui raamatupidamislikud ülesanded on tehtud. Algoritm oskas need seosed kasutaja harjumustest välja lugeda ning ülesandeid tööplaani paigutada nii, et ülesanne, mis on teise eelduseks, on tööplaanis eespool.

Raamatupidaja nägi tööplaani korrigeerimises väärtust ning tema hinnangul olid algoritmi poolt väljapakutavad uued alustusajad päris mitme ülesande puhul paremini tööplaani paigutatud kui varem. Kuid leidus ka ülesandeid, mille puhul raamatupidaja ei soovinud, et neid korrigeeritaks. Tegemist on teatud laadi ülesannetega, mis on raamatupidajale olulised, et need oleksid tööplaanis alates kuu esimesest päevast, kuna tema soov on nendega alustada nii kiiresti kui võimalik, aga alustusaeg sõltub suuresti kliendist ning enamasti on selliste ülesannete alustamisaja varieeruvus kuu lõikes suur.

7 Rakendamine

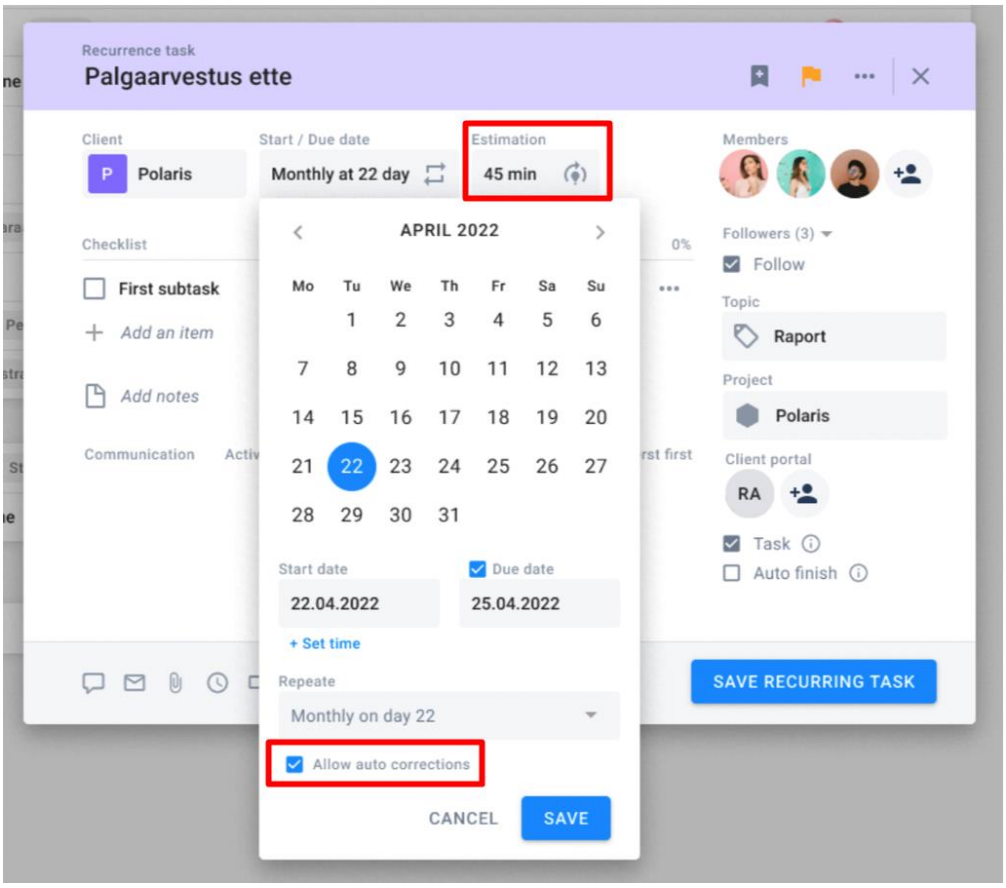
Selles peatükis tuleb juttu, kuidas plaanitakse rakendada lõputöös loodud funktsionaalsused Uku kasutajaliideses. Peatükis kasutatavad eesrakenduse vaated, on koostatud Uku UI/UX disaineri poolt.



Joonis 21. Näidiskasutaja ühe kliendi korduvate ülesannete vaade (koostatud UI/UX disaineri poolt)

Joonisel 21 on näha näidiskasutaja ühe kliendi baasülesandeid, mille põhjal luuakse vastavalt intervallile korduvad ülesanded. Üleval on kaks nuppu „Automaatne algusaeg“ ning „Ajahinnangute ennustamine“. „Automaatne algusaeg“ sisselülitamisel, on võimalik kõikidele kliendi kuupõhistele ülesannetele rakendada algoritmi poolt genereerituid kordumisreegleid. Selle tulemusel paigutatakse töölaual ülesanded ringi vastavalt korrigeeritud reeglitele. Välja lülitamisel saab tagasi minna vanadele reeglitele. Korrigeeritud reeglit hakatakse hoidma esialgsest reeglist eraldi, mis muudab vahetuse nende vahel võimalikuks. Analoogselt toimib ka „Ajahinnangute ennustamine“, mille

sisselülitamisel lisatakse kõigile ülesannetele edaspidi lõputöös loodava funktsionaalsuse poolt pakutav ajahinnang.



Joonis 22. Baasülesande vaade (koostatud UI/UX disaineri poolt)

Lisaks saab eraldi ülesande all nii „Automaatne algusaeg“ kui ka „Ajahinnangute ennustamine“ sisse või välja lülitada (vt joonis 22). Näiteks kasutaja soovib rakendada tööplaani korrigeerimist viiele ülesandele kuuest. Selleks saab kasutaja esmalt lülitada sisse kõigi ülesannete algusaja korrigeerimise ning eraldi ühe ülesande algusaja korrigeerimise välja lülitada või lülitada ühekaupa sobilikele ülesannetele algusaja korrigeerimise sisse.

8 Järeldused ja edasised sammud

Selles peatükis tehakse järeldus käesolevas töös loodud funktsionaalsuste kohta ning kirjeldatakse edasisi samme.

Tööplaani optimeerimine

Esialgu oli eesmärk luua kasutaja harjumustest parem tööplaan ehk süsteem ise paigutab ülesanded tööplaani vastavalt kestusele ja tähtajale. Algse hüpoteesi kohaselt tekkis mulje, et raamatupidaja määrab tööde teostamise järjekorra ise, kuid nii valideerimise kui erinevate arutelude käigus selgus, et tegelikult sõltub raamatupidajate töö väga palju tema kliendi dokumentide esitamisest ehk kaudselt raamatupidaja tööplaani koostavad tema kliendid. Sellest tulenevalt sai tuvastatud tegelik ärivajadus ning realiseeritud algoritm, mis paigutab ülesanded tööplaani vastavalt kasutaja harjumustele.

Enne algoritmi rakendamist oli ülesande keskmine distants planeeritud alustusaja ning tegeliku alustusaja vahel 9.1 päeva ning peale algoritmi rakendamist 4.4 päeva. See näitab, et bakalaureusetöös valminud algoritm suudab tuvastada kasutaja harjumusi ning nendest lähtuvalt planeerida ülesandeid tööplaani paremini. Tulemusi kinnitab ka valideerimine, kus raamatupidaja nägi tööplaani korrigeerimises väärtust ning tema hinnangul olid algoritmi poolt väljapakutavad uued alustusajad päris mitme ülesande puhul harjumusi järgivad ning paremini tööplaani paigutatud kui varem.

Ülesannete ajahinnangute ennustamine

Ajahinnangute ennustamiseks loodud masinõppel baseeruv mudel oli eksperiment, et teada saada, kui täpselt on võimalik ajahinnanguid ennustada. Lõputöös valminud mudelite tegelikke täpsuseid vaadates, on näha, et õigesti ennustamise tõenäosus on üsna madal 20-30%. Seega ei saa mudeli tulemusi sisendina mõne teise probleemi lahendamiseks rakendada.

Kuna ettevõtte üheks sooviks on tulevikus kasutajate töökoormust prognoosida ning finantsmeeskonda hoiatada, kui see kasvavas trendis liigub, siis on plaanis tulevikus proovida teistsugust lähenemist. Nimelt iga ülesande kestuse eraldi ennustamise asemel

proovida ennustada kogu kliendile kuluvat aega. Kui igale ülesandele leida eraldi ajahinnang, siis paratamatult tekib ennustusvigu ning ühe kliendi lõikes ajahinnanguid summeerides viga suureneb. Kui aga ennustada kogu kliendile kuluvat aega, siis on tegemist ühe ajahinnanguga, millel saab viga tekkida. Sellest tulenevalt on hüpoteesiks, et kliendile kogu kuluva aja ennustamine on täpsem kui ülesannetele eraldi ajahinnangute ennustamine ning nende summeerimine.

Edasised sammud

Põhiliseks eesmärgiks on alustada lõputöös loodud funktsionaalsuse integreerimist põhirakendusega kohe, kui Uku avalik rakendusliides valmis saab. See loob võimaluse testkeskkonnas rohkematel raamatupidajatel algoritmi headust valideerida ning täienduste ja paranduste soovitusi jagada. Kuna arenduse käigus oli keeruline näha tervikpilti, siis kindlasti ühe sammuna tuleks uuesti loodud funktsionaalsusele otsa vaadata ning seda refaktoreerida. Kolmanda sammuna on plaanis luua kliendipõhine ajahinnangu mudel kogu kestuse ennustamiseks ning võrrelda ennustustäpsust käesolevas töös saavutatud täpsustega.

9 Kokkuvõte

Käesoleva töö raames on valminud algoritm, mis suudab kasutaja korduvaid ülesandeid paigutada tööplaani selliselt, et need läheksid paremini kokku kasutaja harjumustega. Lisaks valmisid masinõppe mudelid ülesannete ajahinnangu ennustamiseks. Loodud funktsionaalsus täidab püstitatud eesmärgid ning on kergesti edasi arendatav. Praeguseks ei ole funktsionaalsus integreeritud põhirakendusega, sellega alustakse peale lõputööd.

Algselt oli plaanis luua algoritm, mis koostaks kasutajale parema tööplaan ehk süsteem paigutaks ülesanded vastavalt nende kestustele ja tähtajale tööplaani, sõltumata sealjuures kasutaja varasematest harjumustest. Andmeid analüüsides ning CH Konsultatsioonid OÜ-ga täpsemaid vajadusi kaardistades selgus, et selline funktsionaalsus ei ole tegelikult võimalik ja vajalik, kuna raamatupidaja tööde järjekord sõltub vägagi palju tema klientidest. Seega sai lõputöö käigus tehtud kindlaks reaalne kasutaja vajadus ning realiseeritud vastav algoritm.

Ülesannete ajahinnangu ennustamisel oli algselt eesmärk toetada tööplaani algoritmi, kuid ärivajaduste muutusel lõputöö jooksul, ei olnud algoritmi tööks ülesannete ajahinnanguid enam tarvis. Seega sai püstitatud uus eesmärk, luua ülesannete ajahinnangute mudelid selleks, et tulevikus saaks kasutajate töökoormust prognoosida.

Käesolev töö andis parema ettekujutuse põhirakenduse andmetest ning nende tegelikust sisust, lisaks sain palju uusi teadmis masinõppe valdkonnast. Suhtlemine Uku klientidega andis võimaluse mõista paremini raamatupidamisprotsesse ning kasutajate tegelikke vajadusi. Omandatud teadmisi saab väga edukalt rakendada nii lõputöös loodud funktsionaalsuste täiendamisel kui ka tulevikus uute probleemide lahendamisel.

Kasutatud kirjandus

[1] C. Richardson, „What are microservices?,“ [Võrgumaterjal]. Available: https://microservices.io/. [Kasutatud 25 Aprill 2022].

[2] P. S. Foundation, „The Python Tutorial — Python 3.7.3 Documentation,“ [Võrgumaterjal]. Available: https://docs.python.org/3/tutorial/index.html. [Kasutatud 25 Aprill 2022].

[3] J. s.r.o, „PyCharm,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/pycharm/. [Kasutatud 25 Aprill 2022].

[4] Scikit-learn, „Scikit-learn Getting started,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/getting_started.html. [Kasutatud 25 Aprill 2022].

[5] NumPy, „What is NumPy?,“ [Võrgumaterjal]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Kasutatud 25 Aprill 2022].

[6] Matplotlib, „Matplotlib: Visualization with Python,“ [Võrgumaterjal]. Available: https://matplotlib.org/. [Kasutatud 25 Aprill 2022].

[7] Gensim, „models.doc2vec – Doc2vec paragraph embeddings,“ [Võrgumaterjal]. Available: https://radimrehurek.com/gensim/intro.html . [Kasutatud 25 Aprill 2022].

[8] Gensim, „Doc2Vec Model,“ [Võrgumaterjal]. Available: https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html. [Kasutatud 25 Aprill 2022].

[9] I. Docker, „Docker overview,“ [Võrgumaterjal]. Available: https://docs.docker.com/get-started/overview/. [Kasutatud 25 Aprill 2022].

[10] MySQL, „What is MySQL?,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Kasutatud 25 Aprill 2022].

[11] SQLAlchemy, „The Python SQL Toolkit and Object Relational Mapper,“ [Võrgumaterjal]. Available: https://www.sqlalchemy.org/. [Kasutatud 25 Aprill 2022].

[12] Redis, „Introduction to Redis,“ [Võrgumaterjal]. Available: https://redis.io/docs/about/. [Kasutatud 25 Aprill 2022].

[13] S. Gupta, „TensorFlow vs. Scikit-Learn: How Do They Compare?,“ 7 Aprill 2017. [Võrgumaterjal]. Available: https://www.springboard.com/blog/data-science/scikit-learn-vs-tensorflow/. [Kasutatud 25 Aprill 2022].

[14] D. K, „Why PyTorch Is the Deep Learning Framework of the Future,“ 30 Oktoober 2019. [Võrgumaterjal]. Available: https://blog.paperspace.com/why-use-pytorch-deep-learning-framework/. [Kasutatud 25 Aprill 2022].

[15] A. C. EESTI, „e-krediidiinfo,“ [Võrgumaterjal]. Available: https://www.e-krediidiinfo.ee/. [Kasutatud 25 Aprill 2022].

[16] Wikipedia, „Web scraping,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Web_scraping. [Kasutatud 25 Aprill 2022].

[17] M. Thakur, „Log Normal Distribution,“ [Võrgumaterjal]. Available: https://www.wallstreetmojo.com/log-normal-distribution/. [Kasutatud 25 Aprill 2022].

[18] Scikit-Learn, „sklearn.linear_model.LassoLars,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLars.html. [Kasutatud 26 Aprill 2022].

[19] Scikit-Learn, „sklearn.linear_model.Lasso,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. [Kasutatud 26 Aprill 2022].

[20] Scikit-Learn, „sklearn.linear_model.Ridge,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html. [Kasutatud 25 Aprill 2022].

[21] Scikit-Learn, „sklearn.neighbors.KNeighborsRegressor,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html. [Kasutatud 26 Aprill 2022].

[22] Scikit-Learn, „sklearn.linear_model.LinearRegression,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. [Kasutatud 26 Aprill 2022].

[23] Scikit-Learn, „Varying regularization in Multi-layer Perceptron,“ [Võrgumaterjal]. Available: https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mlp_alpha.html . [Kasutatud 26 Aprill 2022].

[24] Dan Jurafsky and James H. Martin, Speech and Language Processing (3rd ed. draft), 2021.

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

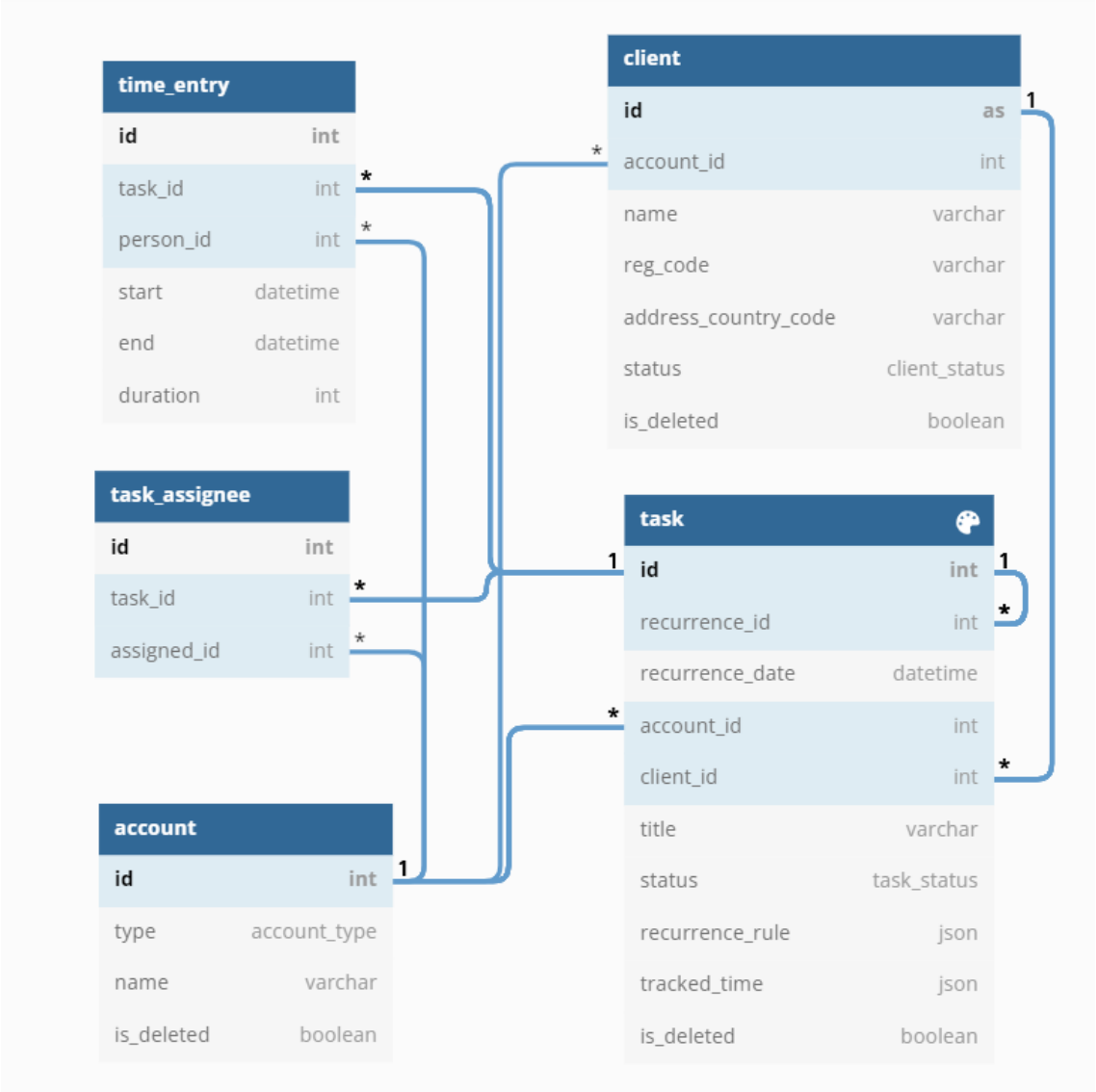
Mina, Marten Tamme

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Finantsmeeskondade tööde planeerimine tööhaldustarkvaras Uku“, mille juhendaja on Priit Järv
   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.

25.05.2022

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 – Käesolevas töös kasutatavad põhirakenduse andmebaasi tabelid



Joonis 23. Käesolevas töös kasutatavad põhirakenduse andmebaasi tabelid

Lisa 3 – Kordumisreegli nihked

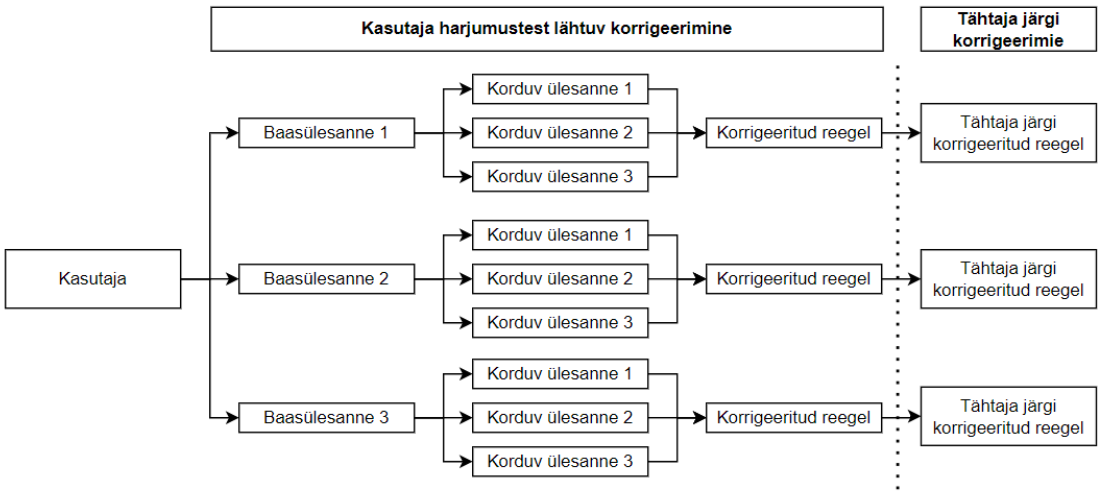
Jaanuar 2022						
Esmaspäev	Teisipäev	Kolmapäev	Neljapäev	Reede	Laupäev	Pühapäev
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Joonis 24. Nihke -1 järgi nädalavahetusele langeva ülesande korrigeerimine

Jaanuar 2022						
Esmaspäev	Teisipäev	Kolmapäev	Neljapäev	Reede	Laupäev	Pühapäev
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Joonis 25. Nihke +1 järgi nädalavahetusele langeva ülesande korrigeerimine

Lisa 4 – Korrigeerimisprotsess



Joonis 26. Korrigeerimisprotsess