TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jaanika Raik 179129IAIB

ANDMEKAEVE PARAMEETRITE VALIKU MÕJU TARGA LIFTI LÕPPKORRUSE ENNUSTAMISEL

bakalaureusetöö

Juhendaja: Uljana Reinsalu
Teadur

Tallinn 2020

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: Jaanika Raik

18.05.2020

Annotatsioon

Käesoleva töö käigus vaadeldakse Tallinna Tehnikaülikooli Infotehnoloogia hoone uksepoolset lifti. Eesmärk on ennustada võimalikult täpselt korrust, kuhu inimene soovib reisida. Luuakse programm, mis võrdleb nelja erineva parameetri mõju statistilise ennustuse täpsushinnangule. Programm väljastab statistilise ennustuse täpsushinnanguid, mis tekivad parameetreid kombineerides.

Töö abil soovitakse parandada liftireisi mugavust ja turvalisust. Kui reisijad ei pea enam sihtkorruse nuppu vajutama, väheneb viiruste leviku risk. Lisaks paraneb nende inimeste olukord, kellel pole mugav käsi kasutada, või kellel on kiire.

Neljast vaadeldud parameetrist osutus kõige olulisemaks kasutatud klassifikaator. Parima täpsushinnangu väljastas kombinatsioon, mille puhul oli klassifikaatoriks otsustuspuu, nädalapäeva ei arvestatud ning ööpäev oli jagatud neljaks osaks. Kombinatsioon oli treenitud mahuka andmestiku peal (38436 rida) ning arvestati reisija vajutatud suunanuppu. Võrreldi viit klassifikaatorit: pertseptron, logistiline regressioon, *k* naabri põhine klassifikaator, tugivektor-masin ja otsustuspuu.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 5 peatükki, 8 joonist, 15 tabelit.

Abstract

Impact of Choosing Data Mining Parameters on the Predicted Destination Floor of a Smart Elevator

In the course of this work, an elevator of the Information Technology building of the Tallinn University of Technology was examined. The aim was to predict as accurately as possible the floor to which a person wishes to travel. A program that compares the effect of four different parameters on the accuracy of the statistical prediction was implemented. The program outputs accuracies of prediction from combining different parameters.

The work aims to improve the comfort and safety of elevator journeys. Pressing elevator buttons increases the risk of contracting a viral infection. In addition, it helps people with disabilities and those in hurry.

Of the four parameters observed, the classifier used was the most important one. The best accuracy was given by a decision tree classifier and day of the week was not one of the parameters. The selected parameters were as follows: time of the day (day divided into four parts) and recorded direction of travel (elevator call buttons pressed). A large dataset of 38,436 rows was used to train the decision tree. Five classifiers were included into the comparison: perceptron, logistic regression, *k* neighbours classifier, support vector machine and decision tree.

The thesis is in Estonian and contains 31 pages of text, 5 chapters, 8 figures, 15 tables.

Lühendite ja mõistete sõnastik

SCAN	Lifti planeerimisalgoritm
KONE	Soome tehnoloogiafirma
IBM	*International Business Machines*
KNeighborsClassifier	*k* naabri põhine klassifikaator
svm	*Support vector machine*, tugivektor-masin
RGB (kaamera)	*Red-blue-green*, punane-roheline-sinine
PostgreSQL	Andmebaasisüsteem
SVC	*Support vector classifier*, tugivektor-klassifikaator
IoT	*Internet of things*, asjade internet
scikit-learn	Masinõppe teek
Jupyter Lab	Töövihiku vormis andmekaeve keskkond

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust

1.1.1 Andmekaevest üldiselt

Andmekaeve mõjutab järjest enam erinevaid eluvaldkondi ja igapäevaolukordi. See aitab vastu võtta otsuseid ning optimeerida erinevaid protsesse.

Andmekaeve protsessi võib defineerida kui andmete otsimist, kogumist, filtreerimist ja analüüsimist hõlmavat protsessi. Kogutud andmeid saab analüüsida ning nende abil saab tuvastada nende abil suhteid ja leida probleemidele lahendusi. Suurandmeid kasutavad firmad, valitsused ja suurorganisatsioonid, et arendada oma äri ja uuringuid [21].

Andmekaeve on mitmel põhjusel kasulik. Selle abil luuakse andmetest tähendusega informatsioon, mis aitab asutustel õigeid otsuseid vastu võtta. Saab analüüsida klientide käitumist ja perspektiive. Viimane aitab muuta ettevõtet edukamaks ning juhtida äri [21].

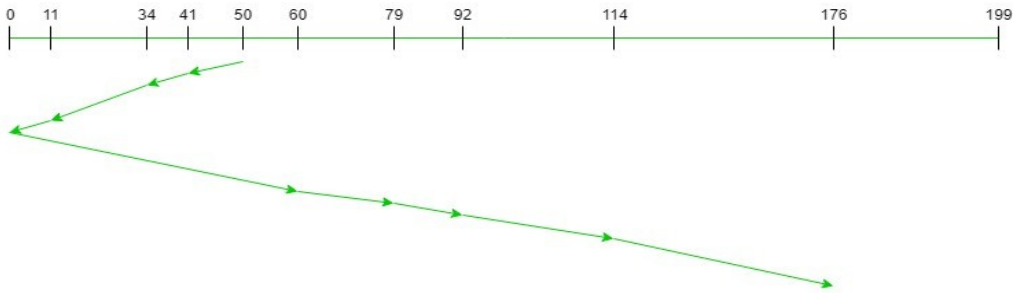
Andmeteadurid alustatvad tööd nõuete defineerimisega. Järgmise sammuna uuritakse nõudeid ning konverteeritakse andmed metaandmeteks. Seejärel valmistatakse andmed ette modelleerimiseks ning viiakse läbi modelleerimine. Lõpuks antakse tulemuste põhjal hinnangud ning kujutatakse andmeid visuaalselt [21].

Paljusid teenuseid saab teostada andmekaeve abil. Näiteks saab analüüsida meediaväljaannete avaldatud uudiseid, et selgitada välja trende ja kriitilisi teemasid. Samuti saab analüüsida firma klientide ja konkurentide käitumist. Nii on firmal võimalus saavutada turul parem positsioon [21].

Andmekaeve on maailmas kasvav tööstusharu. Andmeteaduritelt eeldatakse, et nad peavad kohtlema andmeid sobival viisil ning lähenema tööle õigesti. Eesmärgiks on saada kasulikku ja vigadeta informatsiooni [21].

1.1.2 Lifti toimimine

Liftid kasutavad standartset algoritmi *elevator algorithm (SCAN)*. Lift hakkab ühest liftišafti otsast liikuma ja liigub teise otsa, võttes teel peale reisijaid, kes selles suunas liiguvad. Lõppu jõudes vahetab lift suunda [18].



Joonis 1. Lifti algoritmi näide [18]

Näiteks, kui tellitud korruste järjekord on 176, 79, 34, 60, 92, 11, 41, 114 ning lift seisab 50ndal korrusel on otsimisjärjekord järgmine: 41, 34, 11, 0, 60, 79, 92, 114, 176 [18].

1.1.3 Seos varasemate projektidega ja tähtsus

Projekte, mille eesmärk on lift senisest targemaks teha, on tehtud mitu. Näiteks on tehnoloogiafirma KONE loomas järjest targemaid lifte. Üles-alla nuppude asemel kasutatakse puutetundlikku ekraani. KONE disainidirektor Timo Tiainen ütleb: „Hooned saavad targemaks ja automaatsemalt töötavaks. Nad suudavad ise mõelda ning vastata automaatselt inimeste vajadustele. [2]" KONE üheks saavutuseks on Hamburgis püstitatud kaarekujuline lift, mis pakub reisijatele võimalust rännata „lõpmatus" liftis. Lisaks on firma aastaid IBM'iga koostööd teinud, et saaks konkreetsetele liftidele spetsiifilist hooldussüsteemi luua, mis avastab probleeme ja riske juba enne, kui hooldupersonal lifti külastada jõuab [20].

Hyundai liftid tõotavad tarka ühendust inimesega. Kliendile lubatakse tõhusaid ja mugavaid tarku süsteeme. Hyundai pakub erilist tüüpi maas asuvat liftinuppu, mida saab käe asemel jalaga vajutada. See aitab inimesi, kellel on mõlemas käes asjad või kellel on terviseprobleemi tõttu raske käsi kasutada. Teine eriline Hyundai nuppude liik on *touch-less* nupud. Nende puhul pole vaja sõrme täielikult nupu vastu panna, et hoida hügieeni ja vältida infektsiooni. Samuti pakub firma viirusevastasest materjalist käetugesid, kuna need levitavad vastasel juhul kergesti viirust. Liftisiseselt kasutatakse

spetsiaalset valgustust, mis rahustab klaustrofoobikuid, jättes mulje nagu inimene ei viibikski tegelikult nii kitsas ruumis. Soovimatute isikute hoonesse lubamise vältimiseks kasutatakse biomeetrilise tuvastusena silmaiirist [19].

Väljaanne *Future IoT* progrnoosib 2030. aastaks liftidele oma mõistust. Muuhulgas arvatakse, et mobiilirakenduste kasutamine lifti kutsumiseks muutub tavaliseks. Samuti peetakse tõenäoliseks, et liftid hakkavad suhtlema ööpäevaringselt serveritega. Lisaks tuuakse välja, et nüüd osatakse toota lifte, mis on kuni 90% võrra elektrisäästlikumad kui 30 aastat tagasi [20].

1.2 Probleem

Üldlevinud lifti algoritm ei paku reisijatele parimat lahendust, kuna ta ei arvesta paljude näitajatega. Kõige tõenäolisem korrus, mida inimene külastada soovib, sõltub paljudest parameetritest. Inimesed soovivad optimaalselt aega säästa ja igapäevaelu protsesse optimeerida.

Käesoleva töö abil soovitakse liftireis kasutajale mitmest küljest mugavamaks muuta. Õige sihtkorruse ennustus võimaldab reisijal pääseda vajadusest panna liftikorrusenupu vajutuse ajaks asjad käest. Samuti on see kasulik osadele terviseprobleemidega reisijatele, kellel on näiteks käevigastus. Lisaks aitab nupuvajutuse vältimine vähendada viiruste levimise tõenäosust. Kui reisija ei pea vajutama lõppkorruse nuppu aitab see kokku hoida ka tema aega.

Vaadeldav objekt on vaikimisi tavaline lift ning töö algushetkeks pole implementeeritud korralikku ennustusalgoritmi. Tal on vaid näotuvastus, mis võimaldab inimesi üksteisest eristada.

1.3 Töö eesmärk

Käesoleva töö eesmärk on ennustada võimalikult täpselt korrust, kuhu inimene soovib reisida, ning analüüsida, milline parameetrite valik annab kõige parema tulemuse. Vaatluse alla võetakse järgmised parameetrid:

- Teadmine selle kohta, kas reisija vajutas enne lifti sisenemist üles- või alla liikumise nuppu
- Vaadeldav ajavahemiku liik (näiteks *täisarvuline tund; tööpäev või puhkepäev*, mis tähendab, et ööpäev on jagatud 24ks täistunniks ning arvestatakse ka seda, kas on töö- või puhkepäev)
- Ennustamiseks kasutatud klassifikaator
- Tööks kasutatud andmestiku maht

Protsessi käigus peab valmima programm, mis väljastab erinevaid parameetreid kombineerides tekkinud täpsushinnangud. Lähteandmed on kogutud enne töö algust.

1.4 Ülevaade tööst

Järgnevalt kirjeldatakse töös kasutatavat metoodikat. Tutvustatakse uuritavat lifti, kasutatavat tarkvara ja sisseehitatud algoritme ning koodi arendusprotsessi. Viimane koosneb andmete pärimisest andmebaasist, statistilisest ennustusest ja selle põhjal täpsushinnangute leidmisest ning saadud täpsushinnangute töövihikutesse sisestamisest. Seejärel avaldatakse uurimuse tulemused kõigi nelja parameetri kohta eraldi ning võrreldakse nende mõju täpsushinnangule. Lõpuks analüüsitakse saadud tulemusi ja antakse neile hinnang. Tehakse ettepanekud projekti edasisteks plaanideks ning antakse soovitused selle kohta, milliseid parameetreid tasub hea täpsuse huvides valida.

2 Metoodika

2.1 Objekt

Vaadeldavaks objektiks on lift, mis asub Tallinna Tehnikaülikooli Infotehnoloogia hoone sissepääsu juures. Hoonel on kaheksa korrust, sealhulgas maa-alune nullkorrus. Antud lift on valitud projekti Tark Lift katseobjektiks. Projekti läbi viivasse meeskonda kuuluvad Mairo Leier, Andri Riid, Tanel Alumäe, Uljana Reinsalu, René Pihlak, Risto Heinsar, Sven Vainküla, Rait Kurg ja Andres Udal.

Joonis 2. Vaadeldav lift

Reisijal on võimalus kasutada häälkäsklustel põhinevat abistajat. Seda on lihtne kohandada erinevatele keeltele. Reisija alustab vestlust sõnadega „Hey, elevator”. Seejärel on tal võimalus anda liftile korraldusi, näiteks „Mine neljandale korrusele.”. Lisaks on reisijal võimalus küsida kellaaega ja kuupäeva ning ilmateadet [1, lk 2].

Lifti on paigutatud Basler RGB kaamera, et viia läbi näotuvastust. See asub nurgas 1,55 meetri kõrgusel, mis on keskmine inimese silmade kõrgus. Liftis asuvad ka neli süvakaamerat, mis aitavad tuvastada reisija asukohta liftis. Lisaks asuvad kabiinis

mikrofon ja kõlar, mida saab kasutada häälkäskluste tarbeks [1, lk 2]. Kõik eelkirjeldatud seadmed on kujutatud joonisel 2.

Projekti käigus koguti iga liftireisi kohta järgmised andmed:

- Reisija profiil
- Reisija nägu identifitseeriv tempel
- Reisija lausutud audiopäringud
- Reisija reiside ajalugu, kus iga reisi kohta on teada:
    - Reisi algkorrus
    - Reisi lõppkorrus
- Lifti reisilogi, mis sisaldab kõiki sellega tehtud reise

Kogutud andmeid hoiustati PostgreSQL andmebaasisüsteemi 11.4 versiooni abil.

Vaatamata loodud automaatsele süsteemile saab reisija ikkagi ise vajutada selle korruse nuppu, kuhu ta reisida soovib [1, lk 3]. See tähendab, et kui lift siiski ennustab lõppkorruse valesti, saab reisija viga parandada.

2.2 Tööriistad

Antud töös kasutatakse mitmesuguseid andmekaeve tööriistu, et andmete põhjal võimalikult hea täpsusega klassi ennustada. Programm kirjutatakse *Jupyter Lab* keskkonnas Pythoni keeles. Imporditekse *scikit-learn*'i tööriistu.

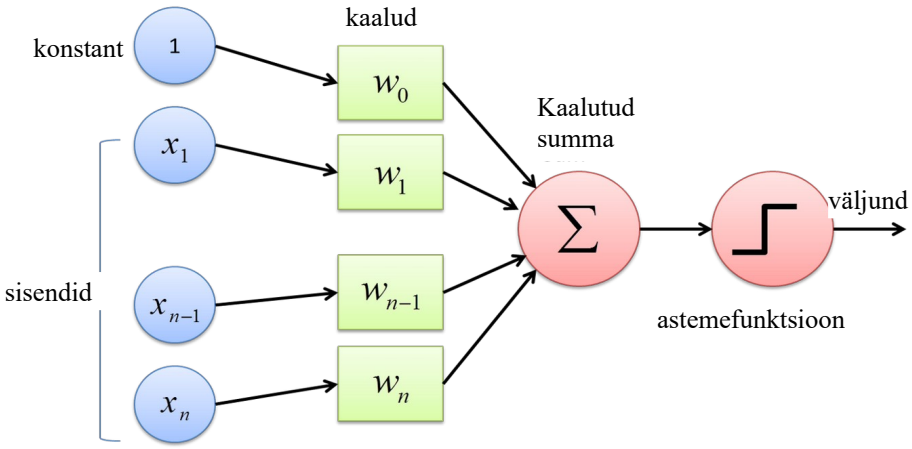
Masinõppes on ennustatavaks klassiks tunnus, mida soovitakse ennustada ning millel on lõplik hulk võimalikke väärtusi [23]. Käesolevas töös on ennustatavaks klassiks liftireisi lõppkorrus ning võimalike väärtuste hulk on 0, 1, 2, 3, 4, 5, 6 ja 7.

Klassivektori iga element on talle vastava objekti klass. Andmemaatriksi iga rida on talle vastava objekti tunnuste loetelu. Käesolevas töös koosneb klassivektor iga vaadeldud reisi lõppkorrustest ning andmemaatriksi rida sisaldab talle vastava reisi andmeid. Töö alguses tuleb nii andmemaatriks kui ka klassivektor jagada kaheks osaks: treening- ja testandmed [14].

Klassifikaator on ennustusmudel, mida treeningandmete peal treenitakse. Seejärel saab mudel testandmete põhjal objekti klassi ennustada. Ennustuse kvaliteeti iseloomustab täpsushinnang, mis näitab, mitu protsenti ennustustest läks õigesti [14]. Alljärgnevalt tutvustatakse kõiki viit töös kasutatavat klassifikaatorit.

2.2.1 Pertseptron

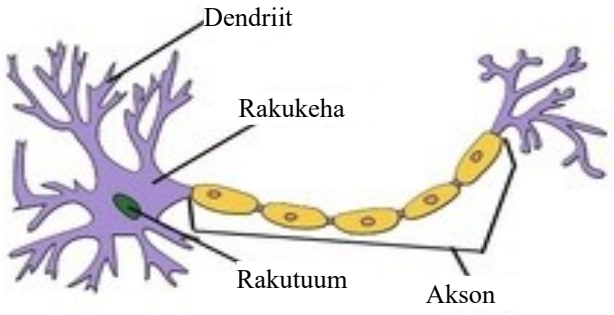
Pertseptron on lineaarne klassifitseerimisalgoritm, mida kasutatakse juhendatud masinõppes [12]. Vaikimisi on ta seadistatud nii, et ta uuendab oma mudelit ainult vigade põhjal. Samuti ei kasutata vaikimisi õppimisammu määramist ega regulariseerimist [13]. Pertseptron sobib tööks kõige paremini siis, kui valida on kahe ennustatava klassi vahel [12].



Joonis 3. Pertseptron [12]

Algoritm algab sellega, et iga sisend (x_n) korrutatakse talle vastava kaaluga (w_n). Kõik saadud väärtused liidetakse kokku, et leida summaarne sisend (z). Viimane antakse sisendiks aktivisatsioonifunktsioonile, mis tagastab nulli, kui sisend on negatiivne, ning ühe, kui sisend on positiivne või võrdne nulliga [12].

$$z = w_1 x_1 + ... + w_n x_n \quad (1)$$

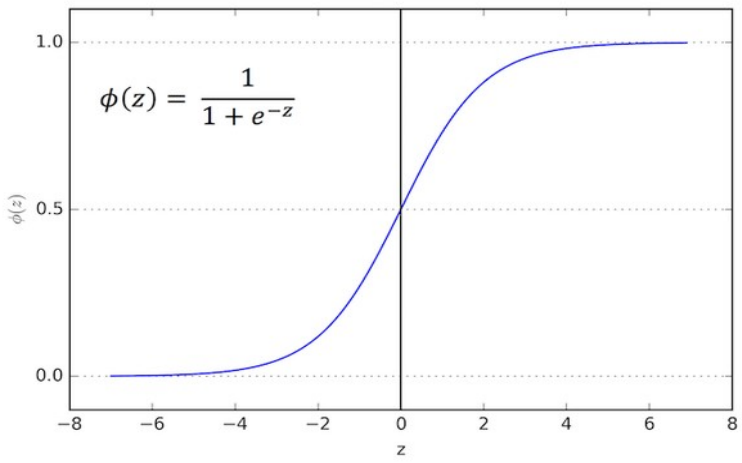


Joonis 4. Närvirakk ehk neuron [16]

Pertseptron meenutab närvirakku, kuna tema sisendid sarnanevad dendriididega ning väljastatav summa aksoniga [14]. Neuron reageerib dendriitide kaudu välistele stiimulitele ning akson juhib rakust väljuva elektrisignaali sünapsi elundisse või teise närvirakku [25].

2.2.2 Logistiline regressioon

Logistiline regressioon on lineaarne klassifitseerimismudel [15]. Tal on peenemad õppimisreegel ja aktivisatsioonifunktsioon kui pertseptronil. Erinevalt pertseptronile pole logistilise regressiooni väljundiks mitte klass vaid tõenäosus nullist üheni. Õppeprotsess järgib gradientlaskumist: mida rohkem on ennustus valesti läinud, seda rohkem kaale muudetakse [14].



Joonis 5. Logistiline regressioon [17]

Summa (z) on logistilise regressiooni puhul sündmuse ja tema vastandsündmuse tõenäosuste suhe

$$\frac{p}{1-p} \tag{2}$$

Eelmainitud suhte logaritm loetakse meelevaldselt võrdseks avaldisega

$$w^T x \tag{3}$$

Avaldise

$$\ln \frac{p}{1-p} \tag{4}$$

saab tõenäosuseks tagasi teisendada

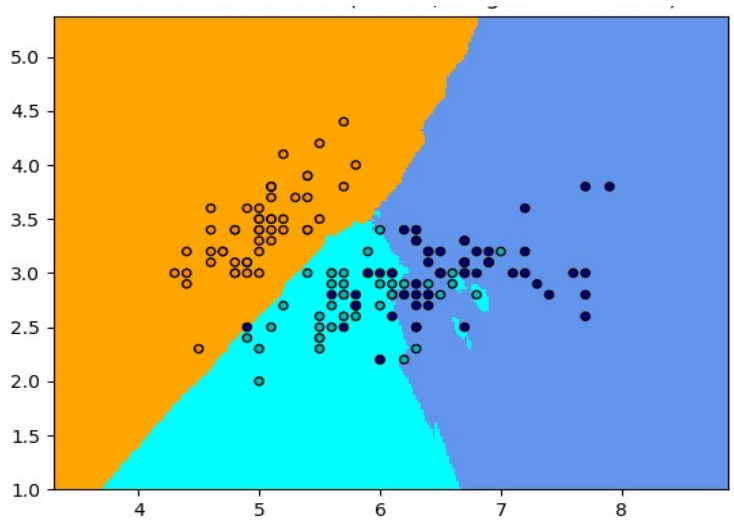
$$(z) = \frac{1}{1+e^{-z}} \tag{5}$$

sigmoidfunktsiooni

abil, mida kasutataksegi aktivisatsioonifunktsioonina [14].

2.2.3 *k* naabri põhine klassifikaator

Naabripõhine klassifikatsioon hoiab mälus instantse, seejuures koostamata üldist sisemist mudelit. Selleks, et klassifitseerida tuleb läbi viia enamushääletus iga punkti lähimatel naabritel. Igale päringupunktile määratakse andmeklass selle järgi, et andmeklassil oleks enim esindajaid punkti lähimate naabrite hulgas [9].

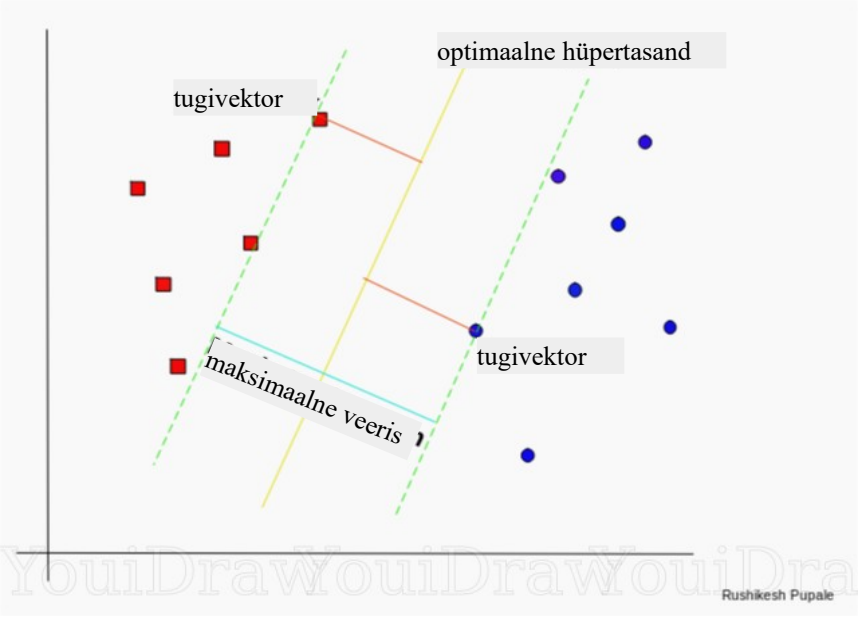


Joonis 6. *k* naabri põhine klassifikaator [11]

k naabri põhine klassifikaator (*scikit-learn*'is *KNeighboursClassifier*) on üks võimalik ning enim kasutatud naabripõhise klassifikatsiooni implementatsioone [9]. Muutuja *k* on kasutaja valitud täisarv ning mudel õpib *k* lähima naabri põhjal. Pythoni teegis *scikit-learn* on *k* vaikimisi väärtus 5 [10]. Suur *k* väärtus vähendab andmetöötluses müra teket, kuid seevastu hägustab klassifikatsiooni piire [9].

2.2.4 Tugivektor-masin

Tugivektor-masin on andmekaeve ning masinõppe tööriist. Tegemist on lineaarse klassifitseerimismudeliga, mis saab lahendada nii lineaarseid kui ka mittelineaarseid probleeme. Algoritmi käigus tõmmatakse kahe klassi vahele neid eraldav sirge või hüpertasand, millele lähimaid punkte kummastki klassist kutsutakse tugivektoriteks. Protsessi lõpuks saavutatakse maksimaalne kaugus tugivektorite ning sirge või hüpertasandi vahel [5].



Joonis 7. Tugivektor-masin [5]

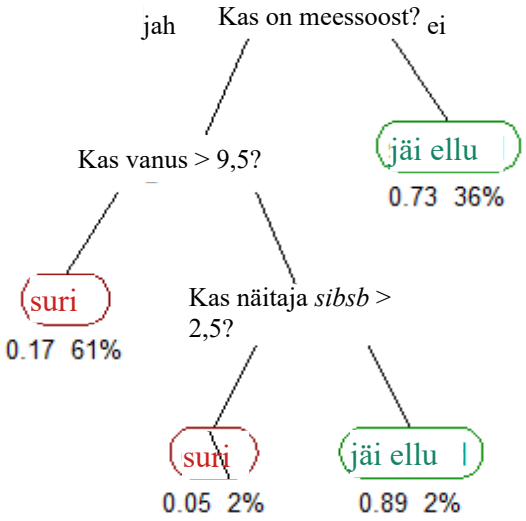
Tugivektor-masinal on mitmeid eeliseid. Ta suudab funktsioneerida efektiivselt ka paljude dimensioonidega ruumis. See kehtib ka siis, kui dimensioone on rohkem kui valimit. Tänu tugivektoritele kasutab masin efektiivselt mälu. Tugivektor-masin pakub mitmekesist tuumade funktsioonide valikut. Antud tööriista puuduseks on võimetus tagastada tõenäosushinnanguid. Valimi mahust oluliselt suurema dimensioonide arvu

korral tuleb tuumafunktsioone ja reguleerimistingimusi valides vältida liiga täpseks muutmist [6].

Käesolevas töös kasutatakse tugivektori klassifikaatorit (lühendatult *SVC*). Sklearn'is tuleb see importida *svm* mooduli kaudu. Soovi korral saab klassifikaatori parameetrites seadistada tuumafunktsioone [7].

2.2.5 Otsustuspuu

Otsustuspuid kasutatakse nii regressioonanalüüsis kui ka klassifitseerimises. Otsustuspuu algab juurest, mille juurde on märgitud tingimus. Kui see tingimus vastab tõele, liigutakse vasakule, ning kui tingimus osutub valeks, valitakse parem haru. Järgmises sõlmes kordub sama protsess. Algoritm peatub puu lehes ehk terminaalsõlmes [3].



Joonis 8. Otsustuspuu [3]

Otsustuspuude üheks eeliseks on nende visuaalne olemus, mis võimaldab neid kergesti mõista ja interpreteerida. Tänu logaritmilisele keerukusele on algoritm kiire. Otsustuspuu kasutab valge kasti mudelit, mis tähendab, et tema sees tekkivad vahetulemused on inimesele vaadeldavad. Algoritm suudab lahendada mitme väljundiga probleeme [4].

On oht, et otsustuspuu koostab liiga täpse mudeli ning ei suuda tulemust piisavalt hästi üldistada. Teiseks antud algoritmi puuduseks on ebastabiilsus: väike erinevus andmetes võib anda tulemuseks täiesti teistsuguse puu. Optimaalse otsustuspuu koostamise ülesanne on NP-keerukas [4].

Käesolevas töös kasutatakse ja võrreldakse kõiki viit eelmainitud klassifikaatorit. Nende parameetrid on seadistatud vaikimisi.

2.3 Protsess

2.3.1 Andmete pärimine

Käesoleva töö tarbeks vajalike algandmete pärimiseks kasutati kahte PostgreSQL andmebaasi: *diplom* ja *diplom_uus*. Kumbagi hoitakse rakenduses DBeaver. Andmebaaside struktuur on ühesugune, kuid *diplom_uus* sisaldab pikema ajavahemiku jooksul kogutud andmeid (vaatlusperiood algas 20.09.2019 ning lõppes 04.03.2020). *diplom* seevastu sisaldab ainult kuni 25.10.2019 kogutud reise, kuid vaatlusperioodi algus on sama. *diplom* sisaldab 4329 reisi andmeid ning *diplom_uus* 38436 reisi andmeid.

Pythoni andmebaasiga ühendamiseks kasutatakse teeki *psycopg2*. Defineeritakse ühenduse objektid *connection* ja *connection2*. Esimest kasutatakse *diplom* andmebaasiga ja viimast *diplom_uus* andmebaasiga ühendumiseks.

Funktsioonis *get_data* viiakse funktsiooni *execute* kasutades läbi päring *postgreSQL_select_query*. Selle käigus ühendatakse *LEFT JOIN*'i abil tabelid *trips* ja *profiles_on_trips*. Tabelite ehitust ja sisu on kirjeldatud lisades (*Lisa 2*). Hangitakse järgmised andmed:

- *profile_id* – reisija unikaalne identifikaator
- *start_floor* – reisi algkorrus
- *destination_floor* – reisi lõppkorrus

- *extract(hour from t.time)* – ajatemplist eraldatakse kellaaja täisarvuline tund kasutades funktsiooni *extarct*
- *extract(dow from t.time)* – ajatemplist eraldatakse nädalapäev kasutades funktsiooni *extarct*

Päritud kirjed salvestatakse listi *test_records*. Selle põhjal luuakse funktsioonis *destination_list* lõppkorruste klassivektor *y*. Välja jäetakse kirjed, kus puudub reisija identifikaator.

Funktsioonis *list_of_lists* luuakse kirjete põhjal andmemaatriks *X*, kuhu sisestatakse järgmised andmed:

- *person* – reisija unikaalne identifikaator
- *start* – reisi algkorrus
- *time* – ööpäeva osa
  - hommik (1): 7.00-12:59
  - lõuna (2): 13:00-15:59
  - õhtu (3): 16:00-19:59
  - öö (0): 20:00-6:59
- *hour* – kellaajast eraldatud täisarvuline tund
- *weekday* – nädalapäev
- *is_weekend* – kaks võimalikku väärtust:
  - tööpäev (0)
  - puhkepäev (1)
- *pressed_up_button* – tõene, kui võib oletada, et reisija vajutas enne lifti sisenemist üles liikumise nuppu ning väär, kui selleks pidi olema alla liikumise nupp. Väärtus arvutatakse alg- ja lõppkorruse põhjal. Kui lõppkorrus on suurem, arvatakse reisija olevat vajutanud üles liikumise nuppu, ning väiksema lõppkorruse korral alla liikumise nuppu.

Käesolevas töös defineeritakse kuus ajavahemikuliiki:

- *ööpäeva osa; nädalapäeva ei arvesta*
- *ööpäeva osa; tööpäev või puhkepäev*
- *ööpäeva osa; arvestatakse nädalapäeva*
- *täisarvuline tund; nädalapäeva ei arvestata*
- *täisarvuline tund; tööpäev või puhkepäev*
- *täisarvuline tund; arvestatakse nädalapäeva*

2.3.2 Andmete põhjal ennustamine

Funktsioonis *main_process* defineeritakse sõnastik *dict*, mille võtmeks on ajavahemiku liigi kirjeldus sõnena ning väärtuseks tõeväärtuste list, kus on märgitud *False* nende veergude indeksitele vastavad elemendid, mida antud ajavahemik ei kasuta. Viimased eemaldatakse ajutiselt andmemaatriksist, et viimane vastaks ajavahemiku iseloomule.

Tabel 1. Ajavahemiku liikidele vastavad andmemaatriksi veerud

Ajavahemiku liik / andmemaatriksi veerg	***person***	***start***	***time***	***hour***	***weekday***	***is_weekend***	***pressed_up_button***
ööpäeva osa; nädalapäeva ei arvestata	*True*	*True*	*True*	***False***	***False***	***False***	*True*
ööpäeva osa; tööpäev või puhkepäev	*True*	*True*	*True*	***False***	***False***	*True*	*True*
ööpäeva osa; arvestatakse nädalapäeva	*True*	*True*	*True*	***False***	*True*	***False***	*True*
täisarvuline tund; nädalapäeva ei arvestata	*True*	*True*	***False***	*True*	***False***	***False***	*True*
täisarvuline tund; tööpäev või puhkepäev	*True*	*True*	***False***	*True*	***False***	*True*	*True*
täisarvuline tund; arvestatakse nädalapäeva	*True*	*True*	***False***	*True*	*True*	***False***	*True*

Üle ajavahemike itereeritakse *for*-tsükliga, mille käigus antakse iga ajavahemik argumendiks funktsioonile *predict_with_button*. Funktsiooni esimeses pooles eemaldatakse andmemaatriksist viimane veerg, milleks on reisija vajutatud nupp, et viia katse läbi seda parameetrit arvestamata. Teises pooles seevastu jäetakse see parameeter alles. Mõlemas pooles kutsutakse välja funktsioon *predictions*.

Funktsioonis *predictions* jagatakse kõigepealt andmed funktsiooni *train_test_split* abil test- ja treeningandmeteks. Seejärel viiakse katse läbi kõigi viie käesolevas töös kasutatud klassifikaatori peal:

- Pertseptron
- Logistiline regressioon
- *k* naabri põhine klassifikaator
- Tugivektor-masin
- Otsustuspuu klassifikaator

Iga klassifikaatori abil kutsutakse välja funktsioon *get_accuracy*, kus viiakse funktsiooni *predict* abil läbi statistiline ennustus ning tagastatakse funktsiooni *accuracy_score* abil konkreetsele katsele vastav täpsushinnang. Antud protsessi lõpp-produktiks on list, mille esimene element on ajavahemiku liigi kirjeldus sõnena (muutuja *option*) ning viimased viis kõigi viie klassifikaatori abil saavutatud täpsushinnangud.

2.3.3 Andmete salvestamine

Programmi väljundiks on täpsushinnangud protsentides. Nende salvestamiseks kasutati *xlwt* teegi tööriista *Workbook*, mille abil loodi kaks uut Exceli töövihikut: *small_dataset_results.xls*, kuhu sisestati väikese andmestiku peal treenitud tulemuste õigesti ennustamise tõenäosused, ning suure andmestiku peal treenitud täpsushinnangud salvestati faili *big_dataset_results.xls*. Kumbki töövihik jagati kaheks leheks: esimesel lehel on täpsushinnangud, mis on treenitud reisija vajutatud nuppu arvestamata, ning teine, kus on arvesse võetud ka seda, kas reisija vajutas enne lifti sisenemist üles- või

allaliikumisnuppu. Kummalegi lehele loodi tabel, kus ridadeks on ajavahemiku liigid ja veergudeks klassifikaatorid. Andmed sisestati ridade kaupa funktsioonis *list_to_sheet*.

3 Tulemused

Käesoleva töö käigus valmis Pythoni programm Jupyteri töövihiku näol. Kood algab kasutavate teekide importimisega: *slkearn*, *numpy*, *psycopg2* ja *xlwt*. Sellele järgnevad kõik defineeritud funktsioonid, mis sisaldavad ka dokumenteeritud kirjeldust. Viimane neist – *main_process* – viib ülejäänud funktsioonide abil töö läbi: loob ühenduse nõutud andmebaasiga, arvutab sealsete andmete abil mudeli täpsushinnangud ning seejärel väljastab need Exceli töövihikusse. Kood lõppeb sellega, et funktsioon *main_process* kutsutakse kaks korda välja: esimene kord antakse argumendiks *diplom* andmebaasile vastav ühendus ning teine kord *diplom_uus* andmebaasi oma.

Protsessi lõpuks valminud programm loob käivitamisel edukalt kaks uut Exceli töövihikut: fail *small_dataset_results.xls*, mis näitab 4329 andmepunktilise andmestiku peal treenides loodud täpsushinnanguid, ning fail *big_dataset_results.xls*, mis kasutab 38436 rida pikka andmestikku. Kumbki töövihik jaguneb kaheks leheks: esimesel neist on täpsushinnangud, mis on treenitud reisija vajutatud nuppu arvestamata, ning teise puhul on arvesse võetud ka seda, kas reisija vajutas enne lifti sisenemist üles- või allaliikumisnuppu. Kummalgi lehel asub tabel, kus on ridadeks ajavahemiku liigid ja veergudeks klassifikaatorid. Tabeli iga lahter on täidetud protsentides antud täpsushinnanguga.

Eelmainitud töövihikute põhjal saab võrrelda järgmiste parameetrite mõju täpsushinnangule:

- Teadmine selle kohta, kas reisija vajutas enne lifti sisenemist üles- või alla liikumise nuppu
- Vaadeldav ajavahemiku liik
- Ennustamiseks kasutatud klassifikaator
- Tööks kasutatud andmestiku maht

Halvim täpsushinnang oli 16,47% ning parim 63,85%. Täiesti juhuslikult valides oleks täpsushinnang 14,29%, sest hoonel on kaheksa korrust ning üks neist on välistatud, kuna isik ei saa soovida reisida samale korrusele, kus ta juba on.

$$\frac{1}{7} \cdot 100\,\% = 14{,}29\,\%$$

Seega on antud töö käigus saadud täpsushinnangud juhuvalikust 1,15 kuni 4,47 korda paremad.

Järgnevalt võrreldakse eraldi kõigi nelja vaadeldud parameetri mõju täpsushinnangule.

3.1 Nupuvajutuse mõju täpsushinnangule

Programmi tulemusena valminud töövihikud näitavad esimesel lehel täpsushinnangud, mis saadi vajutatud nuppu arvestamata. Teisel lehel seevastu arvestati ka seda, kas reisija vajutas üles- või allaliikumisnuppu.

Pea kõikidel juhtudel andis nupuvajutusega versioon parema tulemuse võrreldes nupuvajutuseta variandiga. Ainsateks eranditeks olid kaks pertseptroni täpsushinnangut:

- Väikese andmestiku puhul ajavahemik *täisarvuline tund; nädalapäeva ei arvestata*
- Suure andmestiku puhul ajavahemik *täisarvuline tund; arvestatakse nädalapäeva*

Nende puhul andis nupuvajutuseta ennustus vastavalt 2,93% ja 4,38% võrra parema tulemuse.

Alljärgnevad tabelid näitavad täpsushinnangute vahet protsentides. Arvud on saadud lahutades nupuvajutuseta versiooni täpsushinnang nupuvajutusega versioonist. Punasega on tähistatud erandid, kus nupuvajutuse mittearvestamine andis parema tulemuse. Rohelisega on tähistatud suurima vahega lahter, milleks on suure andmestiku pertseptroni täpsushinnang *ööpäeva osa; tööpäev või puhkepäev* ajavahemikuga.

Tabel 2. Nupuvajutust arvestava ja mittearvestavate versioonide täpsushinnangute vahe protsentides väikese andmestiku puhul

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	23.51	11.27	13.65	11.16	11.38
ööpäeva osa; tööpäev või puhkepäev	11.59	11.27	12.24	10.29	11.48
ööpäeva osa; arvestatakse nädalapäeva	25.57	9.97	11.81	10.94	12.78
täisarvuline tund; nädalapäeva ei arvestata	-2.93	10.4	12.35	10.73	14.3
täisarvuline tund; tööpäev või puhkepäev	10.51	10.4	12.78	10.51	14.52
täisarvuline tund; arvestatakse nädalapäeva	21.34	10.51	13.22	10.73	13

Tabel 3. Nupuvajutust arvestava ja mittearvestavate versioonide täpsushinnangute vahe protsentides suure andmestiku puhul

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	8.26	7.74	10.29	9.51	10.27
ööpäeva osa; tööpäev või puhkepäev	26.41	7.77	10.07	9.31	10.21
ööpäeva osa; arvestatakse nädalapäeva	4.6	7.9	10.28	9.65	11.23
täisarvuline tund; nädalapäeva ei arvestata	18.39	7.61	9.9	9.2	10.88
täisarvuline tund; tööpäev või puhkepäev	3.78	7.77	10.16	9.44	11.08
täisarvuline tund; arvestatakse nädalapäeva	-4.38	7.78	10.61	9.23	11.14

Eelnevast järeldub, et nupuvajutuse arvestamine tuleb peaaegu alati ennustuse täpsusele kasuks. Keskmiselt parandas nupuvajutuse arvestamine tulemust 9,84% protsenti. Mõju oli suurem väiksema andmestiku puhul (keskmiselt 10,14%) ning väiksem suure andmestiku puhul (keskmiselt 9,54%). Enim sõltusid nupuvajutuse valikust pertseptroni täpsushinnangud (keskmiselt 14,29%) ning kõige vähem logistilise regressiooni omad (keskmiselt 10,65%). Ajavahemiku liikidest lasi end nupuvajutusest kõige rohkem mõjutada ajavahemiku liik *ööpäeva osa; nädalapäeva ei arvestata* (keskmiselt 13,03%) ning kõige vähem *täisarvuline tund; nädalapäeva ei arvestata* (keskmiselt 10,46%).

3.2 Ajaparameetrite valiku mõju täpsushinnangule

Programmi käigus valminud töövihikute kõik neli tabelit koosnevad kuuest reast. Iga rida vastab ühele kuuest käesoleva töö käigus defineeritud ajavahemiku liigile.

Alljärgnevad tabelid näitavad täpsushinnanguid protsentides. Igale klassifikaatorile vastavast veerust on tähistatud rohelisega parim täpsushinnang. Võrdsete tulemuste korral on värvitud mõlemad täpsushinnangud.

Tabel 4. Töövihiku *small_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestamata

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	23.08	40.74	39.98	45.94	40.2
ööpäeva osa; tööpäev või puhkepäev	34.13	41.93	40.85	46.15	40.63
ööpäeva osa; arvestatakse nädalapäeva	19.93	41.5	40.3	46.48	41.39
täisarvuline tund; nädalapäeva ei arvestata	19.39	39.98	41.6	44.1	39.11
täisarvuline tund; tööpäev või puhkepäev	34.78	41.93	41.39	44.42	39.22
täisarvuline tund; arvestatakse	26	41.82	42.47	45.18	43.55

nädalapäeva					

Tabel 5. Töövihiku *small_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestades

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	46.59	52	53.63	57.1	51.57
ööpäeva osa; tööpäev või puhkepäev	45.72	53.2	53.09	56.45	52.11
ööpäeva osa; arvestatakse nädalapäeva	45.5	51.46	52.11	57.42	54.17
täisarvuline tund; nädalapäeva ei arvestata	16.47	50.38	53.95	54.82	53.41
täisarvuline tund; tööpäev või puhkepäev	45.29	52.33	54.17	54.93	53.74
täisarvuline tund; arvestatakse nädalapäeva	47.35	52.33	55.69	55.9	56.55

Tabel 6. Töövihiku *big_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestamata

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	41.87	46.22	52.03	47.44	53.58
ööpäeva osa; tööpäev või puhkepäev	21.62	46.23	52.44	47.24	53.58
ööpäeva osa; arvestatakse nädalapäeva	35.82	46.52	50.06	47.6	48.76
täisarvuline tund; nädalapäeva ei arvestata	25.22	46.08	50.95	47.08	50.07
täisarvuline tund; tööpäev või puhkepäev	38.52	46.17	50.85	46.89	49.77

täisarvuline tund; arvestatakse nädalapäeva	38.56	46.69	47.67	47.88	45.54

Tabel 7. Töövihiku *big_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestades

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	50.14	53.96	62.32	56.94	63.85
ööpäeva osa; tööpäev või puhkepäev	48.03	54	62.51	56.55	63.79
ööpäeva osa; arvestatakse nädalapäeva	40.42	54.42	60.34	57.25	59.99
täisarvuline tund; nädalapäeva ei arvestata	43.61	53.69	60.85	56.28	60.96
täisarvuline tund; tööpäev või puhkepäev	42.3	53.94	61.01	56.33	60.85
täisarvuline tund; arvestatakse nädalapäeva	34.18	54.47	58.29	57.12	56.69

Ajavahemiku liigi valiku mõju osutus ülejäänud parameetrite mõjuga võrreldes oluliselt ebamäärasemaks. Halvimaks osutus ajavahemikuliik *täisarvuline tund; nädalapäeva ei arvestata*, sest ta ei andnud ühelgi tingimusel parimat täpsushinnangut. Parim oli seevastu ajavahemikuliik *täisarvuline tund; arvestatakse nädalapäeva*, kuna ta saavutas kõrgeima täpsushinnangu kaheksal juhul kahekümnest võimalikust. Hästi on näha ka korrelatsiooni nuputa ja nupuga tabelites, võidavad samad lahtrid.

3.3 Klassifikaatori valiku mõju täpsushinnangule

Programmi käigus valminud töövihikute kõik neli tabelit koosnevad viiest veerust. Igaüks neist vastab ühele viiest antud töös kasutatud klassifikaatorile.

Alljärgnevad tabelid näitavad täpsushinnanguid protsentides. Igale ajavahemiku liigile vastavast reast on tähistatud rohelisega parim täpsushinnang. Kui mudelit väiksema andmestiku peal treenida on parim klassifikaator tugivektor-masin. Ainsaks erandiks on kombinatsioon, mis arvestab nupuvajutust ning kasutab ajavahemikuliiki *täisarvuline tund; arvestatakse nädalapäeva*. Viimase puhul annab otsustuspuu tugivektor-masinast 0,65% võrra parema tulemuse. Seevastu suurema andmestiku puhul on pooltel juhtudel parim *k* naabri põhine klassifikaator ning pooltel otsustuspuu.

Tabel 8. Töövihiku *small_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestamata

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	23.08	40.74	39.98	45.94	40.2
ööpäeva osa; tööpäev või puhkepäev	34.13	41.93	40.85	46.15	40.63
ööpäeva osa; arvestatakse nädalapäeva	19.93	41.5	40.3	46.48	41.39
täisarvuline tund; nädalapäeva ei arvestata	19.39	39.98	41.6	44.1	39.11
täisarvuline tund; tööpäev või puhkepäev	34.78	41.93	41.39	44.42	39.22
täisarvuline tund; arvestatakse nädalapäeva	26	41.82	42.47	45.18	43.55

Tabel 9. Töövihiku *small_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestades

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	46.59	52	53.63	57.1	51.57
ööpäeva osa; tööpäev või puhkepäev	45.72	53.2	53.09	56.45	52.11
ööpäeva osa; arvestatakse	45.5	51.46	52.11	57.42	54.17

nädalapäeva					
täisarvuline tund; nädalapäeva ei arvestata	16.47	50.38	53.95	54.82	53.41
täisarvuline tund; tööpäev või puhkepäev	45.29	52.33	54.17	54.93	53.74
täisarvuline tund; arvestatakse nädalapäeva	47.35	52.33	55.69	55.9	56.55

Tabel 10. Töövihiku *big_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestamata

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	41.87	46.22	52.03	47.44	53.58
ööpäeva osa; tööpäev või puhkepäev	21.62	46.23	52.44	47.24	53.58
ööpäeva osa; arvestatakse nädalapäeva	35.82	46.52	50.06	47.6	48.76
täisarvuline tund; nädalapäeva ei arvestata	25.22	46.08	50.95	47.08	50.07
täisarvuline tund; tööpäev või puhkepäev	38.52	46.17	50.85	46.89	49.77
täisarvuline tund; arvestatakse nädalapäeva	38.56	46.69	47.67	47.88	45.54

Tabel 11. Töövihiku *big_dataset_results.xls* täpsushinnangud, mis on treenitud nupuvajutust arvestades

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	50.14	53.96	62.32	56.94	63.85
ööpäeva osa; tööpäev või puhkepäev	48.03	54	62.51	56.55	63.79

ööpäeva osa; arvestatakse nädalapäeva	40.42	54.42	60.34	57.25	59.99
täisarvuline tund; nädalapäeva ei arvestata	43.61	53.69	60.85	56.28	60.96
täisarvuline tund; tööpäev või puhkepäev	42.3	53.94	61.01	56.33	60.85
täisarvuline tund; arvestatakse nädalapäeva	34.18	54.47	58.29	57.12	56.69

Klassifikaatori valik avaldas arvestatavat mõju täpsushinnangule. Väikese andmestiku puhul andis selgelt parima tulemuse tugivektor-masin. Seevastu suure andmestiku puhul osutusid parimateks hoopis *k* naabri põhine klassifikaator ja otsustuspuu. Viimaste puhul andis *k* naabri põhine klassifikaator parema tulemuse, kui ajavahemik oli kellaaja ja nädalapäevade alusel rohkem tükeldatud, ning otsustuspuu, kui ajalist tükeldamist oli vähem. Logistiline regressioon ja eriti pertseptron jäid ülejäänud klassifikaatoritele oluliselt alla.

3.4 Andmestiku suuruse mõju täpsushinnangule

Programmi tulemusena valminud esimene töövihik näitab täpsushinnanguid, mis on saadud väikese andmestiku peal treenides. Teine töövihik seevastu kasutab suurt andmestikku.

Enamasti andis suure andmestiku peal treenimine kõrgema täpsushinnangu kui väike andmestik. Suurimateks eranditeks olid kaks pertseptroni täpsushinnangut:

- Väikese andmestiku puhul ajavahemik *ööpäeva osa; tööpäev või puhkepäev*
- Suure andmestiku puhul ajavahemik *täisarvuline tund; arvestatakse nädalapäeva*

Nende puhul andis väiksema andmestiku peal treenimine vastavalt 12,51% ja 13,17% võrra parema tulemuse. Lisaks esines väikeseid erandeid tugivektor-masina puhul.

Alljärgnevad tabelid näitavad täpsushinnangute vahet protsentides. Arvud on saadud lahutades väikese andmestiku peal treenintud versiooni täpsushinnang suure andmestikuga versioonist. Punasega on tähistatud erandid, kus väiksema andmestiku peal treenimine andis parema tulemuse. Rohelisega on tähistatud suurima vahega lahter, milleks on nupuvajutust arvestav pertseptroni täpsushinnang *täisarvuline tund; nädalapäeva ei arvestata* ajavahemikuga.

Tabel 12. Väikest- ja suurt andmestikku kasutavate versioonide täpsushinnangute vahe protsentides; nupuvahutust ei arvestata

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	18.8	5.48	12.05	1.5	13.39
ööpäeva osa; tööpäev või puhkepäev	-12.51	4.3	11.59	1.09	12.95
ööpäeva osa; arvestatakse nädalapäeva	15.88	5.02	9.76	1.12	7.37
täisarvuline tund; nädalapäeva ei arvestata	5.82	6.11	9.34	2.99	10.96
täisarvuline tund; tööpäev või puhkepäev	3.75	4.24	9.46	2.46	10.55
täisarvuline tund; arvestatakse nädalapäeva	12.56	4.87	5.2	2.7	1.99

Tabel 13. Väikest- ja suurt andmestikku kasutavate versioonide täpsushinnangute vahe protsentides; arvestatakse ka nupuvajutust

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	3.55	1.96	8.7	-0.15	12.28
ööpäeva osa; tööpäev või puhkepäev	2.31	0.81	9.42	0.1	11.68
ööpäeva osa; arvestatakse	-5.08	2.96	8.23	-0.17	5.81

nädalapäeva					
täisarvuline tund; nädalapäeva ei arvestata	27.14	3.31	6.89	1.46	7.55
täisarvuline tund; tööpäev või puhkepäev	-2.98	1.61	6.84	1.4	7.11
täisarvuline tund; arvestatakse nädalapäeva	-13.17	2.14	2.6	1.21	0.13

Eelnevast järeldub, et suurema andmestiku peal treenimine tuleb reeglina ennustuse täpsusele kasuks. Keskmiselt parandas suurema andmestiku peal treenimine täpsushinnangut 5,27% protsenti. Suurem oli mõju nupuvajutuseta versiooni puhul (keskmiselt 6,69%) ning väiksem nupuvajutust arvestava versiooni puhul (keskmiselt 3,85%). Enim sõltusid treenimiseks kasutatava andmestiku suurusest otsustuspuu täpsushinnangud (keskmiselt 8,48%) ning kõige vähem tugivektor-masina omad (keskmiselt 1,31%). Ajavahemiku liikidest lasi end andmestiku suurusest kõige rohkem mõjutada ajavahemiku liik *täisarvuline tund; nädalapäeva ei arvestata* (keskmiselt 8,16%) ning kõige vähem *täisarvuline tund; arvestatakse nädalapäeva* (keskmiselt 2,02%).

Kokkuvõttes võib järeldada, et enim mõjutas täpsushinnangut klassifikaatori valik. Teiseks kõige rohkem mõjutas ennustuse täpsust see, kas arvestati, kumba nuppu reisija vajutas. Pisut vähem oli oluline andmetestiku maht, mille peal mudelit treeniti. Kõige ebamäärasemat ja väiksemat mõju avaldas täpsushinnangule valitud ajavahemiku liik.

4 Analüüs

4.1 Hinnang töö tulemustele

Käesoleva töö tulemused näitavad, et kõik neli töö käigus vaadeldud parameetrit mõjutavad suuremal või vähemal määral ennustuse täpsushinnangut.

Klassifikaatori valik avaldas täpsushinnangule suurimat mõju. Kõige nõrgemaks klassifikaatoriks osutus pertseptron, sest talle ei sobinud võimalike ennustatavate klasside rohkus. Logistilise regressiooni nõrgemapoolsete tulemuste taga võis olla see, et andmete sõltuvus pole lineaarne, kuid logistiline regressioon on lineaarne mudel. Tugivektor-masin oli kõige täpsem, kui andmeid oli vähe. Suure andmestiku puhul oli parim klassifikaator otsustuspuu. Tema väiksema edu põhjuseks väiksema andmestiku korral võis olla algoritmi ebastabiilsus: väike erinevus andmetes võis anda tulemuseks täiesti teistsuguse puu. Andmete rohkus ilmselt vähendas mudeli tundlikkust ebastabiilsuse vastu. *k* naabri põhine klassifikaator oli edukaim, kui ajavahemik oli tugevalt tükeldatud.

Väljaanne *Analytics India* võrdleb artiklis „7 Types of Classification Algorithms" seitset klassifikaatorit, sealhulgas logistilist regressiooni, tugivektor-masinat, *k* naabril põhinevat klassifikaatorit ja otsustuspuud. See töö esines käesolevast tööst klasside arvu poolest: neid oli vaid kaks. Erinevate demograafiliste näitajate põhjal ennustati, kas inimese palk on kuni 50000 dollarit või kõrgem [22]. Katse tulemusi kirjeldab tabel 14.

Tabel 14. Klassifikaatorite täpsushinnangute võrdlus [22]

Klassifikaator	Täpsushinnang protsentides
Logistiline regressioon	84.6
Otsustuspuu	84.23
Tugivektor-masin	84.09
k naabril põhinev klassifikaator	83.56

Teiseks kõige rohkem mõjutas ennustuse täpsust see, kas arvestati, kumba nuppu reisija vajutas. Selle parameetri ainsaks puuduseks on hetkel võimalus, et see on algandmetes vahel vigane, kuna seda informatsiooni pole otse kogutud, vaid arvutatud alg- ja lõppkorruse põhjal.

Arvestatavat mõju avaldas ka andmestiku suurus, mille peal mudelit treeniti. Sellest võib järeldada, et täpsushinnangute parandamiseks võiks veelgi rohkem andmeid koguda. Praeguse seisuga pole baas piisavalt suur, millest tulenevalt on täpsus väiksem.

Väljaanne *Machine Learning Mastery* toob oma artiklis „Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates" välja, et andmestikku suurendades paranevad täpsushinnangud mingi piirini, kuid see protsess peatub ühel hetkel ning hakkab isegi aeglaselt langema. Antud katses vaadeli ringide probleemi ja genereeriti juhuslikud andmed [24]. Katse tulemusi kirjeldab tabel 15.

Tabel 15. Täpsushinnangud ringide probleemi katse puhul [24]

Treeningandmestiku suurus	Täpsushinnang protsentides
100	72.04
1000	83.72
5000	84.06
10000	84.03

Kõige ebamäärasemat ja väiksemat mõju avaldas täpsushinnangule valitud ajavahemiku liik. Väiksema andmestiku peal andis pisut parema tulemuse ööpäeva jagamine täiarvulisteks tundideks ning suurema puhul ööpäeva jagamine neljaks osaks. Väikese andmestiku puhul aitas nädalapäeva arvestamine tulemust veidi parandada, kuid suure andmestiku puhul see ei avaldanud mõju.

4.2 Tulevikuväljavaated

Edaspidi tasub vaadeldud lifti toimimist parandada. Selleks tuleb valida parimad parameetrid ning luua süsteemile lisafunktsionaalsusi. Tulemustest järeldub, et väikese andmestiku peal treenides saadakse täpseim ennustus, kui klassifikaatorina kasutatakse tugivektor-masinat. Suurema andmestiku puhul on parim klassifikaator otsustuspuu. Erandiks on variandid, kus andmed on ajaliselt tükeldatud. Viimasel juhul annab teistest klassifikaatoritest parema tulemuse k naabri põhine klassifikaator.

Käesoleva töö tulemuste põhjal võib soovitada andmestikku võimalikult palju suurendada, et täpsus paraneks. Lisaks tuleks kasuks, kui lifti suunanupu vajutus kogutaks koos algandmetega, kuna tulemused näitavad, et see aitab täpsushinnangut parandada. Lisaks üles- ja allaliikumisnupule võib tulevikus nupuvajutuse võimalike väärtuste hulka lisada veel kaks võimalikku väärtust:

- Vajutatud on nii üles- kui ka allaliikumisnuppu. Vahel ootavad lifti korraga kaks reisijat, kellest üks soovib liikuda üles ning teine alla.
- Kumbagi nuppu pole vajutatud. Selline olukord võib tekkida siis, kui reisija ei jõudnudki lifti kutsuma hakata, vaid uks oli juhuslikult õigel hetkel lahti.

Lisaks eelnevale tasub anda kasutajale võimalus edastada süsteemile lisaparameetreid, näiteks oma tunniplaani. Seejuures tuleks tagada saadud andmete anonüümsus: need tuleks sisuda anonüümse identifikaatoriga. Lisaks võib iga semestri alguses anda süsteemile ette kogu hoone tundide tunniplaani, et süsteem saaks sellega arvestada.

5 Kokkuvõte

Käesoleva töö eesmärk oli ennustada liftireisi korrust, kuhu inimene kavatseb jõuda. Oli tarvis luua programm, mis leiab statistilise ennustuse täpsushinnangud, vaadeldes seejuures kõiki parameetreid eraldi. Protsessi lõpuks valmiski nõuetele vastav programm, mis väljastas täpsushinnangud Exceli töövihikutesse.

Tulemuste põhjal jõuti järeldusele, et väikese andmestiku puhul saadakse täpseim ennustus, kui klassifikaatorina kasutatakse tugivektor-masinat. Suurema andmestiku puhul annab reeglina parima tulemuse seevastu otsustuspuu. Kui aga andmed on tükeldatud, edestab teisi klassifikaatoreid k naabri põhine klassifikaator. Klassifikaator mõjutaski saadud täpsushinnanguid neljast vaadeldud parameetrist kõige rohkem. Võrreldi viit klassifikaatorit: pertseptron, logistiline regressioon, k naabri põhine klassifikaator, tugivektor-masin ja otsustuspuu. Veel parandas saadud täpsushinnangut arvestataval määral suuremal andmestikul treenimine ning reisija vajutatud lifti kutsumise nupu suund. Kõige vähem olenes sooritus sellest, kuidas ööpäev ja nädal ajavahemikeks jagati.

Täpsuse parandamise huvides tasub hakata arvestama ka uusi parameetreid. Näiteks saab hakata koguma andmeid selle kohta, kas olid alla vajutatud ülesliikumisnupp, allaliikumisnupp, mõlemad või mitte kumbki, ning hakata arvestama tunniplaani.

Kasutatud kirjandus

[1] Projekti Tark Lift tutvustav artikkel (avaldamata)

[2] Silja Kudel; I, Elevator. https://www.kone.com/en/news-and-insights/stories/i-elevator.aspx (18.05.2020)

[3] Prashant Gupta; Decision Trees in Machine Learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (18.05.2020)

[4] Scikit-learn documentation: Decision Trees. https://scikit-learn.org/stable/modules/tree.html#tree (18.05.2020)

[5] Rushikesh Pupale; Support Vector Machines(SVM) — An Overview. https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989 (18.05.2020)

[6] Scikit-learn documentation: Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html (18.05.2020)

[7] Scikit-learn documentation: SVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (18.05.2020)

[8] Universal Database Tool. https://dbeaver.io/ (18.05.2020)

[9] Scikit-learn documentation: Nearest Neighbors Classification. https://scikit-learn.org/stable/modules/neighbors.html#classification (18.05.2020)

[10] Scikit-learn documentation: KNeighborsClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (18.05.2020)

[11] Scikit-learn documentation: Nearest Neighbors Classification. https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py (18.05.2020)

[12] Sagar Sharma; What the Hell is Perceptron? https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53 (18.05.2020)

[13] Perceptron. https://scikit-learn.org/stable/modules/linear_model.html#perceptron (18.05.2020)

[14] Ants Torim; Tallinna Tehnikaülikooli aine *Andmekaeve suurandmetest (IDN1605)*, 4. ülesanne

[15] Scikit-learn documentation: Logistic regression. https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (18.05.2020)

[16] Liina-Liis Nõmme, Silver Schnur; Inimese elundkonnad. https://elundkonnad.weebly.com/naumlrvisuumlsteem.html (18.05.2020)

[17] Sachin Malhotra; Demystifying Gradient Descent and Backpropagation via Logistic Regression based Image. https://www.freecodecamp.org/news/demystifying-gradient-descent-and-backpropagation-via-logistic-regression-based-image-classification-9b5526c2ed46/ (18.05.2020)

[18] SCAN (Elevator) Disk Scheduling Algorithms. https://www.geeksforgeeks.org/scan-elevator-disk-scheduling-algorithms/ (18.05.2020)

[19] Hyundai Elevator. https://www.hyundaielevator.com/en/technology/smart (18.05.2020)

[20] Gigi Onag; AI and IoT are the keys to smarter lifts and escalators. https://futureiot.tech/ai-and-iot-are-the-keys-to-smarter-lifts-and-escalators/ (18.05.2020)

[21] Data Mining and Its Importance. https://www.loginworks.com/blogs/217-data-mining-and-its-importance/ (18.05.2020)

[22] Rohit Garg; 7 Types Of Classification Algorithms. https://analyticsindiamag.com/7-types-classification-algorithms/ (18.05.2020)

[23] Taivo Pungas; Masinõpe: mittetehniline ülevaade. https://pungas.ee/masinope-mittetehniline-ulevaade/ (18.05.2020)

[24] Jason Brownlee; Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates. https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/ (18.05.2020)

[25] Meeli Roosalu; Inimese anatoomia, lk 182, Kirjastus Koolibri, 2010, ISBN 978-9985-0-2606-9

Lisa 1 – Programmi kood

```
from sklearn.linear_model import Perceptron
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier

import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import psycopg2

import xlwt
from xlwt import Workbook
```

Joonis 9. Programmi kood: importimine

```
def list_to_sheet(sheet, row, insert_list):
    """Funktsioon võtab sisendiks objekti, mis esindab Exceli lehte,
    kuhu soovitatakse andmeid väljastada,
    reanumbri, kuhu andmed sisestatakse ning listi sisestatavatest andmetest.
    Funktsiooni käigus sisestatakse andmed Excel'i tabelisse."""
    for column in range(6):
       sheet.write(row, column, insert_list[column])

def predict_with_button(X, y, option, row, sheet1, sheet2):
    """Funktsioon võtab sisendiks andmemaatriksi, klassivektori, ajavahemiku
liigi,
    Exceli reanumbri ning kaks Excel'i lehte, kuhu soovitakse lõpptulemust vä
ljastada
    Esimesele lehele väljastab programm andmed nii, et ei arvesta reisija vaj
utatatud nuppu.
    Teisel lehel arvestatakse ka nupuvajutust."""
    X_with_button = X[:,:-1]
    result_list = predictions(X_with_button, y, option)
    list_to_sheet(sheet1, row, result_list)
    result_list = predictions(X, y, option)
    list_to_sheet(sheet2, row, result_list)
```

Joonis 10. Programmi kood: funktsioonid 1

```
def predictions(X, y, option):
  """Funktsioon võtab sisendiks andmemaatriksi, klassivektori ja
    ajavahemiku liigi nime.
    Ta arvutab tõenäosused kasutades järgmisi klassifikaatoreid:
    1) Perceptron
    2) LogisticRegression
    3) KNeighborsClassifier
    4) SVC
    5) DecisionTreeClassifier
    Funktsioon väljastab listi, mille esimene element on ajavahemiku liigi ni
mi
    ning järgmised viis eelmainitud klassifikaatorite põhjal arvutatud tõenäo
sused.
    """
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                        random_state=0)
    sc = StandardScaler()
    sc.fit(X_train)
    classifier_list = ['Perceptron', 'LogisticRegression',
                       'KNeighborsClassifier', 'SVC', 'DecisionTreeClassifier
']
    result_list = [option]
    X_train_std = sc.transform(X_train)
    X_test_std = sc.transform(X_test)
    for classifier in classifier_ list:
        if classifier == 'Perceptron':
            predictive_model = Perceptron(random_state=0)
        elif classifier == 'LogisticRegression':
            predictive_model = LogisticRegression()
        elif classifier == 'KNeighborsClassifier':
            predictive_model = KNeighborsClassifier()
        elif classifier == 'SVC':
            predictive_model = svm.SVC()
        else:
            predictive_model = DecisionTreeClassifier()
        result = get_accuracy(X_train_std, y_train,
                      X_test_std, y_test, predictive_model)
        result_list.append(result)
    return result_list
```

Joonis 11. Programmi kood: funktsioonid 2

```
def get_accuracy(X_train_std, y_train, X_test_std, y_test, predictive_model):
  """Funktsioon võtab sisendiks standardiseeritud andmemaatriksi (treeningand
med),
    klassivektori (treeningandmed),
    standardiseeritud andmemaatriksi (testandmed),
    klassivektori (testandmed) ja kasutatava klassifikaatori.
    Funktsioon viib läbi ennustusprotsessi ning tagastab, mitu protsenti
    läks õigesti.
    """
    predictive_model.fit(X_train_std, y_train)
    y_pred = predictive_model.predict(X_test_std)
    return accuracy_score(y_test, y_pred) * 100

def list_of_lists(test_records):
    """Funktsioon võtab sisendiks päringust saadud kirjete listide listi.
    Ta loob listide listi, kus iga alamlist sisaldab
    1) reisija identifikaatorit
    2) liftireisi algkorrust
    3) ööpäeva osa: öö, hommik, lõuna või õhtu
    4) kellaja täisarvulist tundi
    5) vastust küsimusele, kas on töö- või puhkepäev
    6) nädalapäeva
    List muudetakse omakorda massiiviks."""
    X_list = []
    for row in test_records:
        person = row[0]
        start = row[1]
        end = row[2]
        pressed_up_button = True
        #võtame arvesse, kas inimene vajutas üles või alla nuppu
        if start > end:
            pressed_up_button = False
        if person != None:
            #kellaaeg kategoriseeritud: 7-12, 13-15, 16-19 ja 20-6
            hour = int(row[3])
            time = 0 #öö
            if hour >= 7 and hour <= 12:
                time = 1 #hommik
            elif hour >= 13 and hour <= 15:
                time = 2 #lõuna
            elif hour >= 16 and hour <= 19:
                time = 3 #õhtu
            weekday = int(row[4])
            #kas on tööpäev või nädalavahtus
            is_weekend = 0
            if weekday == 0 or weekday == 6:
                is_weekend = 1
            list_ = [person, start, time, hour, weekday, is_weekend, pressed_
up_button]
            X_list.append(list_)
    X = np.array(X_list)
    return X
```

Joonis 12. Programmi kood: funktsioonid 3

```
def destination_list(test_records):
  """Funktsioon võtab sisendiks päringust saadud kirjete listide listi.
    Ta valib igast kirjest sihtkorruse, välja arvatud juhul,
    kui reisija identifikaator puudub."""
    y_list = []
    for row in test_records:
        if row[0] != None:
            y_list.append(row[2])
    y = np.array(y_list)
    return y

def get_data(connection):
    """Ühenduse connection abil võetatakse ühendust soovitud baasiga.
    Tehakse päring, milles küsitakse andmebaasist iga liftireisi kohta järgmi
sed andmed:
    1) reisija identifikaator
    2) algkorrus
    3) lõppkorrus
    4) kellaja täisarvuline tund
    5) nädalapäev
    Funktsioon väljastab listi kujul andmemaatriksi ja klassivektori."""
    try:
        cursor = connection.cursor()
        postgreSQL_select_query = """SELECT pot.profile_id,
        t.start_floor, t.destination_floor,
        extract(hour from t.time), extract(dow from t.time)
         FROM trips as t
         left join profiles_on_trips AS pot
        ON t.trip_id=pot.trip_id"""

        cursor.execute(postgreSQL_select_query)
        test_records = cursor.fetchall()

        #andmed
        X=list_of_lists(test_records)
        y=destination_list(test_records)
        return X, y

    except (Exception, psycopg2.Error) as error:
        print("Error while fetching data from PostgreSQL", error)

    finally:
        # andmebaasi ühenduse sulgemine
        if (connection):
            cursor.close()
            connection.close()
```

Joonis 13. Programmi kood: funktsioonid 4

```
def main_process(connection, wb_name):
  """Funktsioon võtab sisendiks soovitud andmebaasile vastava ühenduse ja
    failinime, mis soovitakse tulevasele Exceli failile panna.
    Luuakse kaks Exceli töövihiku lehte.
    Kõigi kuue ajavahemiku liigi põhjal viiakse läbi ennustamised.
    Massiivist X valitakse ajavahemikele vastavad veerud.
    Kõik andmed salvestatakse soovitud nimega Exceli faili.
    """
    X, y = get_data(connection)

    #töövihiku loomine
    wb = Workbook()

    sheet1 = wb.add_sheet('Ilma nuputa')
    sheet2 = wb.add_sheet('Koos nupuga')

    insert_list = ['Ajavahemiku liik', 'Perceptron', 'LogisticRegression',
                           'KNeighborsClassifier', 'SVC', 'DecisionTreeClassi
fier']
    list_to_sheet(sheet1, 0, insert_list)
    list_to_sheet(sheet2, 0, insert_list)

    dict = {"ööpäeva osa; nädalapäeva ei arvestata": [True, True, True, False
, False, False, True],
       "ööpäeva osa; tööpäev või puhkepäev": [True, True, True, False, False,
True, True],
       "ööpäeva osa; arvestatakse nädalapäeva": [True, True, True, False, Tru
e, False, True],
       "täisarvuline tund; nädalapäeva ei arvestata": [True, True, False, Tru
e, False, False, True],
       "täisarvuline tund; tööpäev või puhkepäev": [True, True, False, True,
False, True, True],
       "täisarvuline tund; arvestatakse nädalapäeva": [True, True, False, Tru
e, True, False, True]}
    row = 1
    for option in dict:
        predict_with_button(X[:,np.array(dict[option])],
                        y, option, row, sheet1, sheet2)
        row += 1

    wb.save(wb_name + '.xls')
```

Joonis 14. Programmi kood: põhifunktsioon *main_process*

```
connection = psycopg2.connect(user="postgres",
                                  password="postgres",
                                  host="127.0.0.1",
                                  port="5432",
                                  database="diplom")

main_process(connection, 'small_dataset_results')

connection2 = psycopg2.connect(user="postgres",
                                  password="postgres",
                                  host="127.0.0.1",
                                  port="5433",
                                  database="diplom_uus")

main_process(connection2, 'big_dataset_results')
```

Joonis 15. Programmi kood: funktsiooni *main_process* välja kutsumine

Koodi link Githubis: https://gist.github.com/jaanikaraik/c32052f8fc969db85f78117a8ebb59c7

Lisa 2 – Väljavõte andmebaasidest

	trip_id	start_floor	destination_floor	time	elevator_id	elevator_load
1	225	1	3	2019-08-30 13:14:31	car:1000000746:1:1:0	2
2	226	1	5	2019-08-30 13:57:45	car:1000000746:1:1:0	13
3	227	1	3	2019-08-30 15:10:59	car:1000000746:1:1:0	22
4	228	6	6	2019-08-30 15:21:37	car:1000000746:1:1:0	13
5	229	1	6	2019-08-30 15:23:35	car:1000000746:1:1:0	1
6	230	1	4	2019-08-30 15:30:21	car:1000000746:1:1:0	4
7	231	1	4	2019-08-30 15:30:22	car:1000000746:1:1:0	4
8	232	3	1	2019-08-30 15:32:41	car:1000000746:1:1:0	11
9	233	3	1	2019-08-30 15:32:42	car:1000000746:1:1:0	11
10	234	6	0	2019-08-30 16:41:08	car:1000000746:1:1:0	2
11	235	4	0	2019-08-30 16:43:08	car:1000000746:1:1:0	2
12	236	1	3	2019-08-30 17:14:00	car:1000000746:1:1:0	4
13	237	5	1	2019-08-30 17:18:46	car:1000000746:1:1:0	4
14	238	5	1	2019-08-30 18:58:45	car:1000000746:1:1:0	1
15	239	6	1	2019-09-03 15:14:46	car:1000000746:1:1:0	1

Joonis 16. Tabeli *trips* 15 esimest kirjet

	ident_id	profile_id	entry_floor	exit_floor	time	trip_id
1	1,237	77	☑	☑	2019-10-15 16:56:44	29,511
2	1,238	120	☐	☑	2019-10-15 16:56:47	29,512
3	1,239	77	☑	☑	2019-10-15 17:01:00	29,511
4	1,240	77	☑	☐	2019-10-15 17:01:00	29,511
5	1,241	120	☐	☑	2019-10-15 17:01:03	29,512
6	1,249	180	☑	☑	2019-10-15 18:13:17	30,086
7	1,250	47	☑	☑	2019-10-15 18:21:18	30,087
8	1,251	47	☑	☐	2019-10-15 18:21:18	30,087
9	1,252	166	☑	☑	2019-10-15 18:21:55	30,089
10	1,253	47	☐	☑	2019-10-15 18:51:45	30,093
11	1,254	47	☑	☐	2019-10-15 18:51:45	30,093
12	1,255	47	☐	☑	2019-10-15 18:52:11	30,094
13	1,256	47	☐	☑	2019-10-15 18:52:40	30,095
14	1,257	47	☐	☑	2019-10-15 18:53:03	30,096
15	1,258	65	☑	☑	2019-10-15 18:53:03	30,096

Joonis 17. Tabeli profiles_on_*trips* 15 esimest kirjet

Lisa 3 – Programmi väljund

Tabel 16. Töövihiku *small_dataset_results.xls* esimene leht

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	23.08	40.74	39.98	45.94	40.2
ööpäeva osa; tööpäev või puhkepäev	34.13	41.93	40.85	46.15	40.63
ööpäeva osa; arvestatakse nädalapäeva	19.93	41.5	40.3	46.48	41.39
täisarvuline tund; nädalapäeva ei arvestata	19.39	39.98	41.6	44.1	39.11
täisarvuline tund; tööpäev või puhkepäev	34.78	41.93	41.39	44.42	39.22
täisarvuline tund; arvestatakse nädalapäeva	26	41.82	42.47	45.18	43.55

Tabel 17. Töövihiku *small_dataset_results.xls* teine leht

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	46.59	52	53.63	57.1	51.57
ööpäeva osa; tööpäev või puhkepäev	45.72	53.2	53.09	56.45	52.11
ööpäeva osa; arvestatakse nädalapäeva	45.5	51.46	52.11	57.42	54.17
täisarvuline tund; nädalapäeva ei arvestata	16.47	50.38	53.95	54.82	53.41
täisarvuline	45.29	52.33	54.17	54.93	53.74

tund; tööpäev või puhkepäev					
täisarvuline tund; arvestatakse nädalapäeva	47.35	52.33	55.69	55.9	56.55

Tabel 18. Töövihiku *big_dataset_results.xls* esimene leht

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	41.87	46.22	52.03	47.44	53.58
ööpäeva osa; tööpäev või puhkepäev	21.62	46.23	52.44	47.24	53.58
ööpäeva osa; arvestatakse nädalapäeva	35.82	46.52	50.06	47.6	48.76
täisarvuline tund; nädalapäeva ei arvestata	25.22	46.08	50.95	47.08	50.07
täisarvuline tund; tööpäev või puhkepäev	38.52	46.17	50.85	46.89	49.77
täisarvuline tund; arvestatakse nädalapäeva	38.56	46.69	47.67	47.88	45.54

Tabel 19. Töövihiku *big_dataset_results.xls* teine leht

Ajavahemiku liik	*Perceptron*	*LogisticRegression*	*KNeighborsClassifier*	*SVC*	*DecisionTreeClassifier*
ööpäeva osa; nädalapäeva ei arvestata	50.14	53.96	62.32	56.94	63.85
ööpäeva osa; tööpäev või puhkepäev	48.03	54	62.51	56.55	63.79
ööpäeva osa; arvestatakse nädalapäeva	40.42	54.42	60.34	57.25	59.99
täisarvuline tund; nädalapäeva ei arvestata	43.61	53.69	60.85	56.28	60.96
täisarvuline tund; tööpäev või puhkepäev	42.3	53.94	61.01	56.33	60.85

täisarvuline tund; arvestatakse nädalapäeva	34.18	54.47	58.29	57.12	56.69