



produced by DALL-E

Praktiline andmeteadus 2023

Projekti lõpparuanne

Erki Jõekalda

https://github.com/erkirb/PAT2023_projekt

Sissejuhatus

Kursuse projekti kavandama asudes võtsin ideede genereerimisel ning alternatiivide otsingutel algseks eesmärgiks soovi, et võimalusel võiks teema seostuda minu tööalase valdkonnaga, ehk olla ühel või teisel viisil seotud finantsturgude suunaga.

Üsna koheselt jõudsin otsusele, et ma ei võta töö aluseks väga konkreetseid oma tööalaste vastutusvaldkondade andmeanalüütika väljakutseid, ühelt poolt loomulikult andmetike konfidentsiaalsuse problemaatika tõttu, kuid teisalt ka väga sügava valdkonna spetsiifika tõttu. Nimetatud kahe aspekti koosmõjul muutuks projekti tulemuste tutvustamine laiemale grupile seosetuks hämaks.

Kuigi see ei tulene otseselt minu vastutusvaldkondade spetsiifikast, siis finantsturgude suunal üldiselt on andmeteaduse kontekstis tegemist väga palju mingisuguste andmeridade põhiste ennustamistega. Seega uurisin lähemalt erinevaid internetis leiduvaid allikaid mis käsitlesid masinõppe meetodikate rakendamist erinevate kauplemisalgoritmide loomisel ja treenimisel. Kaalusin tõsiselt näiteks hetkel toimuvat Kaggle võistlust [Optiver - Trading at the Close](#). Teise alternatiivina mõtlesin võimalustele ja enda võimekusele püüda ehitada lihtsakoelisem mudel, mis püüaks ennustada erinevate globaalse ja regionaalsete majandusindikaatorite käitumise baasilt Euroopa ja USA keskpankade intressipoliitika kujunemist ning läbi selle nii eraisikutele kui ettevõtetele pakutavate laenude aluseks olevate referentsintressimäärade (EURIBOR etc) oodatavat dünaamikat.

Mõlema ülesande püstituse puhul on tegemist väga keeruka eesmärgiga. Kuna finantsturgude käitumine on pidevalt muutuv ja sõltub väga paljudest mitte reeglipäraselt käituvatest ja ennustamatutest faktoritest, siis ajalooliste andmete pealt treenitud tulevikku suunatult stabiilselt edukalt toimivate kauplemismudelite loomine on väga väljakutsuv ülesanne millele juhtivad investeerimispannad kulutavad aastas miljardeid dollareid. Antud kontekstis on huvitav näiteks värskest oktoobris avaldatud informatsioon, et läänemaailma juhtiv pank JP Morgan on otsustanud lõpetada klientidele suunatud masinõppel aastaid treenitud tehisnärvivõrkude põhiste valuutakauplemise mudelite kasutamise. Põhjuseks on toodud mudelite keerukust

ning probleeme nende käitumise tõlgendamisel ja selgitamisel. Mudelite tehtavad otsused hakkasid sarnanema aastkümnend kasutusel olnud klassikalisemate kauplemisalgoritmide toimimisega. See on üks hea näide hetkel finantsturgudel toimuva massiivse AI-maania valguses, mis viitab masinõppe võimekuse piiridele ning fundamentaalsetele probleematiikatele tänasel päeval.

Seetõttu pärast põhjalikku kaalutlust hindasin Optiveri võistluses osalemise hetkel liiga ambitsioonikaks ettevõtmiseks arvestades minu praeguseid eeldusi. Hinnates teist alternatiivset turuintresside ennustusmodeli potentsiaali tuvastasin mitmeid sobivaid allikaid vajalike majandusnäitajate ja finantsturgude referentsindeksite aegridade hankimiseks (Refinitiv, Bloomberg, WorldBank etc.). Siiski, hinnates realistlikult enda ajalisi ressursse ning teisalt andmestiku kogumiseks, sünteesiks, sobivaks normaliseerimiseks ja kalibreerimiseks eeldatavalt vajaminevat aega, samuti oskamata hinnata kas üldse ja millise töömahu põhiselt oleks võimalik mingisuguse vähegi potentsiaali omava mudelini jõuda, tegin lõpuks järelduse, et hetkel käesolevas ajaraamistikus ma tõenäoliselt ei suudaks ennast rahuldava tulemuseni jõuda. Võimalik, et püüan seda ideed siiski millalgi hiljem edasi viia.

Pärast ülal toodud analüüsi ja kaalutluste läbimist võtsin siiski suuna veidi standardiseerituma andmeteandusprojekti leidmisele mis vastaks paremini hetke võimekusele ja oleks teostatav eeldatavas ajaraamis. Enda jaoks sedastasin projekti valikus ka eesmärgi, et töö võiks pigem aidata kinnistada ja praktiseerida seni kursuse käigus läbitud ja vähem või rohkem juba proovitud meetodikaid. Antud eesmärgist lähtuvalt valisin Kaggle keskkonnast välja aktiivse [Spaceship Titanic](#) võistluse.

Andmete kirjeldus ja analüüs

Spaceship Titanic võistluse näol Kaggle keskkonnas on tegemist klassikalise Titanicu laevahuku juhtumi replikatsioonidga fantaseeritud tingimustes, mis on andnud võimaluse luua originaaliga võrreldes suurem ja keerukam andmestik andmeteanduse meetodikate rakendamiseks. Seega antud projekti raames jääb vahele andmeallikate otsimise ja andmestiku kogumise protsess.

Nagu ka klassikalise Titanicu näite puhul on Spaceship Titanic ülesande lõppeesmärgiks treenida mudel mis suudaks maksimaalse täpsusega ennustada kas kosmoselaeva reisija transporditi teise dimensiooni või mitte. Me alustame projekti andmetega tutvumise ja nende mõistmise protsessiga. Kuna sisendi puhul on tegemist fantaasiaga, siis käesolev faas on väga oluline alus tulemusteni jõudmisel. See tähendab, et me ei saa lähtuda ühestki tavalooigikale alluvast eeldusest.

Ülesande sisendina on meile antud treeningandmestik, mis sisaldab järgnevaid välju:

PassengerId: iga reisija unikaalne tunnus fromaadis gggg_pp mis moodustub reisija grupi koodist (gggg) ja nende järjekorra numbrist grupis

HomePlanet: reisija päritolu planeet

CryoSleep: tõeväärtuse kujul tunnus, mis määratleb kas reisija valis oma reisi ajaks vegetatiivse *cyrosleep* oleku, ühtlasi viitab, et reisija veedab kogu reisi aja oma kajutis

Cabin: kajuti number mis viitab ka kajuti asukohale kosmoselaevas (formaad: [kajuti tüüp/kajuti number/kajuti asukoht kosmoselaevas praemal või vasakul küljel])

Destination: vastava isiku reisi sihtkoht

Age: reisija vanus

VIP: tõeväärtuse formaadis tunnus reisija VIP staatuse kohta

RoomService, *FoodCourt*, *ShoppingMall*, *Spa*, *VRDeck*: atribuudid mis näitavad reisija poolt tehtud kulutusi kosmoselaeva erinevates teenuspunktides

Name: reisija nimi

Transported: tõeväärtusena esitatud tunnus kas reisija transporditi teise dimensiooni või mitte (antud ülesandes 'target label')

[Github repository: [spaceship-titanic_start.ipynb](#)]

Loeme treeningfaili andmeraami ning teostame esmased vaatlused.

```
In [2]: df.head()
```

```
Out[2]:
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	False
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	True
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	False
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	Solam Susent	False
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	Willy Santantines	True

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId      8693 non-null   object
1   HomePlanet       8492 non-null   object
2   CryoSleep        8476 non-null   object
3   Cabin            8494 non-null   object
4   Destination       8511 non-null   object
5   Age              8514 non-null   float64
6   VIP              8490 non-null   object
7   RoomService      8512 non-null   float64
8   FoodCourt        8510 non-null   float64
9   ShoppingMall     8485 non-null   float64
10  Spa              8510 non-null   float64
11  VRDeck           8505 non-null   float64
12  Name             8493 non-null   object
13  Transported      8693 non-null   bool
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
```

```
df.describe()
```

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

Näeme, et meie treeningandmestikus on 8693 rida ning veerudes esineb erinevad andmetüüpe. Varasemast on meile teada, et mõnedes veergudes esineb komposiitandmeid.

Oluline asjaolu mille teeme kindlaks ka andmestiku visuaalsel vaatlusel ja analüüsil Excelis on see, et andmetes ei tundu esinevat selgeid kõrvalekaldeid või konkreetselt genereeritud vigu. Seega antud juhul me ei pea tegelema andmekvaliteedi probleemide haldusega.

Järgnevalt uurime kui palju esineb meil andmeraamis puuduvaid andmeid.

```
nan_counts_per_column = df.isna().sum()
print(nan_counts_per_column)
```

```
PassengerId      0
HomePlanet       201
CryoSleep        217
Cabin            199
Destination       182
Age              179
VIP              203
RoomService      181
FoodCourt        183
ShoppingMall     208
Spa              183
VRDeck           188
Name             200
Transported       0
dtype: int64
```

```
number_of_rows_with_nan = df.isna().any(axis=1).sum()
print(f"Number of rows with at least one NaN value: {number_of_rows_with_nan}")
```

```
Number of rows with at least one NaN value: 2087
```

Tulemused ülal näitavad, et siin esineb meil vajakajäämine millega peaksime kindlasti tegelema. Peaaegu kõigis veergudes esineb u. 200 tühja välja. Mis on veel olulisem, ridu kus esineb vähemalt üks tühi väli on meil kokku 2087 ja see moodustab 24% kogu meie andmestikust.

Seega kui läheneda kõige lihtsama põhimõttega, eemaldades kõik read kus esineb tühi väli, siis me kaotaksime selgelt liiga suure osa oma treeningandmestikust. Siit tulenevalt me peaksime projekti käigus panustama tühjade väljade asendamisele.

Tunnuste analüüs, atribuutide loomine

Järgnevalt võtame ette tööloõigu kus me vaatleme iga veeru tunnuseid eraldi, nende jaotust ning korreleeruvust meie poolt ennutatava atribuudi *Transported* väärtustega. Esmalt uurime otse vaadeldavaid atribuute ning nende sobivust ning potentsiaalset kasutegurit mudelite treenimisel.

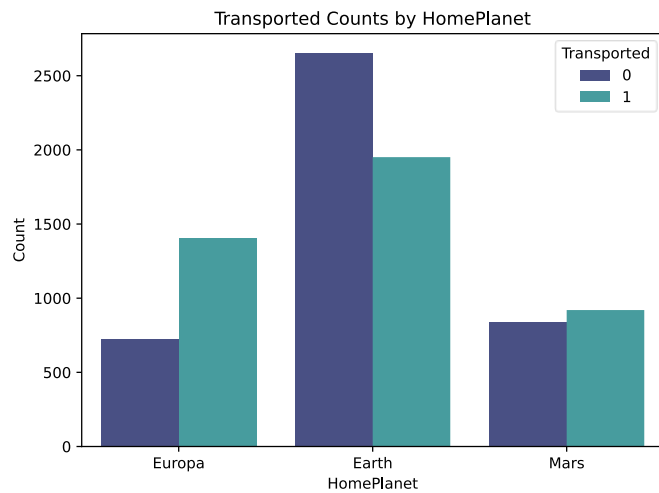
Enne analüüsi juurde liikumist tahaksin ka mainida, et projekti käigus võtsin atribuutide omavaheliste seoste uurimisel lisaks korrelatsioonile kasutusele ka *Mutual Information* kontseptsiooni. Tegemist ei olnud mulle varasemast tuttava terminiga ning idee ja huvi selle kasutamise proovimiseks sain Kaggle artiklist:

[Why you should not use correlation | Kaggle](#)

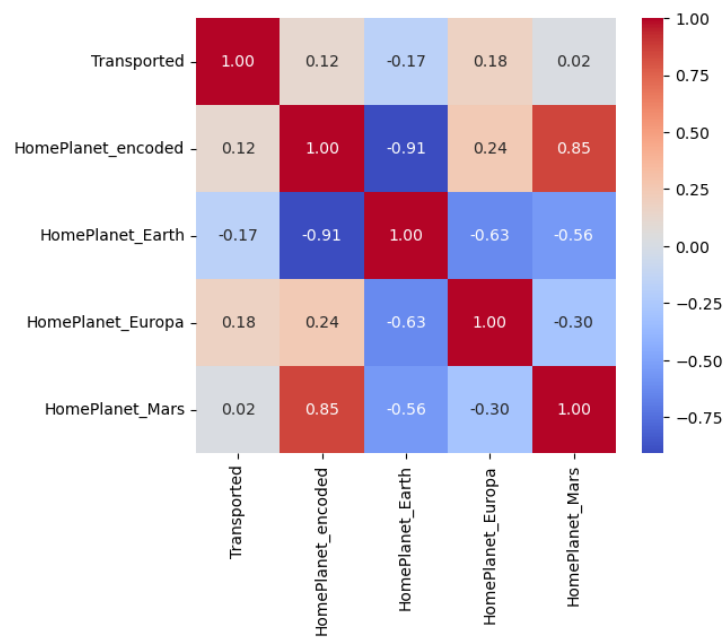
Edasise analüüsi käigus kalkuleerisin valdavalt enamike tunnuste omavahelise seose tuvastamiseks nii korrelatsiooni kui *mutual information (Mi)* skoorid ning kasutasin otsuste tegemisel mõlemaid sisendeid.

[Github repository: [spaceship-tit_addit_feature_selection.ipynb](#)]

HomePlanet



Graafikult on visuaalselt näha, et selgelt esineb teatud seos *HomePlanet* ja *Transported* atribuutide väärtuste vahel

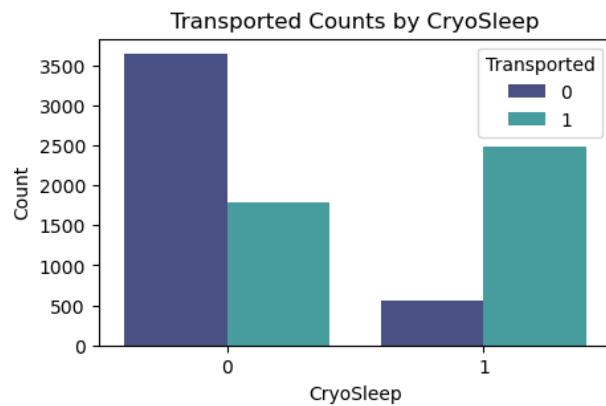


```
from sklearn.metrics import mutual_info_score
mi = mutual_info_score(HomePlanet_feat_df['HomePlanet_encoded'], HomePlanet_feat_df['Transported'])
print(f"Mutual Information Score: {mi}")
```

Mutual Information Score: 0.019375549487534616

Mõõdukat seost kinnitavad ka korrelatsiooni ja Mi skoorid

CryoSleep



CryoSleep ja *Transported* vahel on näha väga selge seos.

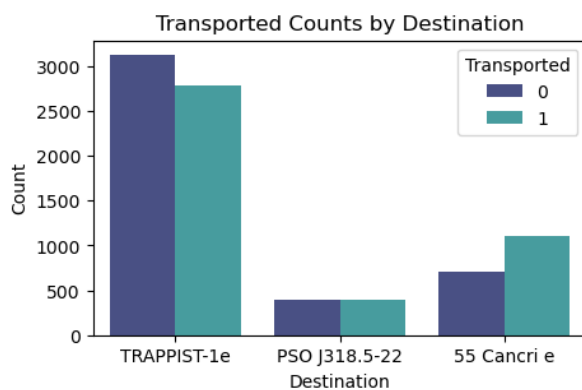
```
# Convert 'Transported' to numeric (True to 1 and False to 0)
CryoSleep_feat_df['Transported'] = CryoSleep_feat_df['Transported'].astype(int)
CryoSleep_feat_df['CryoSleep'] = CryoSleep_feat_df['CryoSleep'].astype(int)

# Calculate correlations
correlation_matrix = CryoSleep_feat_df[['CryoSleep', 'Transported']].corr()
print(correlation_matrix)
```

	CryoSleep	Transported
CryoSleep	1.000000	0.468645
Transported	0.468645	1.000000

Mutual Information Score: 0.11644931405098938

Destination



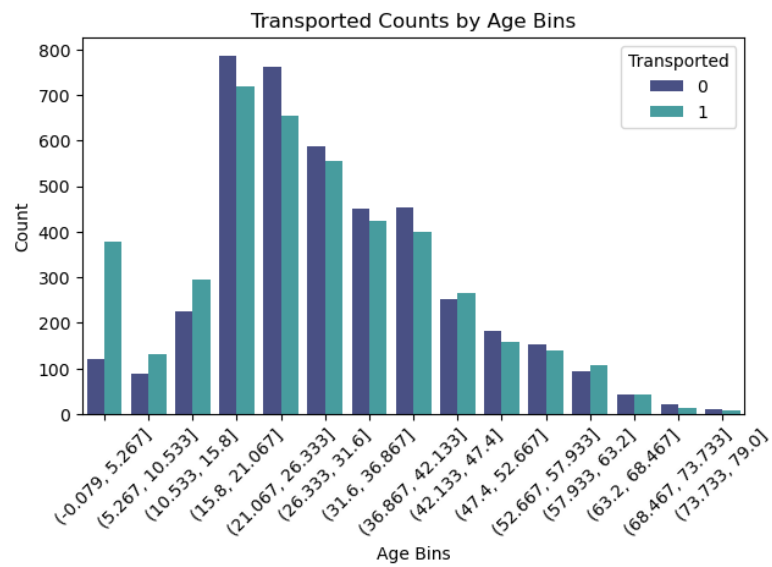
Destination ja *Transported* puhul esineb kerge seos

Korrelatsioon:

	Destination_encoded	Transported
Destination_encoded	1.000000	-0.109806
Transported	-0.109806	1.000000

Mutual Information Score: 0.006292699419124942

Age



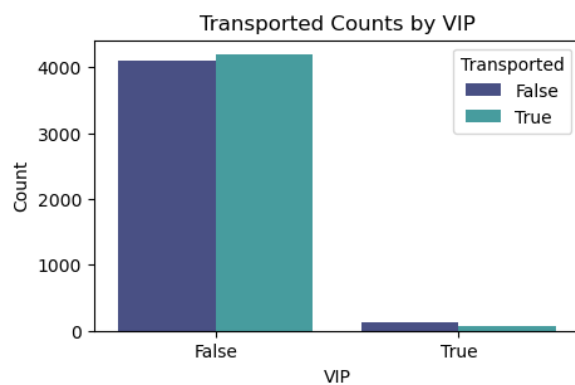
Korrelatsioon:

	Age	Transported
Age	1.000000	-0.075026
Transported	-0.075026	1.000000

Mutual Information Score: 0.015895228769092537

Age atribuudi seos tundub olema vähene, kuid näiteks kõige nooremas vanuserühmas on mõju üsna oluline.

VIP



	VIP	Transported
VIP	1.00000	-0.03765
Transported	-0.03765	1.00000

Mutual Information Score: 0.000714937192564602

VIP seos *Transported* atribuudiga on peaaegu olematu

[Github repository: [spaceship-t EDA services analy.ipynb](#)]

RoomService, FoodCourt, ShoppingMall, Spa, VRDeck

Antud punktis me alustame juba lisaatribuutide loomisega. Me teeme hüpoteesi, et luues olemasolevate tunnuste baasil uusi atribuute me võime sattuda lahenduse peale, mis aitab mudelil veel paremini ennustusi genereerida. Kuna kõik ülal mainitud tunnused on kulud mida reisijad on teinud erinevatele kosmoselaevas pakutavatele teenustele, siis me võime luua sellelt baasilt veel järgnevad lisaatribuudid:

All_Zero – atribuut viitab kas reisija tegi laevas mõne kulutuse või ei kulutanud üldse midagi

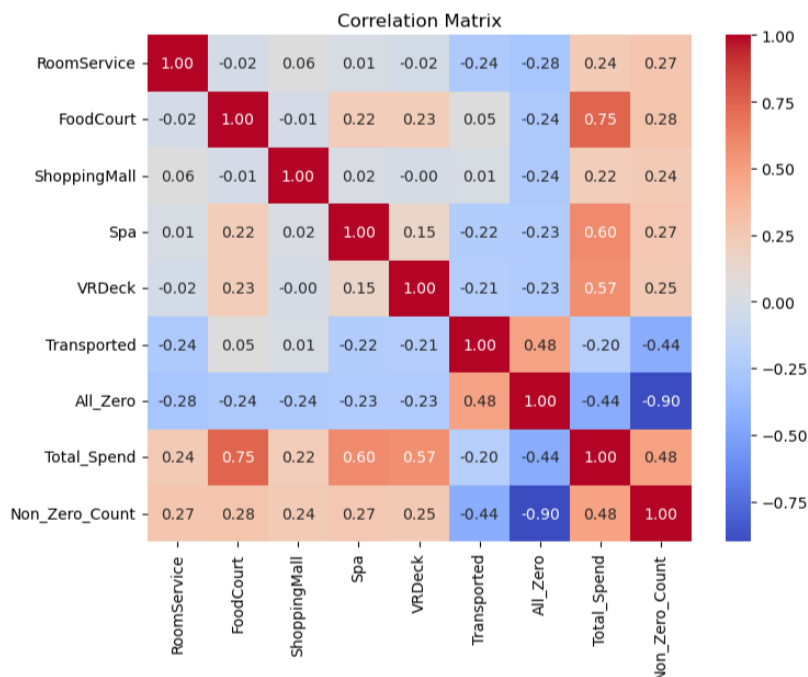
Total_Spend – summeerime kokku kõik laevas tehtud kulutused

Non_Zero_Count – loeme reisija põhiselt kokku mitmele erinevale teenusele ta laevas olles raha kulutas

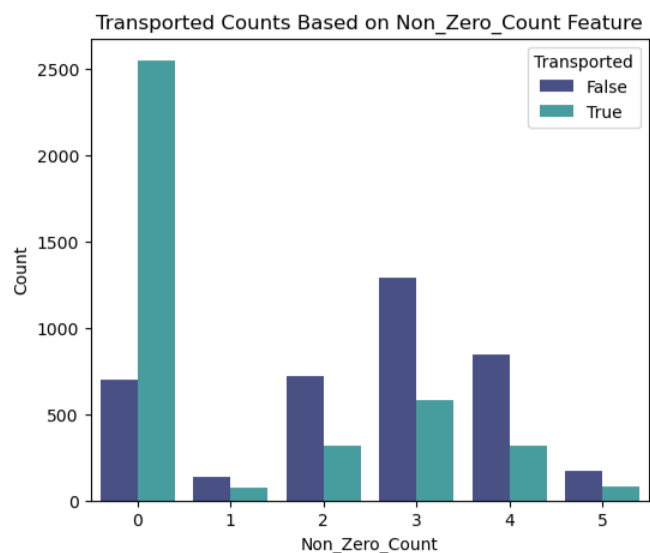
Andmeraam koos uute loodud tunnustega näeb välja järgnev:

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported	All_Zero	Total_Spend	Non_Zero_Count
0	0.0	0.0	0.0	0.0	0.0	False	True	0.0	0
1	109.0	9.0	25.0	549.0	44.0	True	False	736.0	5
2	43.0	3576.0	0.0	6715.0	49.0	False	False	10383.0	4
3	0.0	1283.0	371.0	3329.0	193.0	False	False	5176.0	4
4	303.0	70.0	151.0	565.0	2.0	True	False	1091.0	5

Uurime nüüd lähemalt erinevate kulutustega seotud tunnuste seoseid *Transported* atribuudiga. Genereerime korrelatsioonide maatriksi:



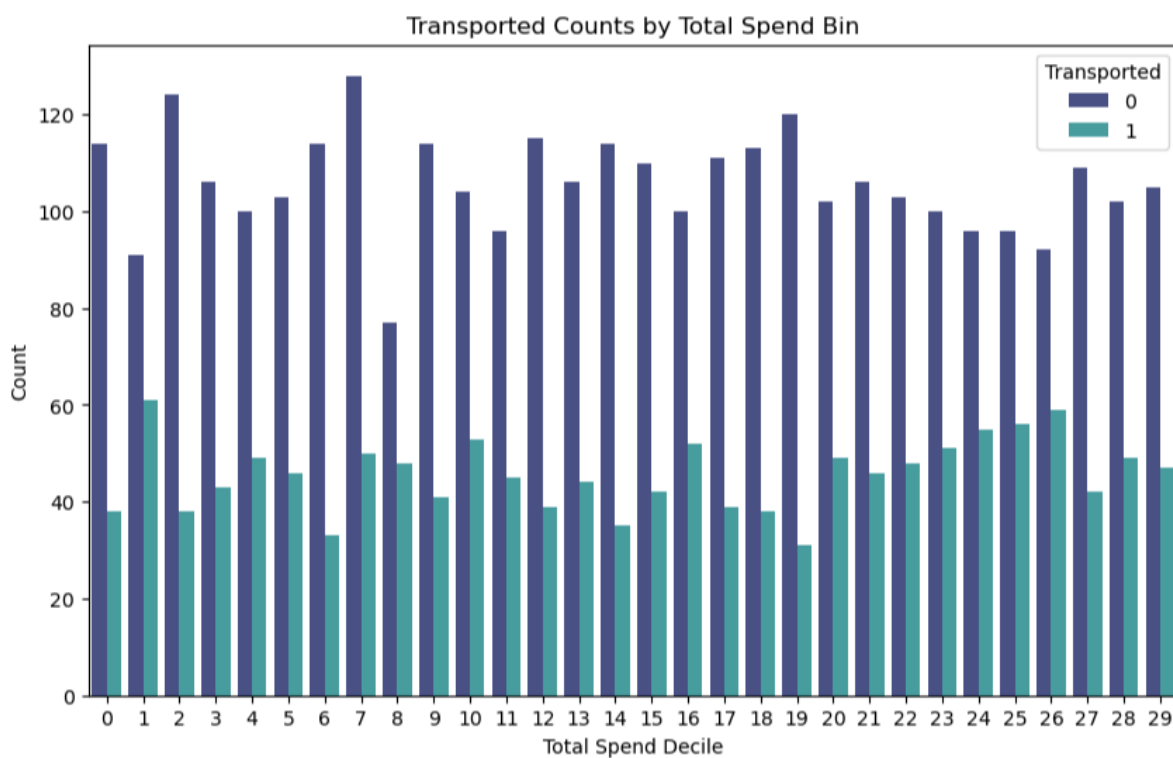
Näib, et *Non_Zero_Count* tunnusel on kõrge korrelatsioon *Transported* vastu, kuid järgnev graafik näitab, et see on suuresti mõjutatud 0-kulutust teinud reisijate poolt. Seda asjaolu adresseerib meil aga juba *All_Zero* atribuut:



Genereerime ka Mi skoorid:

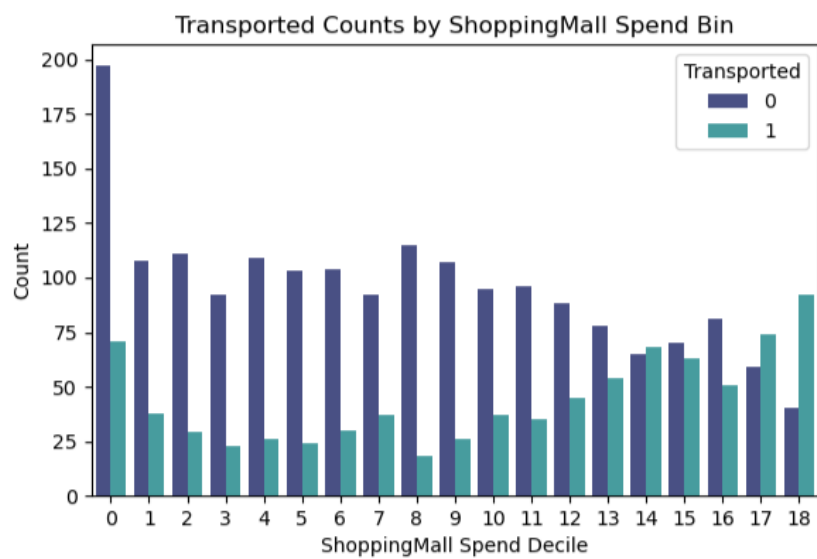
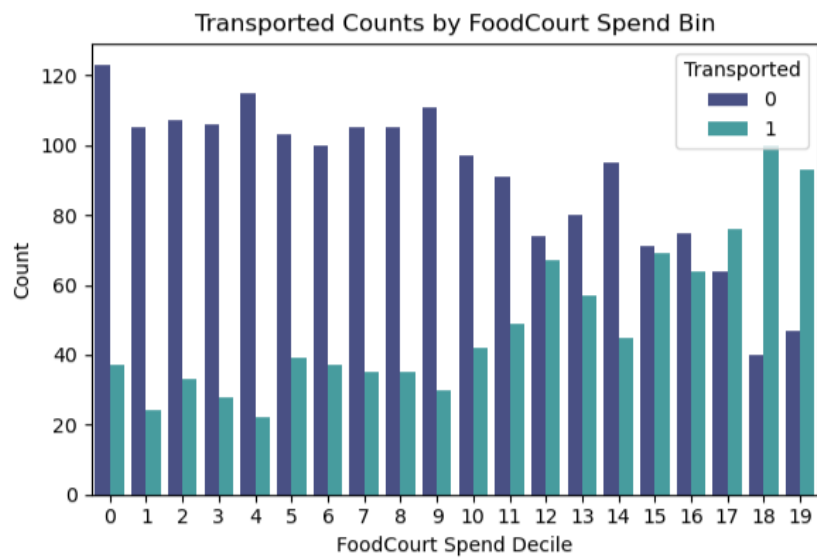
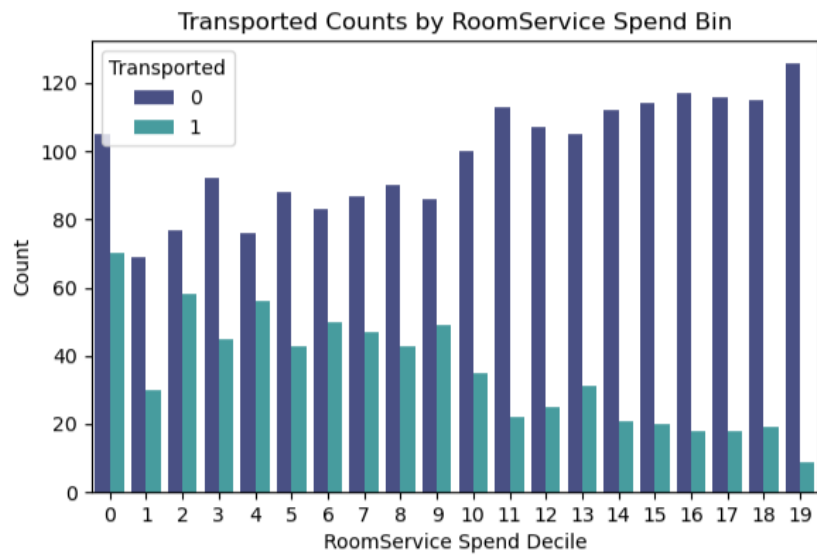
```
RoomService    0.080745
FoodCourt      0.044279
ShoppingMall   0.054525
Spa            0.070205
VRDeck         0.064943
All_Zero       0.118704
Total_Spend    0.129999
```

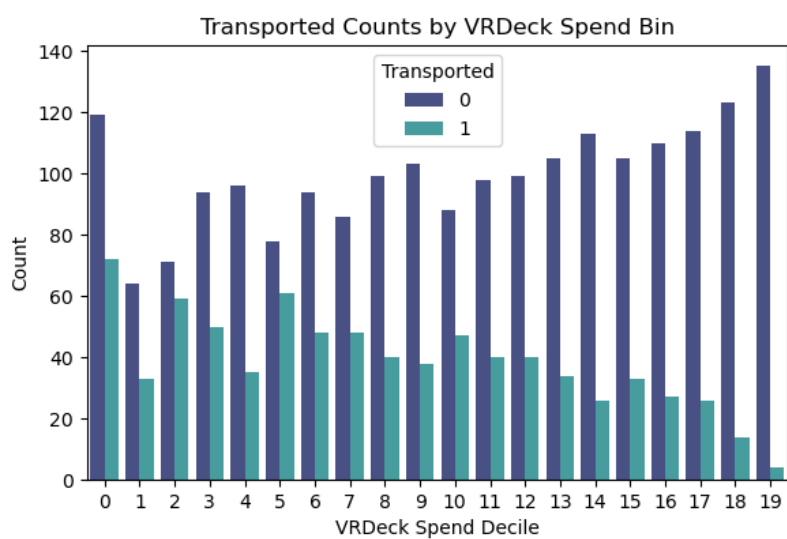
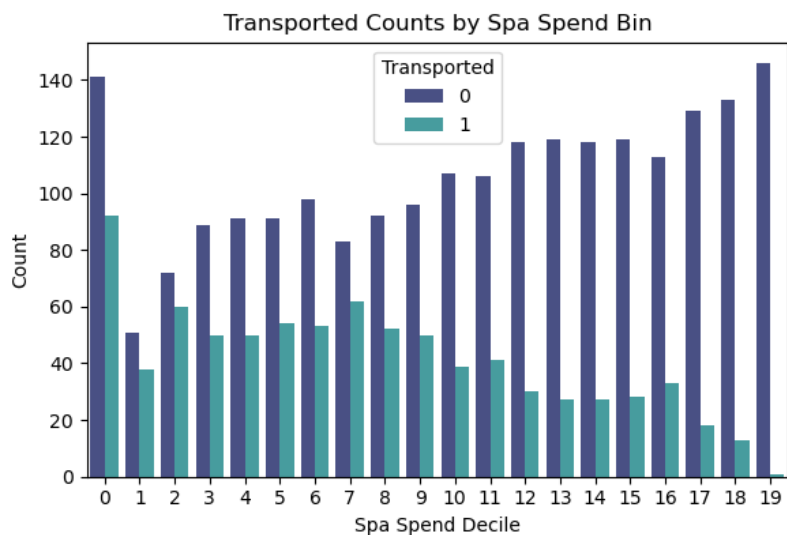
Visualiseerime huvipärast vaate, et kas kogukulutste summa suurus mõjutab oluliselt *Transported* tõenäosust:



Graafik näitab, et kogusumma suurus oluliselt tulemust ei mõjuta.

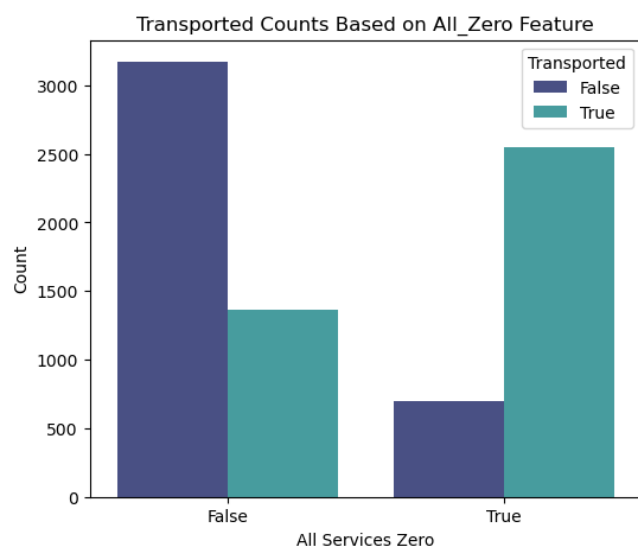
Visualiseerime nüüd aga kõigi kuluelementide seosed eraldi:





Nagu näha eraldiseisvana on kuluelementide summa suurusel mõju *Transported* tõenäosusele.

Lõpuks vaatleme veel *All_Zero* tunnust:



Graafikul on näha väga selge seos kahe atribuudi vahel.

Järgnevalt võtame vaatluse alla komposiitatribuudid.

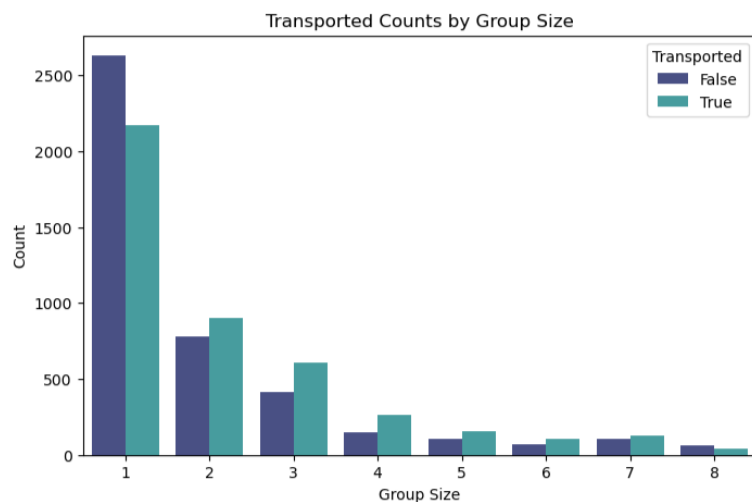
[Github repository: [spaceship-t EDA group analy.ipynb](#)]

PassengerId

Kuna tunnus koosneb kahest elemendist, reisija grupi koodist ja tema järjekorra numbrist grupis, siis sellina ta ilmselt lisaväärtust meile ei anna. Seega jagame atribuudi kaheks elemendiks: *Group* ja *PassengerNo*. Ka antud juhul meil oleks mõistlik proovida luua veel üks lisatunnus – meil on võimalik omistada igale reisijale grupi suurus kuhu ta kuulub *Group_size*. Meie andmearaam näeb välja järgnev:

	PassengerId	Transported	Group	PassengerNo	Group_size
0	0001_01	False	0001	01	1
1	0002_01	True	0002	01	1
2	0003_01	False	0003	01	2
3	0003_02	False	0003	02	2
4	0004_01	True	0004	01	1
5	0005_01	True	0005	01	1

Uurime kas grupi suurus annab meile mingisugust lisainfot:



Nagu näha kui üheliikmelises grupis on tõenäolisem, et *Transported = False*, siis suuremates gruppides on olukord vastupidine.

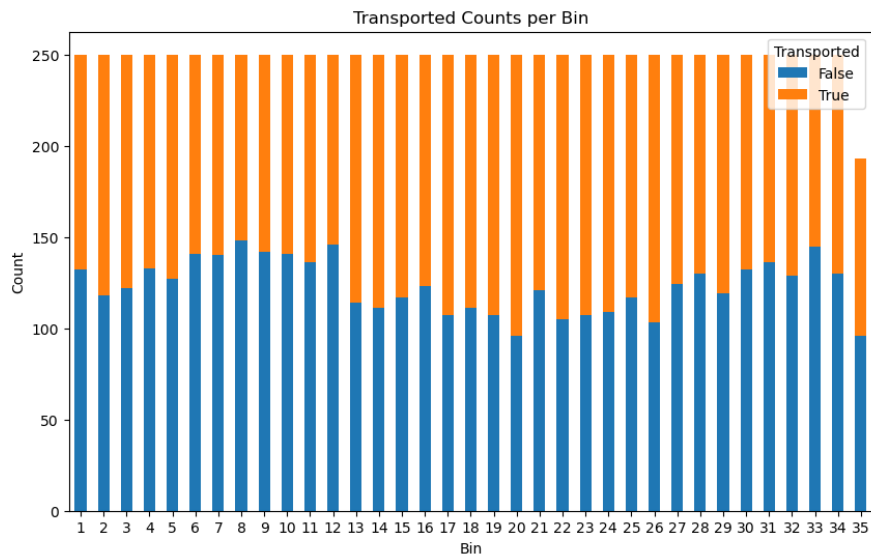
Arvutame ka korrelatsioonid:

	Transported	Group	Group_size	PassengerNo
Transported	1.000000	0.021491	0.082644	0.066390
Group	0.021491	1.000000	0.014753	0.011170
Group_size	0.082644	0.014753	1.000000	0.757107
PassengerNo	0.066390	0.011170	0.757107	1.000000

Mi skoor:

```
Group      0.512914
Group_size 0.017335
PassengerNo 0.004350
```

Viimase asjaoluna peaksime uurima, kas transporditaks muutmine sõltub kuidagi sellest, mitmendasse gruppi reisija kuulub:



Joonis viitab selgelt, et grupi järjekorra number ei oma seost reisija staatusega.

Cabin

[Github repository: [spaceship-t EDA Cabin analy.ipynb](#)]

Järgnevalt uurime atribuuti *Cabin*. Antud tunnus koosneb kolmest elemendist - kajuti tüüp; kajuti number; kajuti asukoht kosmoselaevas paremal või vasakul küljel. Seega peame atribuudi jagama eraldi tema kolmeks elemendiks: *deck*; *cab nr*; *side*. Meie andmeraam on järgnev:

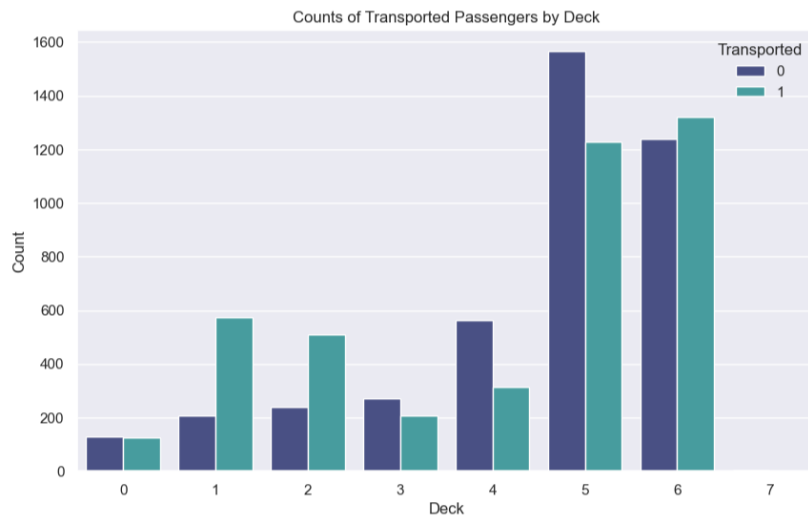
	Cabin	Transported	deck	cab nr	side
0	B/0/P	False	B	0	P
1	F/0/S	True	F	0	S
2	A/0/S	False	A	0	S
3	A/0/S	False	A	0	S
4	F/1/S	True	F	1	S

Arvutame tunnuste korrelatsioonid:

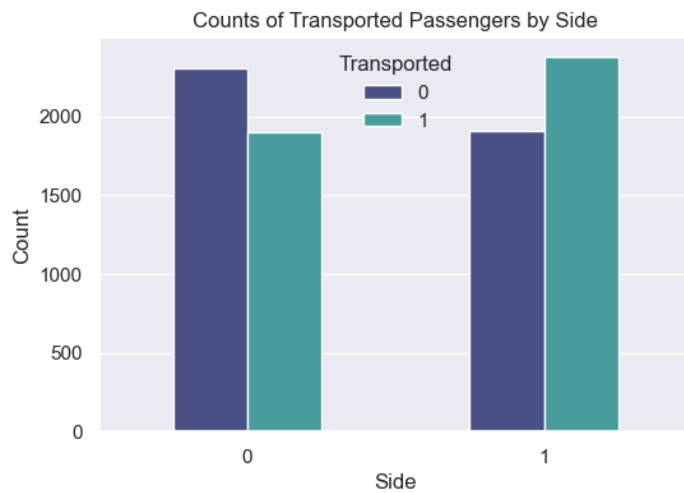
	Transported	cab nr	deck_encoded	side_encoded
Transported	1.000000	-0.045097	-0.109925	0.093319
cab nr	-0.045097	1.000000	0.531449	-0.037996
deck_encoded	-0.109925	0.531449	1.000000	0.093644
side_encoded	0.093319	-0.037996	0.093644	1.000000

Mi skoor:

deck	0.023700
cab nr	0.010968
side	0.005394

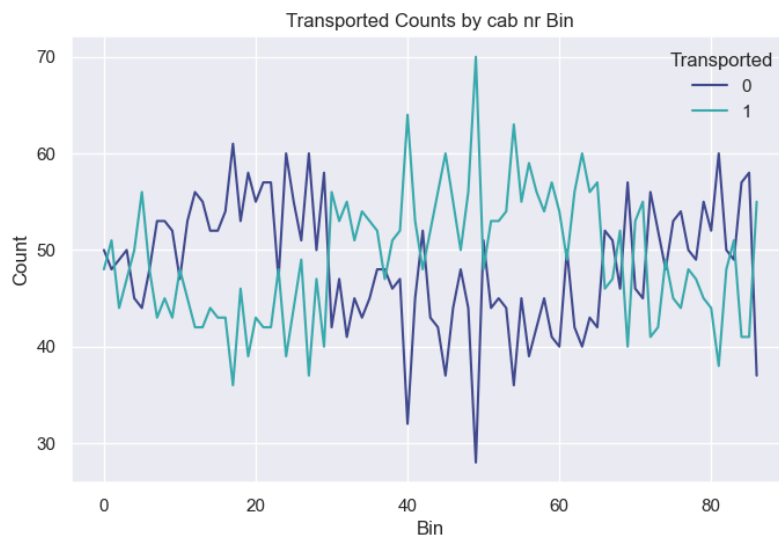


Graafik näitab, et reisija staatus sõltub mõningasel määral tema tekki numbrist.



Väikene mõju tundub olevat ka sellel, kummal poolel kajut asub. Kuid see sõltuvus on üsna nõrk.

Kontrollime veel ka seda, kas kajuti järjekorra numbril on mignisugune seos reisija staatusega:



Tundub, et selline seos on väga marginaalne.

Name

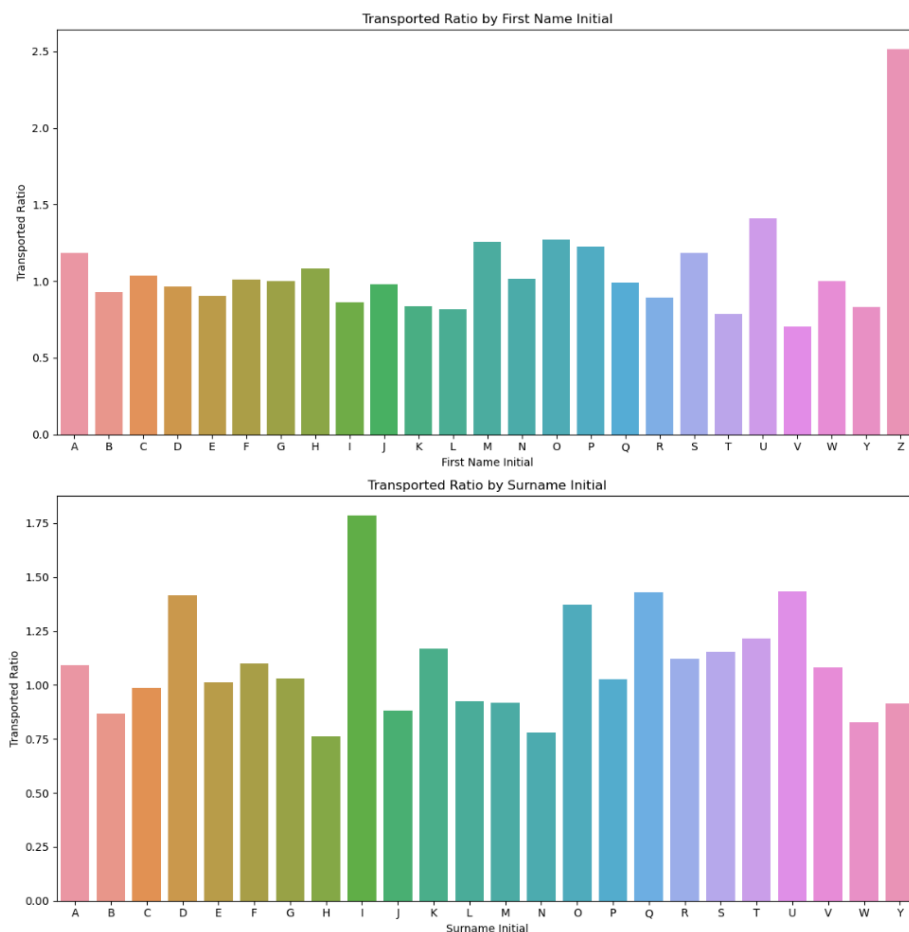
[Github repository: [spaceship-t EDA name analy.ipynb](#)]

Viimase atribuudina võtame suurimise alla reisija nime. Siinkohal mängib olulist rolli asjaolu, et meil on tegemist fantaasia ülesandega. Kogu alusandmestik on väljamõeldis, seega me ei tohi teha eeldusi mis võiksid kehtida meie tavamaailmas. Kui tavapäraselt samalaadsetes ülesannetes jäetakse kohe analüüsist kõrvale isikute nimed kuna nendes oletatavalt puudub igasugune lisandväärtus eesmärgi saavutamiseks, siis antud juhul me peame kindlasti kontrollima hüpoteesi, et reisija nime tunnustesse on kodeeritud mingisugune lisainfo.

Nimesid võiks põhimõtteliselt manipuleerida päris paljudeks erinevateks uuteks tunnusteks, kuid käesoleva projekti raames arvestades ajalisi piiranguid me piirdume mõningate lihtsate tunnustega: eraldame eraldi veergudesse reisija eesnime esimese tähe, perekonna nime esimese tähe, reisija initsiaalid ning nime pikkuse. Uued lisandunud atribuudid on: *FirstNameInitial*, *SurnameInitial*, *Initials*, *Name_length*. Meie andmeraam on järgnev:

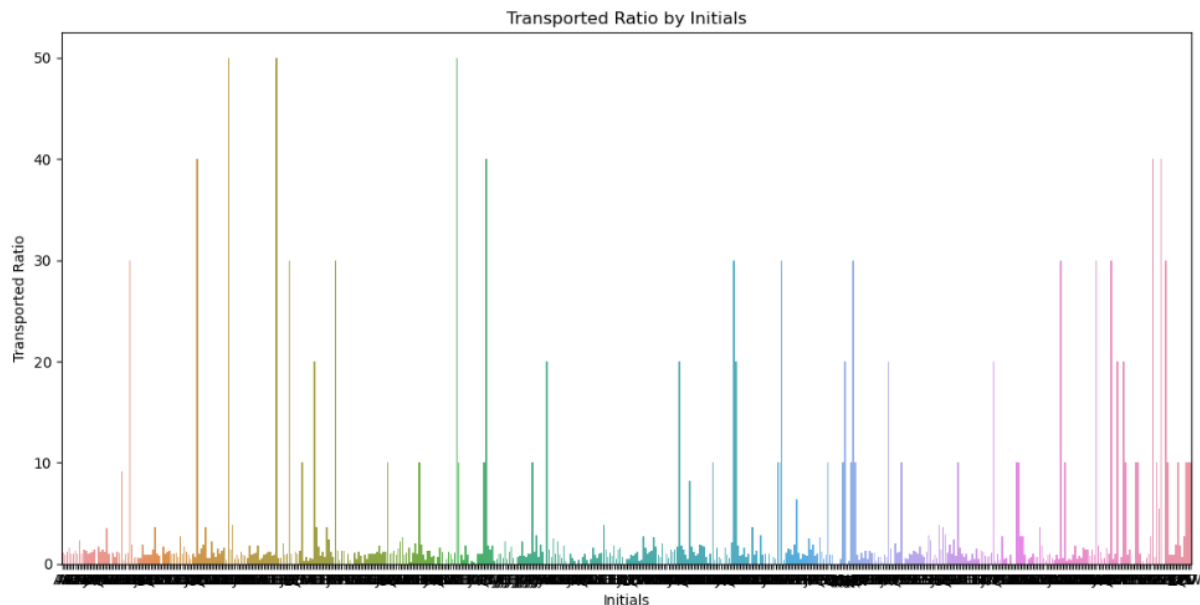
	Name	Transported	Name_length	FirstNameInitial	SurnameInitial	Initials
0	Maham Ofracculy	False	15	M	O	MO
1	Juanna Vines	True	12	J	V	JV
2	Altark Susent	False	13	A	S	AS
3	Solam Susent	False	12	S	S	SS
4	Willy Santantines	True	17	W	S	WS

Analüüsi lihtsustamiseks arvutame veel kõigi initsiaali versioonide kohta välja koefitsendi mitu reisijat transporditi iga mitte transporditud reisija kohta (*Transported = True / Transported = False*). Genereerime antud koefitsentide põhisel graafikud.



Joonistelt on näha et mõningased seosed reisija staatuse ja initsiaalide vahel esinevad, eriti perekonna nime puhul.

Graafik kõigi initsiaali paaride kohta on aga juba väga segane. Kuna kõiki paare esineb andmestikus üsna vähe, siis tõenäoliselt initsiaalide info kasutamine mudelis lisab suuresti ainult andmemüra. Seega jätame initsiaalide paarid mudelist välja.



Vaatame ka veel nime pikkuse korrelatsiooni *Transported* tunnusesse:

Correlation between 'Transported' and 'Name_length': 0.005056567966418165

Mutual Information score between 'Transported' and 'Name_length': 0.0009945863618998456

Nagu näha antud seos on väga nõrk.

Meetodid ja tulemused

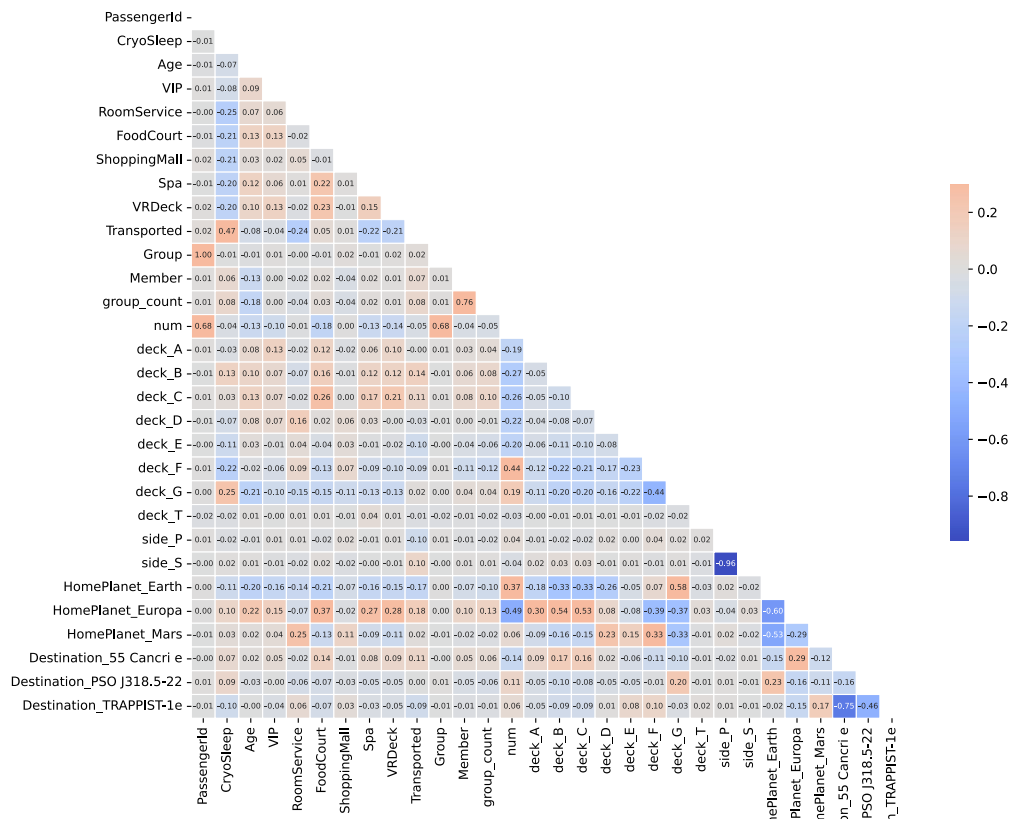
Andmete asendamine

Pärast andmete esmast analüüsi ning hindamist jõuame projekti faasi kus tuleb asuda puudvaid andmeid asendama. Nagu vaarasemalt juba tõesime siis puudvat infot esineb pea kõigi tunnuste osas ning mõjutatud on umbes veerand kõigist andmeridast.

Arvestades käesoleva projekti ühe eesmärgina erinevate kursusel läbitud meetodite praktiseerimist ning samuti üldist põhimõtet, et me ei peaks minema kergema vastupanu teed aktsepteerides nõrgemat andmekvaliteeti, kui mõõduka pingutuse kaudu on võimalik tõenäoliselt saavutada pädevam tulemus, siis jätame esialgu kõrvale väga lihtsustatud üldlevinud meetodid andmete asendamise jaoks. Me peaksime ülesandesse suhtuma tõsisemalt ning kõigi andmete asendust vajavate atribuutide puhul tuvastama ning rakendama targemaid ja loodetavasti täpsemaid andmeasenduse meetodikaid.

Vaatame kõik atribuudid järjest läbi ning rakendame asendused. Protsessi abivahendina genereerime kõigepealt kõigi atribuutide ülese korrelatsiooni maatriksi.

[Github repository: [spaceship-titanic_start.ipynb](#)]



Antud maatriks annab meile väga palju infot erinevate atribuutide omavaheliste seoste kohta ning võimaldab teha otsuseid milliseid tunnuseid oleks võimalik teiste olemasolevate tunnuste baasilt ennustada.

HomePlanet

[Github repository: [spaceship-tit_NaN_imputation.ipynb](#)]

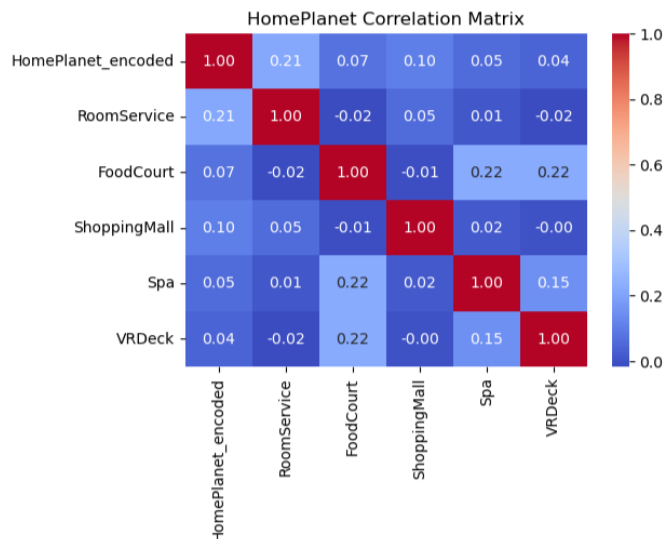
Alustame atribuudist *HomePlanet*. Arvestades üldist korrelatsiooni maatriksit me peaksime uurima lähemalt võimalusi kasutada ennustamisel *Cabin* tunnuse elemente ja teenuste kulude atribuute. Arvutame vastavad korrelatsioonid ja Mi skoorid.

	num	HomePlanet_encoded	deck_encoded	side_encoded
num	1.000000	-0.203172	0.533102	-0.034872
HomePlanet_encoded	-0.203172	1.000000	-0.418904	0.000528
deck_encoded	0.533102	-0.418904	1.000000	-0.033546
side_encoded	-0.034872	0.000528	-0.033546	1.000000

Mi skoorid:

```
deck_encoded    0.640803
side_encoded    0.000824
num             0.425031
```

```
RoomService    0.068046
FoodCourt      0.110602
ShoppingMall   0.053048
Spa            0.048386
VRDeck         0.075282
```



Ülal toodud andmete vaatlusel tundub, et kõige parema tulemuse *HomePlanet* ennustamisel võiks anda *deck* ja *num* atribuutide kasutamine. Seega liigume vajaliku mudeli treenimise juurde.

[Github repository: [spaceship-tit_HomePlanet_predict.ipynb](#)]

Proovime läbi DecisionTree ja RandomForest mudelid, siin juures treenime DecisionTree ennustama kahe ja ühe atribuudi põhisealt. Tulemused on järgnevad:

```
DecisionTree 'deck_encoded', 'num' model accuracy: 0.8
```

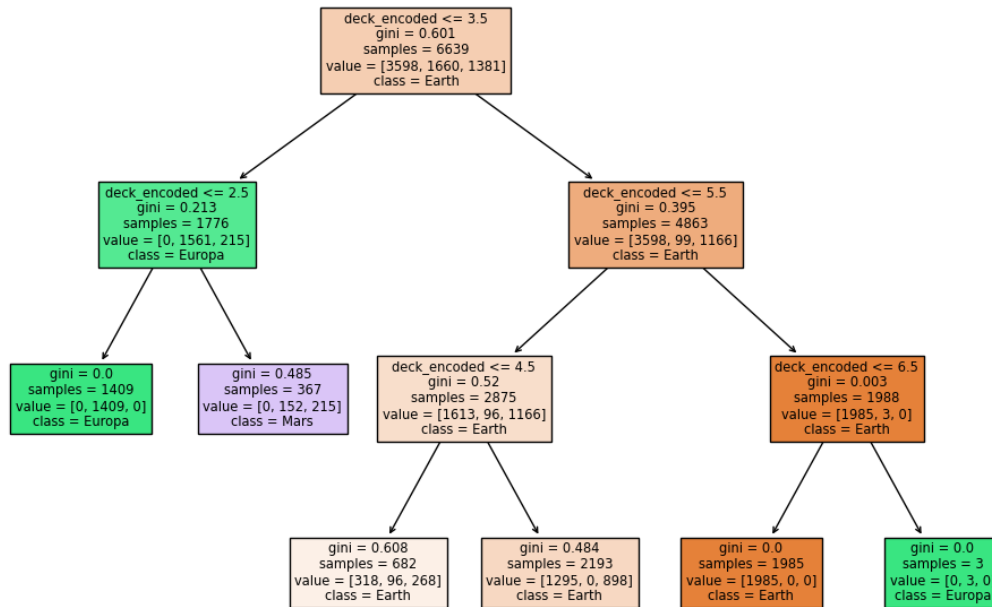
```
DecisionTree 'deck_encoded' model accuracy: 0.7951807228915663
```

```
RandomForest model accuracy: 0.8
```

```
{'deck_encoded': 0.5090750894078379, 'num': 0.4909249105921622}
```

Tulemustest on näha, et RandomForest ei ennusta paremini kui DecisionTree ning ühe atribuudi põhine ennustamine on praktiliselt samaväärne kahe atribuudi kasutamisega. Kuna ühe atribuudi põhine DecisionTree mudel on lihtne ning sobiv ka visualiseerimiseks, siis kasutame seda.

Lõplik mudel visualiseeritult:



Cabin

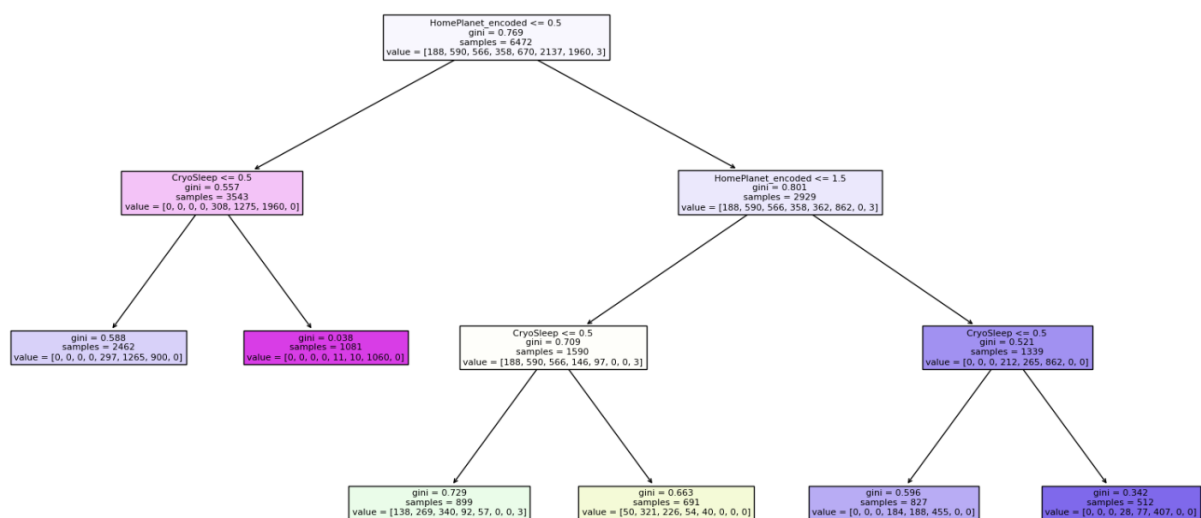
[Github repository: [spaceship-tit Cabin predict.ipynb](#)]

Järgnevalt tegeleme *Cabin* tunnuse asendamisega. Kuivõrd vastav tunnus koosneb kolmest erinevast elemendist, siis peame ka nende asendamisele lähenema kolmel erineval viisil. *Deck* elemendi ennustamiseks proovime kasutada atribuute *HomePlanet* ja *CryoSleep*. Treenime kaks mudelit mille tulemused on järgnevad:

DecisionTree 'HomePlanet_encoded' model Accuracy: 0.5243977764051884

DecisionTree 'HomePlanet_encoded', 'CryoSleep' model Accuracy: 0.5898702903026559

Kahte atribuuti kasutades me saame veidi parema täpsuse, seega läheme edasi selle mudeliga ning visualiseerime otsustuspuu:



Elemendi *num* ennustamiseks kasutame atribuuti *Group*. Treenime vastava mudeli:

```
DecisionTree 'Group' model Accuracy: 0.3143915997529339
```

Mudeli täpsus ei ole väga hea, kuid antud juhul meie eesmärk ei olegi väga täpselt ennustada *num* väärtust vaid saavutada tulemus kus väärtus on enam-vähem lähedane tõenäolisele kajuti tegelikule numbrile.

Viimase sammuna asendame *side* elementi väärtused. Kuna antud elemendil puudub märgatav korrelatsioon ühegi teise atribuudiga, siis läheneme *side* väärtuste asendamisele genereerides need juhusliku valikuna kahe võimaliku väärtuse hulgast.

Lõpuks kompileerime *Cabin* tunnuse andmeraami jaoks tema algsel kujul.

CryoSleep

[Github repository: [spaceship-tit CryoSleep impute.ipynb](#)]

CryoSleep tunnus üldise korrelatsiooni maatriki alusel on üks olulisemaid atribuute meie lõpliku ennustusmudeli teermise jaoks seega peame püüdma antud veerus andmeid maksimaalselt adekvaatselt asendada.

CryoSleep puhul esmalt me saame lähtuda loogilisest reeglist: kui reisija on tarbinud kasvõi ühte teenust, siis järelikult peab tema puhul kehtima *CryoSleep = False*.

Teiseks on võimalik tuvastada, et juhul kui reisija asus tekkidel A,D,E või F, siis suure tõenäosusega taas kehtib *CryoSleep = False*.

Kuna kõik ülejäänud *CryoSleep* asendust vajavad reisijad ei tarbinud ühtegi teenust, siis asendame viimased puuduvad väärtused *CryoSleep = True*.

Name

[Github repository: [spaceship-tit Name impute.ipynb](#)]

Jätkame *Name* atribuudiga. Kuna selgeid korrelatsioone ei esine, siis läheneme lihtsamate meetoditega.

Kasutame *Name_length* atribuudi asendamiseks ülejäänud väärtuste mediaani.

FirstNameInitial ja *SurnameInitial* väärtuste asendamisel genereerime puuduvad väärtused taas juhusliku valiku teel teiste vastavas veerus leiduvate väärtuste hulgast.

Ülejäänud tunnuste asendamisel kasutame väga selgete korrelatsioonide puudumise tõttu tavapäraseid andmete asendamise meetodeid.

[Github repository: [spaceship-tit other imputations.ipynb](#)]

Destination atribuudi asendused teeme *mode*-meetodiga.

Age puhul kasutame asenduseks kõigi reisijate vanuste mediaani

VIP atribuudil kasutame taas *mode*-meetodit

RoomService, *FoodCourt*, *ShoppingMall*, *Spa*, *VRDeck* kulutuste atribuutide puhul ehitame veidi keerukama asenduse loogika. Juhul kui ühel nendest tunnustest väärtus puudub, aga kõigi ülejäänute väärtused on null, siis statistiliselt on suure tõenäosusega ka puuduv väärtus null, selliselt ta ka asendame.

Ülejäänud puuduvad väärtused asendame vastava veeru keskmiste väärtustega.

Viimase sammuna kompileerime kõik varasemate sammude jooksul asendatud väärtused üheks täielikuks andmeraamiks.

[Github repository: [spaceship-tit_compile_final_imputations.ipynb](#)]

Mudelite treenimine

Varasema andemanalüüsi protsessi käigus me tuvastasime kõigi võimalike algsete ja lisaks loodud atribuutide korrelatsiooni ja seosed *Transported* atribuudi suhtes. Selle informatsiooni baasilt saame kokku koondada eeldatava atribuutide olulisuse pingerea, kui hästi võiks üks või teine tunnus mudeli ennustamise võimekust parendada.

Lõpliku pingerea koostamisel koondasin kokku kõigi atribuutide kohta arvutatud korrelatsioonid ning Mi skoorid *Transported* suhtes ning summeerisin need. Koostasin kolm erinevat pingerida: korrelatsiooni põhiselt, Mi skoori põhiselt ning summeeritud väärtuste põhiselt. Lõpliku pingerea saavutamiseks liitsin kõik kolm ränkingut kokku ning kõige madalama tulemuse saavutanud atribuudid peaksid olema masinõppe mudelitele kõige paremaks sisendiks:

	total
All_Zero	5
CryoSleep	8
RoomService	13
Spa	15
VRDeck	17
Group	20
Total_Spend	22
HomePlanet	25
deck	30
Group_size	32
Destination	33
FoodCourt	34
ShoppingMall	38
Age	40
num	47
VIP	51
FirstNameInitial	51
SurnameInitial	54
Name_length	57

Arvestades toodud mudelite treenimisel potentsiaalselt kasutatavate atribuutide nimekirja teostame veel viimased modifikatsioonid ning ebavajalike andmete eemaldamised meie treenimiseks kasutatava andmeraami sisus.

[Github repository: [spaceship-tit_feature_engineer.ipynb](#)]

Esimeseks treenimise faasiks valisin välja nimekirja laialt kasutatavatest masinõppe mudelitest mida treenisin kogu atribuutide andmestiku peal. Tulemused olid järgnevad:

[Github repository: [spaceship-tit_model_training.ipynb](#)]

```
LogisticRegression accuracy: 0.7832087406555491
precision    recall  f1-score   support

False        0.80      0.74      0.77      861
True         0.77      0.82      0.79      878
```

```
DecisionTree accuracy: 0.7515813686026452
precision    recall  f1-score   support

False        0.75      0.76      0.75      861
True         0.76      0.75      0.75      878
```

```
RandomForest accuracy: 0.7935595169637722
precision    recall  f1-score   support

False        0.78      0.82      0.80      861
True         0.81      0.77      0.79      878
```

```
Gradient Boosting Machine accuracy: 0.7924094307073031
precision    recall  f1-score   support

False        0.81      0.75      0.78      861
True         0.78      0.83      0.80      878
```

```
Support Vector Machine accuracy: 0.7872340425531915
precision    recall  f1-score   support

False        0.79      0.77      0.78      861
True         0.78      0.80      0.79      878
```

```
K-Nearest Neighbors accuracy: 0.7676825761932144
precision    recall  f1-score   support

False        0.75      0.79      0.77      861
True         0.78      0.74      0.76      878
```

```
Gaussian Naive Bayes accuracy: 0.7590569292696953
precision    recall  f1-score   support

False        0.77      0.73      0.75      861
True         0.75      0.78      0.77      878
```

```
Feedforward Neural Networks accuracy: 0.7860839366912842
```

Esimese proovi kohta on tulemused päris head kui võrrelda Kaggle Spaceship Titanic võistluses esitatud mudelite keskmist täpsust. Üldiselt kõik mudelid annavad üsna lähedased tulemused, kuid kõige kõrgema täpsuseni jõuavad Random Forest ja Gradient Boosting Machines (GBM) mudelid.

Pärast erladiseivate baasmudelite treenimist proovisin läbi ka mudelite ansambli loomise kasutades *Stacked Generalization* meetodit. Antud juhul valisin baasmudeliteks eraldiseisvana kolm parimat tulemust näidanud mudelit: Random Forest, Gradient Boosting Machine ja Support Vector Machine.

Stacking mudeli treenimise tulemusena paranes tulemuste täpsus ainult väga marginaalselt:

Stacked Model	Accuracy: 0.7987349051178838			
	precision	recall	f1-score	support
False	0.81	0.78	0.79	861
True	0.79	0.82	0.80	878

Mudeli parendamine

Mudelite parendamise eesmärgiga võtsin esmalt vaatluse alla treenimisel kasutatavad erinevad tunnuste kombinatsioonid. Arendamiseks valisin Random Forest mudeli kuna siit saime ka esimeses treenimise faasis kõige parema tulemuse. Kirjutasin koodi mis genereerib 18 tunnuse kogumi alusel kõik võimalikud tunnuste kombinatsioonid kasutades korraga 16 tunnust (kokku 153 kombinatsiooni). Kood treenib mudelit kõigi võimalike kombinatsioonidega eesmärgiga leida kõige kõrgema potentsiaaliga kombinatsioonid.

[Github repository: [spaceship-tit_model_featuretest.ipynb](#)]

Parima 20 Random Forest poolt tuvastatud tunnuste kombinatsiooni baasilt proovisin treenida omakorda uuesti DecisionTree, GradientBoostingClassifier ja Support Vector Machines mudeleid, kuid paraku selline lähenemine nende ennustusvõimekust ei parandanud.


Järgnevalt teostas Random Forest mudeli *Hyperparameter Tuning* protsessi. Ka antud protseduur ei andnud mudelile oluliselt paremat täpsust.

[Github repository: [spaceship-tit_RandomForest_improve.ipynb](#)]

Kaggle keskkonnas esitasin test-andmestiku pealt teostatud ennustused kasutades esialgselt parimat tulemust näidanud Random Forest ning Stacking ansambli mudeleid. Hetkel parima tulemuse saavutas järgnev esitus, mis paikneb Spaceship Titanic võistluse pingereas üsna täpselt keskel.

1290


Erki Joekalda



0.79191

2

20s



Your Best Entry!

Your most recent submission scored 0.79191, which is an improvement of your previous score of 0.78723. Great job!

[Tweet this](#)

Arvestades mudelite parendamiseks jäänud piiratud ajalist ressursi siis tulemust võib hinnata rahuldavaks.

Kaggle keskkonda esitatud tulemuste aluseks olevad treenitud mudelid on salvestatud kasutades pythoni moodulit *pickle*.

trained_RandomForest.pkl

trained_Stacked.pkl

Järeldused ja arutelu

Nagu sissejuhatuses märgitud sai seadsin projekti peamiseks eesmärgiks püüda töö koostamisel olulises mahus praktiseerida kursuse käigus läbitud meetodeid ja mudeleid. Kuna alusandmestikuna on kasutusel täielikult välja mõeldud juhtum, siis mingisuguste populaarteaduslikult huvipakkuvate lahendusteni ma antud juhul jõuda ei saakski.

Arvestades ootusi ja situatsioonilist taustsüsteemi ma arvan projekt täitis täiel määral oma eesmärgi. Kuivõrd ette antud andmed olid tundmatud ning ei olnud võimalik teha ka mingisuguseid reaalmaailmas vaikimisi prevaleerivaid eeldusi, siis panustasin olulises mahus aega ja ressursi andmete ja nende vaheliste seoste uurimisele. Samuti proovisin läbi mitmete tuletatud atribuutide loomise protsessi ning erineva keerukusastmega lähenemised puuduvate andmete asendamiseks. Ülesanne ja sellega seotud andmestik annaks ruumi veel väga olulisel määral analüüsi laiendada ja süvendada, kuid projekti jaoks allokeeritav ajaline ressurss paraku seab siin piirid.

Ühe huvipakkuva teemana millesse peaks veel rohkem sisse vaatama on adekvaatse balansi leidmine puuduva andmestiku asendamise vajaduse ning selle asendamise protsessi panustatava ressursi vahel. Ehk kuidas määratleda optimum kas teatud andmete asendamisel on mõistlikum läheneda lihtsustatud tavameetoditega või tuleks püüda asendamisel samuti kasutada keerulisemaid ennustavaid meetodikaid, kui see on teoreetiliselt tehniliselt võimalik. Siin saaks olemasoleva projekti andmestiku erinevate versioonide põhisel analüüsi jätkata ning see oleks järeltegevustena ka plaanis.

Huvitavaks tulemuseks enda vaates peaksin seda, et kõik treenimisel kasutatavad erinevad masinõppe mudelid genereerisid üllatavalt sarnase täpsusega tulemusi. Intuiitiivselt oleksin eeldanud, et hajuvus mudelite täpsuses oleks suurem ning teatud tüüpi andmete peal võiks mõned mudelid toimida selgemalt paremini kui teised. Antud vaatenurgast näen kindlasti arengu kohta enda jaoks saavutamaks paremat tunnetust millistele mudelitele keskenduda olenevalt andmete struktuurist.

Kindlasti loeksin enda jaoks antud kursuse teeneks olulise fookuse ChatGPT ja võibolla ka teiste AI põhiste laiemalt kättesaadavate teenuste ja lahenduste kasutamisele. Varasemalt ma ei olnud seni antud lahendusi kuigi laialdaselt kasutanud, kuid nüüd näen tõesti üsna selgelt kuidas AI rekenduste areng teatud tegevusvaldkondi väga tugevalt muutma võiks hakata.

Kokkuvõtlikult nii kursus kui töö projekti kallal on andnud hulgaliselt infot ja mõõdukalt praktilist kogemust suunal kuidas andmeteaduse meetodikaid realselt rakendada ettevõtte igapäevaste andmeanalüütika ja andmekvaliteedi probleemaatikate adresseerimisel. Siit on võimalik minna edasi mõtete ja visioonide genereerimisel otsimaks võimalusi uuel tasemel protsesside disainimiseks ning leidmaks võimalusi integratsioonideks olemasolevatesse tehnilise infrastruktuuri elementidesse ja andmevoogudesse.