TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Informaatika aluste õppetool

Autojuhi tuvastamine sõitude koordinaatide analüüsi ja masinõppe abil

Magistritöö

Üliõpilane: Sergei Beregov
Üliõpilaskood: 121795IAPM
Juhendaja: Tarmo Veskioja

Tallinn
2015

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.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesolevas magistritöös keskenduti autojuhi sõitude andmete analüüsimisele, et kindlas teha, kas roolis olnud auto juht oli auto omanik. Iga sõit sisaldas sõidu jooksul läbitud koordinaatide kogumikku. Selline andmeanalüüs on kasutatav näiteks kindlustusfirmade poolt, et lahendada kindlustusjuhtumeid, kus näiteks auto on ärandatud või üritatakse läbi viia kindlustuspettusi.

Töö käigus vaadeldud andmed pärinesid Kaggle poolt korraldatud andmeanalüütikutele suunatud konkursilt ning kõik andmed pärinesid reaalsetest GPS seadmetest. Algandmete kohta oli teada, et enamik ühe auto sõitudest kuuluvad autojuhile (selle auto omanikule). Töö käigus normaliseeriti sõidu andmed ja eemaldati sõidu andmetest korduvad koordinaadid. Samuti kasutati andmete normaliseerimiseks Ramer-Douglas-Peuckeri algoritmi kõvera punktide vähendamiseks. [1]

Töös kirjeldatud meetodite abil võrreldi autojuhi sõitude punktide koordinaate ja meetod andis otsuse, kas sõit kuulub auto omanikule või mitte. Samuti kasutati töös *Random Forest* masinõppe meetodit, et hinnata tõenäosust, kas sõit kuulub juhile või mitte. [2]

Töö tulemusena realiseeriti andmete normaliseerimise ja töötlemise algoritmid programmeerimiskeeltes Go ja Python. Töö tulemusi hinnati kasutades Kaggle veebipõhist tulemuste hindamise platvormi. Töö tulemusena saavutati Kaggle veebilehel skooriks 0.89917.

Töö sisaldab endas kasutatud meetodite ja loodud algoritmide detailseid kirjeldusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 9 peatükki, 8 joonist, 6 tabelit, 1 lisa.

Abstract

Current Master's thesis analyses the data of car driver's trips in order to identify whether the person behind the wheel was the car's owner or not. Every drive included collections of coordinates measured throughout the trip. Such data analysis is applicable for the insurance companies in order to solve insurance cases, e.g. cases in which the car was stolen or the person concerned attempted to carry out an insurance fraud.

The collected data originates from data analysis competition by Kaggle and all the data originated from real GPS devices. Regarding the trip data, it was known that the majority of one car's drives belonged to the driver who was also the owner of the car. Throughout the thesis the data was normalized and the repetitive coordinates were removed. Also, Ramer-Douglas-Peucker algorithm was used in order to normalize the data.

With the help of the methods described in the thesis, the driver's trips coordinates were compared and the method resulted in a probability, that the person behind the wheel was the owner of the car. In addition to coordinate comparison mothod, Random Forest machine learning method was used in order to assess the probability of whether the trip belonged to the driver or not.

As an additional result of the analysis within this thesis, the normalizing and processing algorithms of the data were realized in programming languages Go and Python. The results were evaluated using Kaggle's web-based assessment platform. The result on the homepage of the aforementioned platform was 0.89917.

The thesis is in Estonian and contains 37 pages of text, 9 chapters, 8 figures, 6 tables, 1 appendix.

Lühendite ja mõistete sõnastik

AUC	***Area Under the Curve*** Pindala ROC-kõvera all. Suurus, mille abil iseloomustatakse meetodi edu- või ebaedu
EDA	***Exploratory Data Analysis*** Uurimuslik andmeanalüüs, kus üritatakse leida seoseid andmete vahel, kus ei ole ette kujutust nende andmete tegelike seoste kohta [3]
HPC	***High Performance Computing*** Paralleeltöötluse kasutamine arvutuste teostamiseks. Tihti kasutatakse selleks riistvaraliselt võimekaid masinaid, mis on omavahel ühendatud ja suhtlevad omavahel
RF	***Random Forest*** Masinõppe meetod, mille puhul konstrueeritakse pseudojuhuslikke otsustuspuid. Iga puu abil ennustatakse objekti kuuluvust klassi [4]
ROC	***Receiver operating characteristic*** Meetod mudeli või testi edukuse hindamiseks, mis tuvastab kui edukalt suudab meetod individuaalsed tulemused jaotada kahte klassi [5]
Otsustuspuu	***Decision tree*** Andmekaeve tööriist objekti klassi ennustamiseks kasutades puu sarnast mudelit. [6]
programmeerimis-vahendid	***Programming tools*** Programmid, utiilitid teegid, mida saab kasutada programmide kirjutamisel [7]
Serialiseerimine	***Serialization*** Objekti konverteerimine bittide jadaks, mida on võimalik salvestada faili ja pärast lugeda failist objekti

Suurte andmehulkade analüüs	***Big Data Analytics*** Protsess, mille jooksul kogutakse, korrastatakse ja analüüsitakse suuri andmehulkasid, et leida sarnaseid käitumismustreid [8]

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Tänapäeva digitaliseeruvas maailmas on suur roll andmetel. Kuna interneti kasutajate arv maailmas kasvab, siis kasvab ka andmete hulk, mis kesutajatest maha jääb. Kasutaja jätab iga kasutuskorra ajal endast maha jälje. Need andmed sisaldavad endas infot kasutaja kohta, milliseid teenuseid ta tarbib ja millal ta seda teeb. Nende andmete abil on võimalik teha kasutajale suunatud pakkumisi, soovitada kasutajale osta tooteid, mis on seotud tema eelnevalt ostetud toodetega ja näiteks soovitada talle vaatamisväärsusi linnas, kuhu kasutaja plaanib sõita.

Kuna interneti kasutajate arv kasvab, siis kasvab ka seadmete arv, mis on internetiga ühendatud. Neid seadmeid on mitmesuguseid nagu näiteks mobiiltelefonid ja tahvelarvutid aga ka näiteks GPS seadmed, mis saavad oma tarkvara ja navigeerimiseks kasutatavaid kaarte läbi interneti uuendada. Samuti suudavad seadmed koostööd teha erinevate teiste seadmete ja teenustega.

Suurte andmehulkade analüüsi (*Big Data Analytics*) abil on võimalik kindlaks teha väga paljusid seadme kasutajaga seotud fakte. Antud magistritöö raames on analüüsitud andmehulka, mis koosneb autojuhtide sõitudest. Magistritöö raames on loodud ka algoritmid, mis erinevaid meetodeid kasutades tuvastavad tõenäosuse, et auto roolis oli antud auto omanik.

1.1 Taust ja probleem

Töö analüüsib autojuhi sõidukäitumist ja leiab seaduspärasused autojuhi sõitude valimite hulgast.

Töö tulemusena saab kindlaks teha, kas autot juhtinud juht on auto tegelik omanik ja talle laieneb temale tehtud kindlustuspakkumine või mitte. Samuti on võimalikuks kasutusjuhuks autojuhi riskikoefitsendi määramine ja talle kindlustuspakkumise tegemine. Lisaks saab töö tulemusena valminud algoritmi kasutada koos teiste andmetega, et leida seaduspärasusi sarnaste autojuhtide sõidukäitumises kasutades ka muid kriteeriume ja ennustada või ennetada autojuhti eeldatavaid riske. Näiteks kui autojuht sõidab tihti ühte teed iga päev ja sellel teel juhtub palju liiklusõnnetusi, siis võib pakkuda autojuhile alternatiivse tee.

Magistritöö teostati Tallinna Tehnikaülikooli Informaatikainstituudis. Magistritöö andmed pärinevad Kaggle Incl äbiviidud konkursilt "Driver Telematics Analysis", mille korraldajaks oli kindlustusettevõte AXA, mis antud konkursi tarbeks andmetega varustas. [9]

Töö tulemuste hinadmiseks kasutatakse Kaggle veebilehel olevat hindamissüsteemi, mis põhineb ROC hindamismudelil.

1.2 Ülesande püstitus

Töö eesmärgiks on luua algoritmid, mille abil suudetakse tuvastada autojuhi sõitude hulgast need, mis on tehtud auto omaniku poolt eeldusel, et enamik sõite autojuhi sõitude valimist on tehtud auto omaniku poolt. Algoritm realiseeritakse programmeerimiskeeles Go ja käivitatakse testimiseks TTÜ HPC arvutusklastri masinates.

Töö teostatakse kasutades uurimusliku analüüsi metoodikat (*Exploratory Data Analysis*), kus on teada eesmärk, mida tahetakse saavutada, kuid kasutatavate meetodite edu või ebaedu on eesmärgi saavutamiseks teadmata. Samuti on pärast töö valmimist võimalik antud tööd edasi arendada kasutades teisi meetodeid. [3]

Töö käigus loodavad algoritmid jaotatakse andmete normaliseerimise algoritmideks ja normaliseeritud andmeid analüüsivateks algoritmideks. Normaliseerimisalgoritmide ülesandeks on algandmete normaliseerimine, et neid saaks kasutada andmete

analüüsimiseks. Andmete normaliseerimisel tehakse erinevaid arvutusi nagu näiteks sõidu punktide vahemaade arvutus ja eemaldatakse andmed, mis on potentsiaalselt vigased.

Andmeid analüüsivad algoritmid tegelevad normaliseeritud andmete analüüsiga ning pärast analüüsi tagastavad tulemuse, milleks on tõenäosus, et töödeldud andmete kohta käiv sõit kuulub antud autojuhile.

Iga töös kasutatud meetodi lõpptulemuseks on CSV fail, mis vastab Kaggle nõudmistele ja mille saab üles laadida tulemuse kontrollimise veebilehele.

Autori eesmärgiks on saavutada skooriks Kaggle veebilehe järgi vähemalt 0.75.

1.3 Metoodika

Töö käigus normaliseeriti algandmed pöörates autojuhi sõidud nii, et nad oleks x-telje suunalised. Ning leiti autojuhi sarnased sõidud võrreldes sõidu punktide koordinaate.

Töös normaliseeriti andmeid ka kasutades Ramer–Douglas–Peuckeri algoritmi kõvera punktide vähendamiseks ning prooviti seejärel võrrelda uuesti punktide koordinaate pärast liiga kaugete punktide eemaldamist. [1]

Töös kasutatakse ka *Random Forest* masinõppe meetodit, et ennustada tõenäosus, et roolis oli auto omanik. Mudelt treeniti mitmeid kordi ning võeti kõikide treeningtulemuste keskmine ning saadi selle meetodi poolt väljastatud lõpptulemus. [2]

Töö käigus saadud tulemust kontrolliti ROC-kõvera analüüsi abil kasutades Kaggle veebilehte. [5]

Töö käigus loodud algoritme käivitati Tallinna Tehnikaülikooli arvutusklastis.

1.4 Ülevaade tööst

Töö esimeses peatükis on selgitatud ülesande püstitust ja töös kasutatavaid metoodeid püstitatud probleemi lahendamiseks.

Töö teises peatükis kirjeldatakse töö käigus kasutatud algandmeid. Kirjeldatakse ära algandmete hulk ja struktuur.

Käesoleva töö kolmas peatükk keskendub töö käigus kasutatava tehnoloogilise baasi kirjeldamisele. Kirjeldatakse ära töö käigus loodud algoritmide realiseerimiseks kasutatud programmeerimiskeeled ja tähtsaimad teegid.

Neljandas peatükis on selgitatud algoritmi tulemuste hindamise printsiipe. Selgitatud on ROC-kõvera meetodit ning tehtud ka lihtne ROC-kõvera punktide arvutamise näidis näiteandmetega.

Viiendas peatükis seletatakse töö käigus kasutatud andmete normaliseerimist. Samuti on käsitletud normaliseeritud andmete analüüsi ja sarnaste sõitude leidmist võrreldes sõitude jooksul läbitud punktide koordinaate.

Kuuendas peatükis on selgitatud Ramer–Douglas–Peuckeri algoritmi olemust ja sõitude ühitamise tulemusi pärast andmete normaliseerimist kasutades Ramer–Douglas–Peuckeri algoritmi kõvera punktide vähendamiseks.

Seitsmendas peatükis on selgitatud masinõppe meetodi *Random Forest* põhimõtet ja selle meetodi abil loodud treeningmudeli treeningtulemusi ja nendest tulemustest lõpptulemuse arvutamist.

Kaheksanda peatüki sisuks on andmete analüüsi tulemuste tõlgendamine. Samuti on välja pakutud antud magistritöö mõned võimalikud edasi arendamise suunad.

Üheksandas peatükis esitatakse töö kokkuvõte.

2. Algandmete kirjeldus

Iga juhi sõidud on jaotatud juhi unikaalse numbrilise identifikaatori järgi kaustadesse. Igas kaustas on CSV failide hulk, kus iga fail sisaldab ühe sõidu x ja y koordinaate. Iga sõit sisaldab x ja y koordinaati. Kõik koordinaadid on pärit reaalsetest GPS seadmetest reaalsetelt inimestelt, kuid on anonümiseeritud kujul. Iga sõidu koordinaadid on tasapinnal juhuslikult pööratud. Iga sõit sisaldab punkti (0; 0), mis tähistab sõidu alguspunkti.

Kuna GPS seadmete signaal võib vahepeal kõikuda, siis võivad algandmed sisaldada ka mitu ühesuguste koordinaatidega punkti. Samuti on võimalik, et koordinaatide andmefailis ei alga punktid koordinaatidega (0; 0), vaid sõidu lõpu koordinaatidega. Sellisel juhul tuleb alustada sõidu analüüsimist ikkagi punktist (0; 0).

Algandmete kohta on teada ka see, et igale juhile on meelega lisatud sõite, mis antud juhile ei kuulu, kuid nende arv jääb kindlasti alla poolte autojuhi sõitudest.

2.1 Algandmete analüüs: üldstatistika

Enne andmete uurimist ja töö alustamist oli mõistlik teha andmehulga kohta üldstatistika. Andmete kohta tehtud statistika on esitatud Tabelis 1.

Tabel 1: Statistika andmete kohta

Autojuhte kokku	2736
Sõite kokku	547200
Keskmiselt sõite ühe autojuhi kohta	200
Algandmete maht	~5.5 GB

3. Tehnoloogiline baas algoritmide realiseerimiseks

3.1 Go

Go puhul on tegemist staatiliselt tüübitud programmeerimiskeelega, mis on avatud lähtekoodiga ja mille arendamist alustati Googles ja mille esimene stabiilne versioon ilmus aastal 2011. Go keel võimaldab ülesandeid jooksutada paralleelselt kasutades ära kogu masina võimsust. Töö tegemise ajal on Go keele stabiilseks versiooniks 1.4.2 ja seda on kasutatud töö käigus loodava algoritmi realiseerimiseks. Go keel sai valitud põhiliselt tema uudsuse tõttu ja selle tõttu, et see võimaldab ülesandeid paralleelselt täita. Antud töö puhul on paralleelseteks ülesanneteks andmete lugemine ja töötlus. Kuna antud töö puhul ei ole andmete hulk väga suur (5.5 GB), siis leidis töö autor, et Go programmeerimiskeel on ülesandes püstitatud eesmärkide saavutamiseks piisav vahend. [10]

3.2 HPC arvutusklaster

Töö käigus realiseeritud algoritmi testiti TTÜ arvutusklastris. HPC (High Performance Computing) arvutusklaster on mõeldud teadusarvutuste tegemiseks, mis nõuavad suurt arvutusvõimsust. Kuna töös kasutatud andmehulk on piisavalt suur ja andmete töötlemine kodukasutuses oleva personaalarvuti abil ei ole eriti mõistlik, otsustas autor kasutada TTÜ arvutusklastrit. Arvutusklastrist oli töö autorile töö tegemise ajaks eraldatud üks arvutusmasin, milles oli 24 Inteli 64bit protsessorit ja 48GB mälu. [11]

3.3 Python

Python on dünaamilise tüübikeelega programmeerimiskeel, mis on vabavaraline ja laiendatav. Python on laialdaselt kasutatav programmeerimiskeel paljudes valdkondades nagu näiteks veebiarendus ja andmetöötlus. Pythoni programme on võimalik laiendada Pythoni tarkvara pakettide abil. Pythoni programmeerimiskeel sai valitud antud magistritöö masinõppe analüüsi tarbeks, sest see omab sobivaid programmeerimisvahendeid, andmeanalüüsi teostamiseks. [12]

3.4 Pythoni tarkvarapakett: NumPy

NumPy on Pythoni tarkvarapakett, mis on mõeldud teadusarvutuste sooritamiseks kasutades Pythoni programmeerimiskeelt. NumPy lihtsustab n-mõõtmeliste massiivide töötlust ja suurte andmehulkade hoidmist massiivides. [13]

3.5 Pythoni tarkvarapakett: scikit-learn

Tarkvarapakett, mis sisaldab endas Pythoni programmeerimiskeele jaoks realiseeritud masinõppe ja andmekaeve algoritme. Samuti sisaldab antud pakett võimalust andmeid visualiseerida. [14]

3.6 Pickle

Pythoni programmeerimiskeele moodul, mis on mõeldud Pythoni objektide serialiseerimiseks. Serialiseerimise käigus muudetakse objekt bittide jadaks, mida on võimalik salvestada faili. Seejärel on võimalik failist lugeda otse Pythoni objekti. Antud lähenemine kiirendab failidest lugemise operatsioone ning on levinud praktika suurte andmehulkade töötlemisel. [15]

4. Algoritmi tulemuse hindamine

Algoritmi tulemuse hindamiseks kasutatakse Kaggle veebilehte, kuhu saab üles laadida CSV faili, mis peab sisaldama kõikide algandmetes antud juhtide ja nende sõitude andmeid. Kaggle platvorm ei võimalda kontrollida ainult osalist tulemust. [16]

Kaggle platvorm kasutab tulemuste hindamiseks ROC-kõvera meetodit. ROC kõver on arvutatud kõikide sõitude kohta. [17]

4.1 ROC-analüüs

ROC-kõver (*Receiver Operating Characteristic Curve*) on efektiivne meetod hindamaks diagnostiliste testide tulemuslikkust, mis on graafikul defineeritud vastavalt: y-teljel tundlikkus ning x-teljel valepositiivse tulemuse tõenäosus ehk (1-spetsiifilisus). ROC-kõver on erinevatele otsustuspiiridele vastavate tundlikkuse ja spetsiifilisuse paaride graafiline esitus hindamaks prognoosi õigsust. [18]

ROC-kõvera (*Receiver Operating Characteristic Curve*) meetod teeb kindlaks kui hästi suudab test või mudel eristada individuaalseid väärtuseid, jaotades nad kahte erinevasse klassi. Antud töö puhul on kaheks erinevaks klassiks väärtus, kas roolis oli auto omanik või mitte. [5]

Kaggle platvorm kasutab tulemuse hindamiseks arvutades pindala ROC-kõvera all (AUC). [17]

Järgnevalt on toodud näide, kuidas käib ROC-kõvera arvutamine, kasutades suvaliselt ühe juhi sõitude andmeid. Näiteandmed, mida on kasutatud ROC-kõvera punktide arvutamiseks on esitatud Tabelis 2.

Tabel 2: ROC-kõvera analüüsi näiteandmed

Sõidu number	**Kas kuulub juhile (tõenäosus = 1)**	**Ennustatud tõenäosus**
1	jah	0.95
2	jah	0.90
3	jah	0.85
4	jah	0.66
5	ei	0.64
6	jah	0.62
7	jah	0.59
8	ei	0.55
9	jah	0.49
10	ei	0.45
11	jah	0.39
12	ei	0.37
13	ei	0.35
14	ei	0.19
15	ei	0.17

Juhul kui sõit ei kuulu juhile, siis loetakse seda positiivseks tulemuseks, vastasel juhul negatiivseks. Järgnevalt tuleb valida otsustuspiirid, mille puhul ennustatakse juhi sõit juhile mitte kuuluvaks. Töö autor otsustas valida otsustuspiiriks vahemiku 0.1 – 1.0.

Lähteandmete järgi koostatud tõeselt positiivsete (TP), tõeselt negatiivsete (TN), vääralt positiivsete (FP) ja vääralt negatiivsete (FN) väärtuste hulgad vastavalt otsustuspiirile on esitatud Tabelis 3.

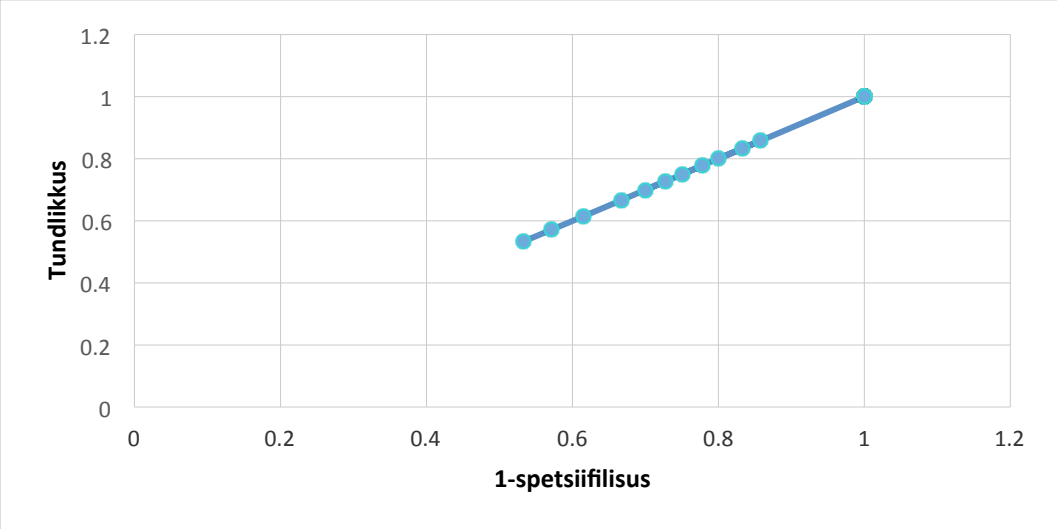
Tabel 3: ROC-kõvera analüüsi abitabel

Otsustuspiir	**TP**	**TN**	**FP**	**FN**
0.1	8	0	7	0
0.18	8	1	6	0
0.2	8	2	5	0
0.36	8	3	4	0
0.38	8	4	3	0
0.42	7	4	3	1
0.47	7	5	2	1
0.52	6	5	2	2
0.57	6	6	1	2
0.6	5	6	1	3
0.63	4	6	1	4
0.65	4	7	0	4
0.75	3	7	0	5
0.87	2	7	0	6
0.93	1	7	0	7
1.0	0	7	0	8

Tabelis 3 esitatud andmete põhjal on võimalik arvutada x-koordinaadid (1-spetsiifilisus) ja y-koordinaadid (tundlikkus). ROC-kõvera punktide arvutamise valemid on esitatud Tabelis 4. Näiteandmete järgi arvutatud ROC-kõvera punktid on esitatud Joonisel 1.

Tabel 4: ROC-kõvera punktide arvutamise valemid

x-koordinaat	$x = 1 - [FP / (FP+TP)]$
y-koordinaat	$y = TP / (FP + TP)$



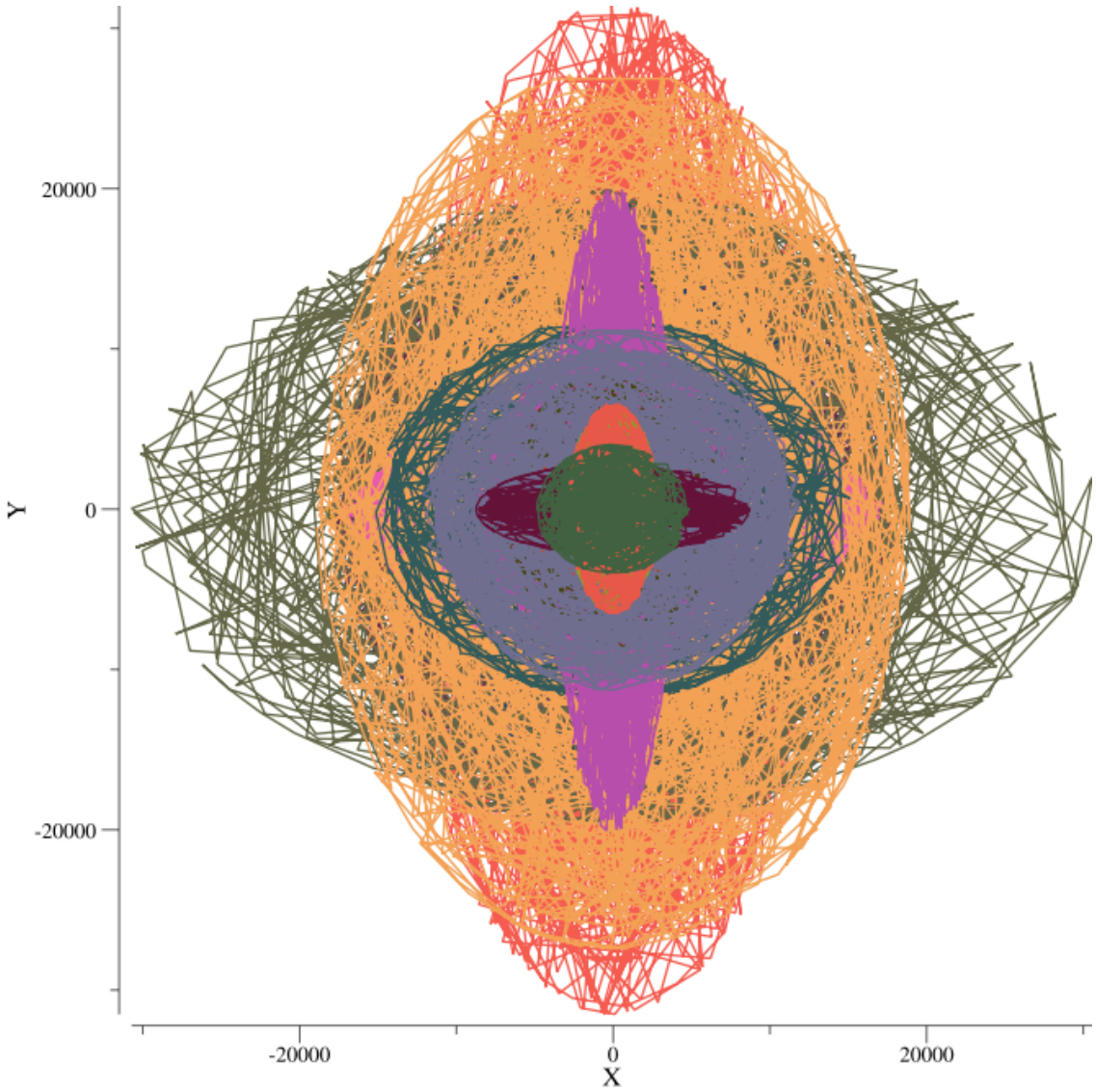
Joonis 1: Näiteandmete abil arvutatud ROC-kõver

5. Andmete esmane normaliseerimine ja sarnaste sõitude leidmine

Igasuguste andmete töötlemise puhul võib ette tulla, et andmete hulka satub valesid või vigaseid andmeid. Kuna antud andmehulga kohta on teada, et koordinaadid on ülesande raskendamiseks pööratud suvaliselt, siis tuli kõigepealt autojuhi kõik sõidud pöörata nii, et nad oleks sama suunaga.

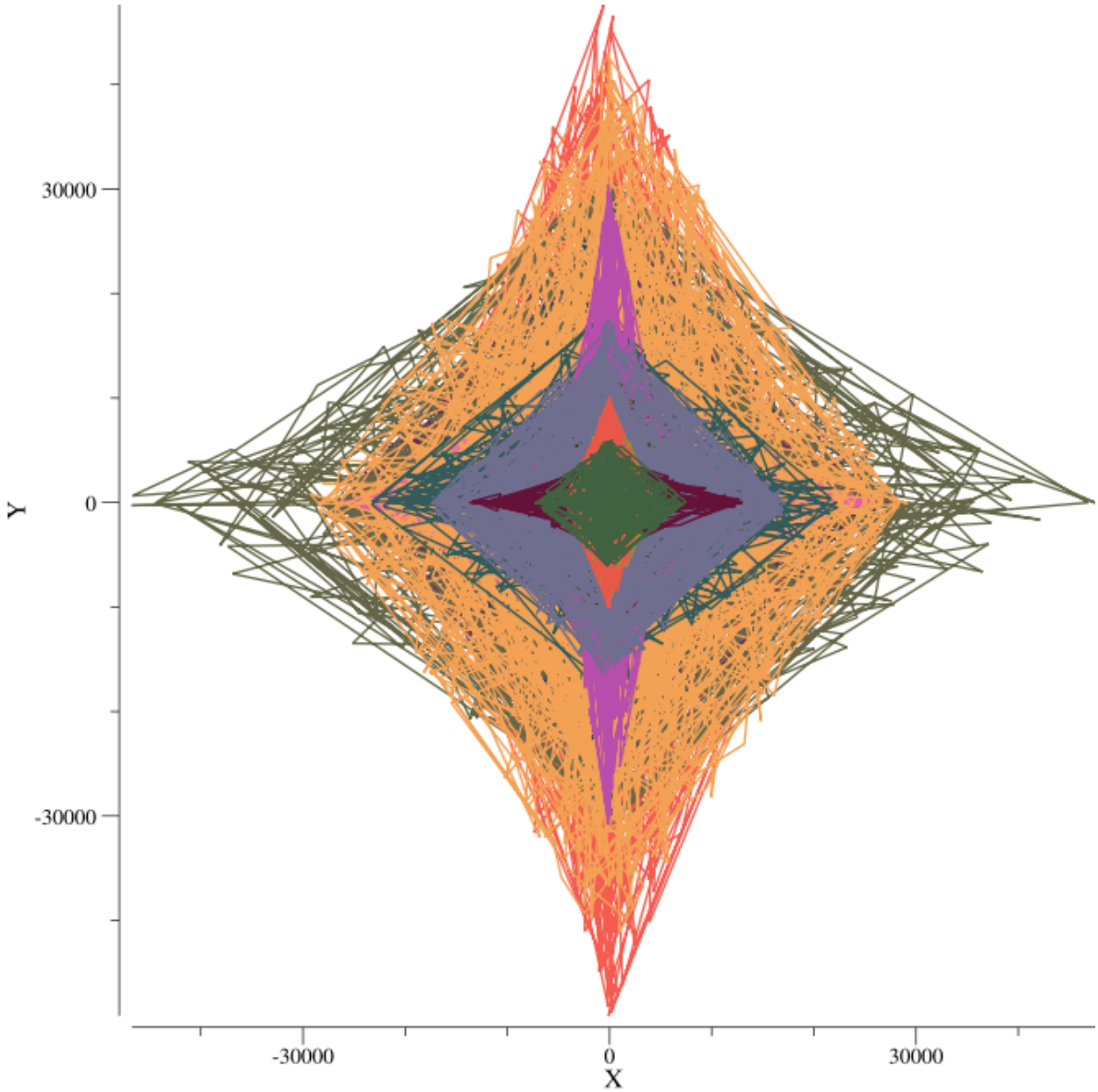
5.1 Andmete normaliseerimine

Enne andmete normaliseerimist otsustas autor teha ühe autojuhi sõitude graafiku, kus kõik sõidud on pandud ühte XY-koordinaatteljestikku. Tulemus on nähtaval Joonisel 2.



Joonis 2: Autojuhi sõitude graafik enne andmete normaliseerimist

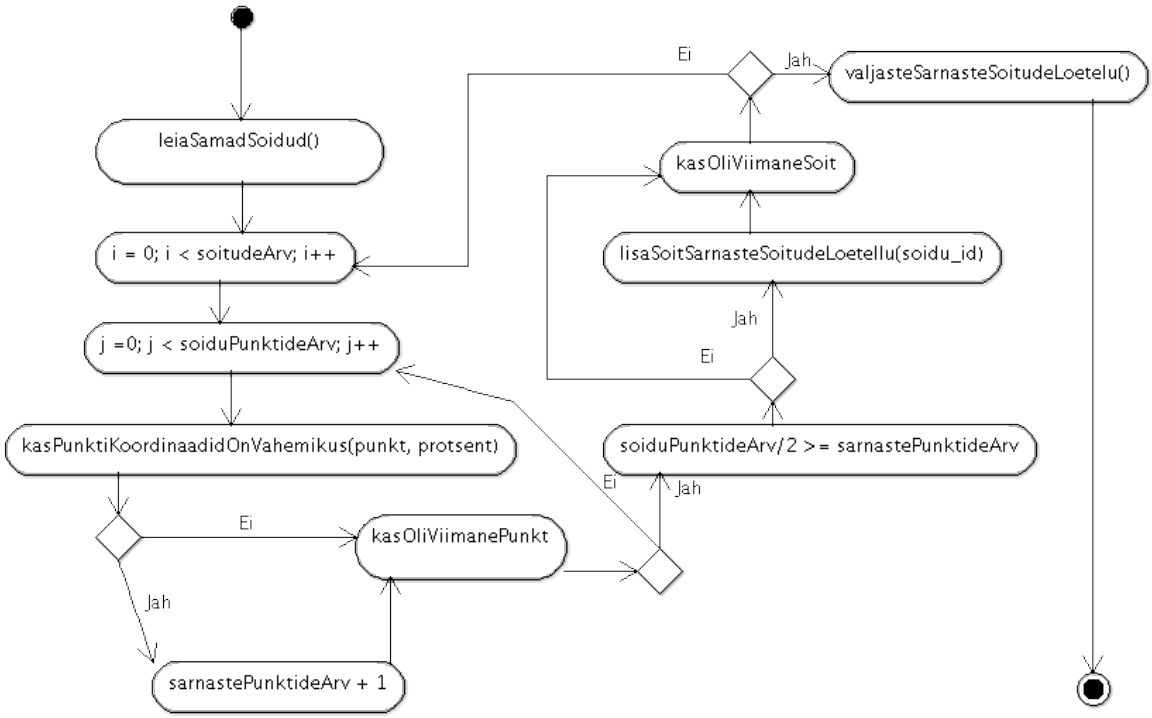
Kuna sõidu andmed on pööratud suvaliselt, siis enne nende reaalset kasutamist on mõistlik need andmed normaliseerida. Antud andmete puhul on nenede esmaseks normaliseerimiseks see, et nad tuli kõik pöörata nii, et nad oleks x-telje suunalised. Samuti tuli järjesatada sõitude koordinaadid, mille esimeseks punktiks ei olnud punkt koordinaatidega (0; 0), vastupidiseks. Pärast andmete normaliseerimist on autojuhi solitude graafik esitatud Joonisel 3.



Joonis 3: Autojuhi sõitude graafik pärast andmete normaliseerimist

5.2 Sarnaste sõitude leidmine

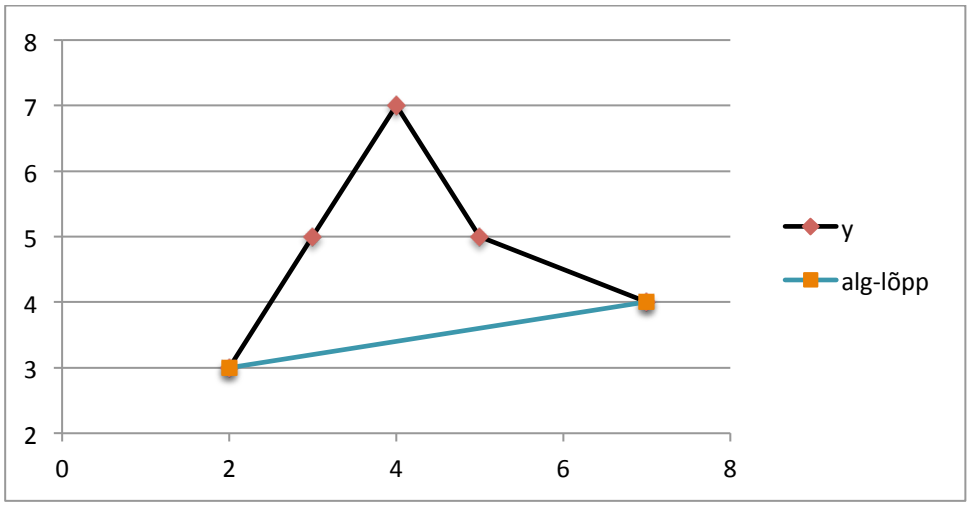
Sarnaste sõitude leidmiseks koostas autor algoritmi, mis võrdleb iga autojuhi sõidu igat järjestikust punkti. Kuna andmete normaliseerimise käigus said kõik punktid pööratud ühte suunda, siis tähendab iga punkti võrdlus siinkohal seda, et võrreldakse iga punkti koordinaate ja kui vähemalt pooled sõidu punktides mahuvad mingi kindla vahemiku sisse, siis järelikult need punktid on pärit sarnastest sõitudest. Autor katsetas punktide võrdlust erinevate vahemikega, milleks olid 5%, 10% ja 15%. Kõige parema tulemuse saavutas autor 10% võrdluse tulemusena. Parimaks tulemuseks oli Kaggle hinnangu järgi 0.52388. Algoritmi tegevusdiagramm on esitatud Joonisel 4.



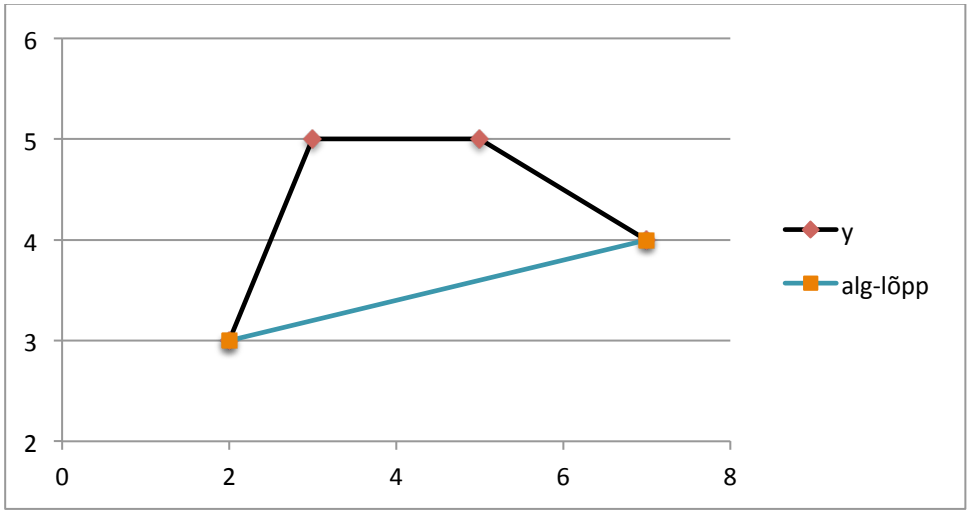
Joonis 4: Sarnaste sõitude leidmise algoritmi tegevusdiagramm

6. Andmete normaliseerimine RDP algoritmi abil

Ramer–Douglas–Peuckeri algoritm on algoritm kõvera punktide vähendamiseks ja seeläbi ühe joone lihtsustamiseks. Algoritmi tööpõhimõte seisneb selles, et kõvera alg- ja lõppunkti vahele tõmmatakse sirge ning leitakse punkt, mis on kõige kaugemal sellest sirgest täisnurga all. Samuti määratakse ära kõige suurem võimalik kaugus ε. Algoritmi tööpõhimõtet on demonstreeritud võttes ε väärtuseks 2 Joonisel 5 ja Joonisel 6. [1]



Joonis 5: Kõver enne töötlust RDP algoritmi abil



Joonis 6: Kõver pärast töötlemist RDP algoritmi abil

6.1 Sõitude ühitamine pärast andmete normaliseerimist RDP algoritmi abil

Pärast andmete normaliseerimist RDP algoritmi abil vähenes iga sõidu punktide arv. Pärast punktide arvu vähendamist eeldas autor, et iga sõidu koordinaatide hulgast eemaldatakse need punktid, mis on sõidu puhul vigased. Näiteks GPS signaal oli nõrk ja asukoha kindlakstegemine polnud kõige täpsem. Autor katsetas andmete normaliseerimist ja sõidu punktide abil andmete analüüsimist erinevate ε väärtustega ning eelduseks oli, et tulemus paraneb. Tegelikkuses aga tulemus hoopis halvenes. Töö autor katsetas andmete normaliseerimist erinevate ε väärtuste korral (5, 10, 15, 20, 25, 30). Kõige parem tulemus kasutades RDP algoritmi oli, kui ε väärtuseks võeti 10. Parim tulemus pärast andmete normaliseerimist RDP algoritmi abil ja sarnaste sõitude algoritmi kasutamist oli 0.50925.

7. Sõitude karakteristikud ja masinõpe

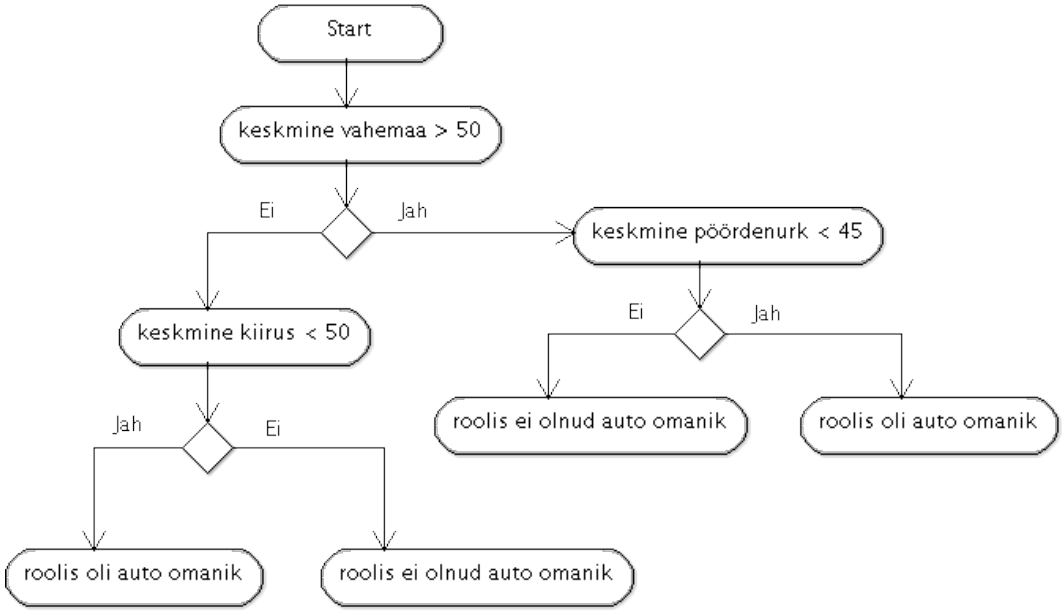
Pärast sõitude ühitamist kasutades punkti koordinaatide võrdlust ja pärast andmete normaliseerimist ei saavutanud töö autor oodatud tulemust. Autor otsustas kasutada RF (*Random Forest*) meetodit autojuhi kindlakstegemiseks. Kuna Go programmeerimiskeeles puudusid töö tegemise ajal sobivad masinõppe teegid, siis otsustas töö autor kasutada masinõppeks programmeerimiskeelt Python.

7.1 Random Forest meetod

RF meetod on erinevateks otstarveteks kasutatav masinõppe meetod. Seda on võimalik kasutada, et ennustada turuolukorra muutuste mõju kliendi käitumisele aga ka näiteks patsiendi haigestumise riski. RF meetodi puhul on võimalik kasutada suurt hulka muutujaid.

RF meetod agregeerib mitmeid otsustuspuid. Iga otsustuspuu klassifitseerib mingi kindla tunnuste hulga järgi need tunnused tõenäosuseks, et uuritav objekt kuulub ühte või teise klassi. RF meetod genereerib otsustuspuud automaatselt. Kuna otsustuspuud genereeritakse pseudojuhuslikult, siis võib juhtuda, et mingi otsustuspuude hulk ei ole nii tähendusrikas, et lahendada etteantud probleemi. [2]

Üks võimalik otsustuspuu, mis võib olla genereeritud RF meetodi poolt on esitatud Joonisel 7.



Joonis 7: Otsustuspuu näide

7.2 Sõidu tunnuste arvutamine

Selleks, et kasutada RF meetodit oli kõigepealt tarvis arvutada tunnuste hulk iga sõidu kohta. Mida rohkem tunnuseid, seda rohkem on võimalik mudelit treenida ja genereerida otsustuspuid, mis annavad võimalikult täpse tulemuse.

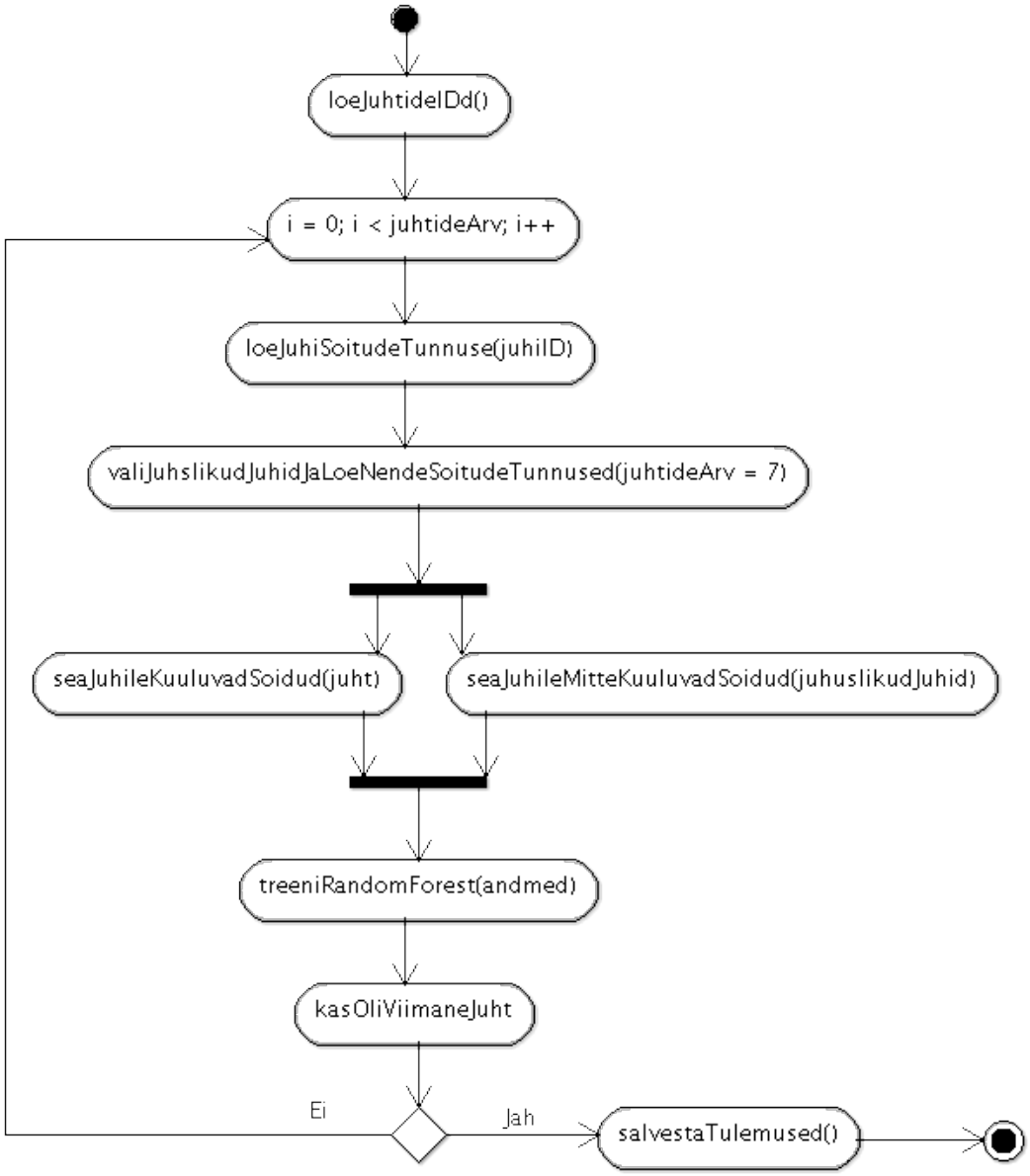
Töö autor arvutas iga sõidu kohta 5 tunnust. Iga sõidu kohta arvutatud tunnused on esitatud Tabelis 5.

Tabel 5: Sõitude kohta arvutatud tunnused

Sõidu pikkus
Keskmine sõidu punktide vaheline kaugus
Keskmine pöördenurk
Keskmine kiirendus punktide vahel
Keskmine kiirus

7.3 Mudeli treenimine

Pärast tunnuste arvutamist oli tarvis RF meetodit kasutades treenida mudelit. Mudeli treenimiseks oli tarvis andmed jagada tõesteks ja valedeks. Kuna töö autor teadis, et enamik autojuhi sõitudest kuuluvad auto omanikule, siis seati kõik autojuhi sõidud tõesteks. Seejärel valiti pseudojuhuslikult välja seitse autojuhti, kelle sõidud seati kõik valedeks. Seejärel treeniti mudelit 32 korda. Treenimisalgoritmi tegevusdiagramm on esitatud Joonisel 8.



Joonis 8: Treenimisalgoritmi tegevusdiagramm

Pärast algoritmi treenimist 32 korda sai sai töö autor 32 erinevat tulemust, millest tuli arvutada lõpptulemus. Kuigi soovitatavaks tulemuste kombineerimise meetodiks RF-meedodi puhul on kasutada aritmeetilist keskmist, otsustas töö autor proovida ka muid meetodeid. Erinevate meetodite tulemused on esitatud Tabelis 6. Kõige parema tulemuse saavutas saavutas töö autor arvutades kõikide treeningtulemuste aritmeerilise keskmise. Parimaks tulemuseks Kaggle veebilehe hinnangu järgi 0.89917.

Tabel 6: Masinõppe treeningtulemuste kombineerimise teel saadud tulemused

Meetod	**Tulemus**
Aritmeetiline keskmine	0.89917
Geomeetriline keskmine	0.89905
Mediaankeskmine	0.89804

8. Analüüsi tulemuste tõlgendused ja edasised uurimissuunad

Töös kasutatud analüüsi meetodite tulemusena selgus, et võrdlemisi lihtsate meetodite puhul ei suutnud töö autor saavutada piisavalt head tulemust. Sõitude koordinaatide võrdlus võrreldes ainult punktide koordinaate ei olnud ülesande lahendamiseks piisav. Selle põhjuseks võiks pidada seda, et tehes sõitude võrdlust on väga raske kontrollida tulemuse õigsust. Kaggle hinnangu järgi saab hinnata ainult terviklikku tulemust, kuid iga üksiku autojuhi sõidu ennustuse õigsus hindamine on keeruline.

Pärast andmete normaliseerimist RDP algoritmi abil oli töö autor veendunud, et tulemus paraneb. Paraku tulemus aga halvenes. Kuigi RDP algoritmi sai proovitud erinevate ε väärtuste korral hinnang ei paranenud.

Masinõppe meetodi puhul sai autor piisavalt hea tulemuse ja jäi sellega rahule. Töö edasiarendamiseks oleks võimalik jätkata masinõppe treenimist kasutades kas mõnda muud masinõppe treenimise meetodit või siis treenida sama meetodiga, kuid kasutades teist lähenemist.

Näiteks oleks võimalik iga treenimise iteratsiooni korral kasutada eelmisest iteratsioonist saadud andmeid. Oleks võimalik paika panna nivoo, mille puhul arvestatakse, et sõit kuulub juhile. Seejärel tuleks seada autojuhile kuuluvateks sõitudeks need, mis arvestades valitud nivood ja eelmise iteratsiooni ennustust kuuluvad antud juhile ning treenida mudelit uuesti. Kuna üksiku juhi sõitude hinangute kohta Kaggle hinnangust tagasisidet ei saa, siis on ka selle meetodi puhul keeruline tulemust parandada, kuid kasutades nivood, mille puhul sõit juhile kuuluvaks määratakse ja uue mudeli treenimise käigus on see siiski võimalik.

Teiseks võimalikuks lähenemiseks oleks jagada autojuhi sõidud mitmeks osaks ja jätta kõrvale teiste autojuhtide sõidud mudeli treenimise ajal. Näiteks võttes autojuhi, kellel on 200 sõitu, märkides esimese iteratsiooni käigus temale kuuluvateks sõitudeks 10% tema sõitude valimite hulgast. Seejärel treenida mudelit. Järgmise iteratsiooni käigus korrata sama tegevust, kuid märkides sellel korral autojuhile kuuluvateks sõitudeks järgmised 10% sõitudest tema valimite hulgast. Pärast seda on võimalik kõikide iteratsiooni tulemuste

puhul arvutada lõpptulemus ning treenida mudelit märkides autojuhile kuuluvateks sõitudeks ennustatud tulemused, mille kohaselt sõit kuulub juhile.

9. Kokkuvõte

Töö eesmärgiks oli analüüsida autosõitude andmeid ning ennustada, millised sõidud juhi sõitude valimist kuuluvad antud juhile (auto omanikule). Eesmärgiks oli realiseerida algoritmid Go ja Pythoni programmeerimiskeeltes, mille abil saavutatakse Kaggle veebilehe hinnangu järgi skooriks vähemalt 0.75.

Töö tulemusena on valminud andmete normaliseerimise ja analüüsi algoritmid, mis leiavad juhi sarnased sõidud võrreldes sõitude koordinaate. Normaliseerimiseks pöörati sõidud nii, et nende suund oleks x-telje poole ning kasutati ka RDP algoritmi kõvera punktide vähendamiseks. Samuti on töö tegemise käigus ennustatud sõidu kuuluvust juhile kasutades *Random Forest* masinõppe meetodit.

Töö tegemise käigus ei saavutanud töö autor piisavalt head tulemust sõidu jooksul läbitud punktide koordinaatide võrdluse abil. Töö autor peab tulemuseks ka seda, et antud ülesande puhul halvenes tulemus, kui andmete normaliseerimiseks kasutati RDP algoritmi.

Random Forest masinõppe meetodi kasutamisel saavutas autor piisavalt hea tulemuse, mis ületas algselt püstitatud eesmärki. Mudelit treeniti 32 korda ja tulemuste kombineerimisel kasutati erinevaid võtteid, milleks olid tõenäosuste aritmeetiline-, geomeetriline- ja mediaankeskmine.

Töö tulemusena saavutati Kaggle veebilehe hinnangu järgi 0.89917. Algselt püstitatud eesmärgiks oli saavutada hinnanguks vähemalt 0.75.

Antud magistritööd on võimalik edasi arendada kasutades muid masinõppe treenimise algoritme või *Random Forest* mudelit treenida valides juhile mitte kuuluvateks sõitudeks sõidud juhi valimi hulgast, mitte juhuslikult valitud juhtide sõite. Samuti on võimalik kasutada eelmiste iteratsioonide andmeid mudeli korduvaks treenimiseks.

Summary

The objective of the current thesis was to analyse data of trips done by drivers and predict which trips belong to the car owner. The goal was to develop algorithms in Go and Python programming languages, to achieve a Kaggle's assessment score of at least 0.75.

As a result, normalizing and analysis algorithms have been developed, which match a driver to their similar trips comparing the trips coordinates. To normalize, the trips were rotated so that their direction was towards the x-axis. Also, the Ramer–Douglas–Peucker algorithm was used to lessen the number of points on the curve. Furthermore, Random Forest machine learning was used to predict the trip's affiliation to the driver.

Over the course of this thesis, the author did not achieve a satisfactory result by using a comparison of the trip's coordinates. The author observed that by using the RDP algorithm for normalizing the data, the result deteriorated. Using the Random Forest machine learning method, the author achieved a satisfactory result, which exceeded the initially set goal. The model was trained 32 times and different methods, arithmetic- , geometric- and median averages, were used to combine the results.

As a result, a Kaggle's assessment of 0.89917 was achieved. The initial goal was to achieve an assessment of at least 0.75.

It is possible to further develop this thesis using alternative machine learning algorithms or to train the Random Forest model by choosing trips that do not belong to the driver, from the driver's trip data, not using randomly picked trips done by other drivers. Also, it would be possible to use data form previous iterations to repeatedly train the model.

Kasutatud kirjandus

1. Correct Ramer Douglas Peucker Algorithm. [WWW]. http://karthaus.nl/rdp/ (05.05.2015)

2. ŷhat | Random Forests in Python. [WWW]. http://blog.yhathq.com/posts/random-forests-in-python.html (08.05.2015)

3. exploratory data analysis. [WWW]. http://economy_en_ru.academic.ru/23866/exploratory_data_analysis (10.05.2015)

4. Statistics Glossary: R. [WWW]. http://www.statsoft.com/textbook/statistics-glossary/r#Random%20forests (10.05.2015)

5. Cook, N., Statistical Evaluation of Prognostic versus Diagnostic Models: Beyond the ROC Curve. — Clinical Chemistry. 2008, 54 (1), 17-23

6. Statistics Glossary: D. [WWW]. http://www.statsoft.com/textbook/statistics-glossary/d#Decision%20Trees (10.05.2015)

7. e-Teatmik: IT ja sidetehnika seletav sõnaraamat. [WWW]. http://www.vallaste.ee/index.htm?Type=UserId&otsing=5325 (09.05.2015)

8. What is Big Data Analytics? A Webopedia Definition. [WWW]. http://www.webopedia.com/TERM/B/big_data_analytics.html (12.05.2015)

9. Description - Driver Telematics Analysis | Kaggle. [WWW]. https://www.kaggle.com/c/axa-driver-telematics-analysis (20.04.2015)

10. The Go Project - The Go Programming Language. [WWW]. https://golang.org/project/ (20.04.2015)

11. Arvutusklaster < Juhendid < IT-teenused < Tugistruktuur < Tallinna Tehnikaülikool - Sinu elustiil!. [WWW]. http://www.ttu.ee/?id=81091 (01.05.2015)

12. About Python | Python.org. [WWW]. https://www.python.org/about/ (09.05.2015)

13. NumPy - Numpy. [WWW]. http://www.numpy.org/ (09.05.2015)

14. scikit-learn: machine learning in Python - scikit-learn 0.16.1 documentation. [WWW]. http://scikit-learn.org/stable/index.html (12.05.2015)

15. 11.1. pickle — Python object serialization - Python 2.7.10rc0 documentation. [WWW]. https://docs.python.org/2/library/pickle.html (09.05.2015)

16. Attach submission - Driver Telematics Analysis | Kaggle. [WWW]. http://www.kaggle.com/c/axa-driver-telematics-analysis/submissions/attach (20.04.2015)

17. Evaluation - Driver Telematics Analysis | Kaggle. [WWW]. http://www.kaggle.com/c/axa-driver-telematics-analysis/details/evaluation (20.04.2015)

18. Roos M., Hüpertensiooni geneetilise riskiskoori prognoosivõime hindamine: analüüs Tartu Ülikooli Eesti Geenivaramu andmebaasi põhjal, Tartu, Tartu Ülikool, 2013.

Lisa 1

Töö lisaks on CD-plaat, mille peal on käesoleva magistritöö jooksul andmete analüüsi tarbeks realiseeritud algoritmid Go ja Pythoni programmeerimiskeeltes. Iga realiseeritud programm omab RADME.md faili, kus on kirjeldatud, kuidas käib programmi kasutamine.