CHURN ANALYSIS OF GITTIGIDIYOR CUSTOMERS

Using Machine Learning Techniques

Özlem Hazal KANTARCI
311602023

İSTANBUL, 2017
MEF UNIVERSITY
MSc Big Data Analytics

Executive Summary

Churn Analysis of GittiGidiyor Customers

Özlem Hazal Kantarcı
Advisor: Dr. Tuna Çakar

SEPTEMBER, 2017, 41 pages

In this project, it is aimed to estimate the loyalty of the customers of the e-commerce company named GittiGidiyor by analyzing the customer movements and examined which movements affected the customer loyalty positively / negatively

In the dataset studied, it was seen that the number of active customers is much higher than that of passive customers. Several methods have been tried to solve this "Class imbalance" problem and it has been decided to replicate some lines of passive customers. Rows of smaller classes are duplicated to compensate classes with generated code.

The data set was divided into training, validation and test and different algorithms were used. One of the innovative approaches was training and validating models in an earlier time window and testing the model with samples from a later time window. As a result of the studies, it was decided to use "Linear Discriminant Analysis" considering its short training time and especially the success of predicting passive customers.

Key Words: churn analysis, class imbalance problem, feature selection, pivot table, correlation, process time, accuracy, classifier, boosting, linear discriminant analysis, decision tree

Özet

GittiGidiyor Müşterileri için Bağlılık Analizi
Özlem Hazal Kantarcı
Tez Danışmanı: Dr. Tuna Çakar
EYLÜL, 2017, 41 sayfa

GittiGidiyor e-ticaret şirketinin müşterileri için bağlılık tahmini yapılması hedeflenen bu projede müşteri hareketlerinin analizi yapılarak müşteri hareketlerinden müşteri bağlılığı tahmin edilmeye çalışıldı.

Üzerinde çalışılan veri setinde, aktif müşteri sayısının pasif müşterilerden çok daha fazla olduğu görüldü. Bu “Sınıf Dengesizliği” sorununu çözmek için bir çok yöntem denendi ve pasif müşterilerin bazı satırlarının çoğaltılmasına karar verildi. Geliştirilen kod ile sınıfları dengelemek için daha küçük sınıfın satırlarının çoğaltılması sağlandı.

Veri seti eğitim, doğrulama ve test olarak üçe ayırarak farklı algoritmalar ile çalışıldı. Uygulanan inovatif yaklaşımlardan birisi eğitim ve doğrulamayı daha önceki bir zaman diliminde yapılan işlermler ile yapılması, testin daha sonraki bir zaman diliminde yapılan işlemler ile yapılması oldu. Çalışmalar sonucunda hem eğitim süresinin kısa olması hem de özellikle pasif müşterileri tahmin etme başarısı göz önünde bulundurularak “Linear Discriminant Analysis” kullanılmaya karar verildi.

Anahtar Kelimeler: bağlılık analizi, sınıf dengesizliği, özellik seçimi, pivot tablo,çalışma süresi, başarı yüzdesi,korelasyon,sınıflandırıcı,karar ağacı, doğrusal diskriminant analizi

Table of Contents

Introduction

GittiGidiyor is one of the biggest e-commerce companies of Turkey. It aims to increase customer satisfaction and customer loyality using Big Data Analytics and Machine Learning. If customers who are likely to leave can be predicted in advance it can be possible to take counter measures to prevent them leave.

“Churned customer” is defined by Gittigidiyor as “a customer who has not made any transactions in the last 365 days. The aim of this project is predicting the customers that will churn in the next x months.

To build a predictive model of churn, historical data of customer transactions was analyzed. GittiGidiyor provided a table of transactions from which the following information could be extracted:

- length of the business relationship
- frequency of customer transactions
- types of products purchased
- average purchase sizes
- bad experience frequency

Customer demographics was not provided by Gittigidiyor. Accuracy of this study can be further improved using attributes such as:

- age
- gender
- segment
- job type
- geographic location

In some real world applications, various problems related to the data can reduce the power of machine learning algorithms. The problem of "Class imbalance" encountered in this project is one of them and a common one." Class imbalance" occurs when a class is represented by many individuals while the other one is represented by only a few people, which may cause some problems. Such a situation can cause problems. Several techniques have been proposed in the literature. One of these techniques was used in this project

For the initial investigation of data, a proprietary data mining software RapidMiner and Microsoft Excel were used. Later, all the coding was done in Python on Jupyter Notebooks using the libraries Pandas, numpy and sckit-learn. Plots were created using graphviz and mplot3d libraries.

Exploring the Dataset

GittiGidiyor gave 2 files containing customer transactions called SMF0.csv and SMF1.csv.

SMF0 includes all the transactions whether they are completed (paid, delivered and approved) or not. SMF1 includes only the approved transactions. Thus, approved transactions exist in both files.

If customers are not satisfied, they can start return/cancellation process. The price of these products are not passed to the sellers and these transactions reside only in SMF0.

If customers don't start return/cancellation process within 15 days after delivery, these transactions are automatically approved and copied into SMF1. It's logical that some transactions will exist in SMF0 but not SMF1.

It may be expected that all transactions those exist in SMF1 will also exist in SMF0. However, since our data is from a limited time window (April 1st 2015 – April 30th 2017), some transactions that were approved between April 1st 2015 and April 15th 2015 may be created before April 1st 2015 and, thus, not exist in SMF0.

In addition, some rare artifacts in data were discovered. Some transaction from mid 2016 exist only in SMF1. This may be caused by losing some lines in the database or problems during exporting the data. (Sample member id: 24165)

In order to have a consistent data, only SMF0 table was used in this study.

In addition to transaction data, some visit data was given by GittiGidiyor. It consisted the number of visits of each customer on each day of the week and on some special days like Valentine's day, Mother's day etc. These numbers were the sum of visits in the last 12 months. Since the dates needed to be parametrized, this data was not used at all.

Exploratory Data Analysis

SMF0 has 125377 transactions. As of April 30th 2017, 82.3% of these transactions belong to churned customers, while 17.7% of these transactions belong to active customers.

Status	Count	Fraction
Active	103229	0.823
Passive	22148	0.177

Table 1: Distribution of transactions as of 30.04.2017

The transactions are between April 1st 2015 and April 30th 2017.

SMF Type	Earliest Date	Latest Date
SMF0	April 1,2015	April 30,2017

Table 2 : Transaction date range

As seen in Figure 1, number of transactions increase dramatically after July 1st 2016.

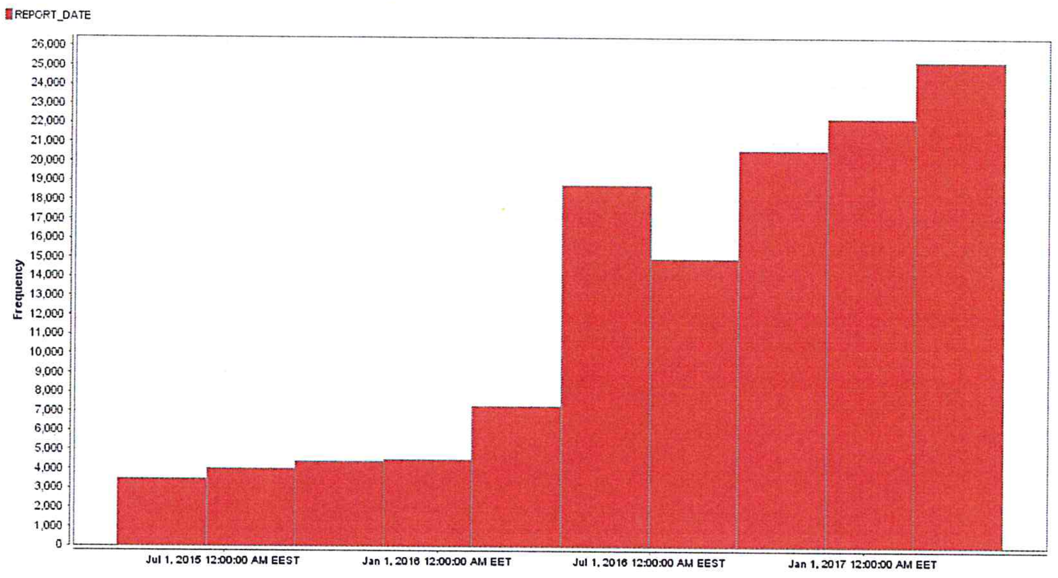
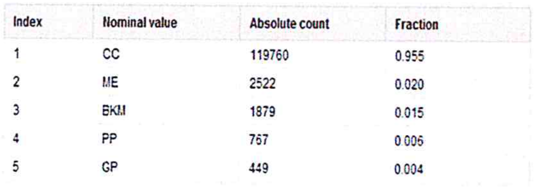


Figure 1 : Transaction date distribution

In Table 3, it's seen that 96% of all purchases are made using credit cards. Since 96% is a quite high ratio, it does not give a discriminative information, and it's not used in the model.



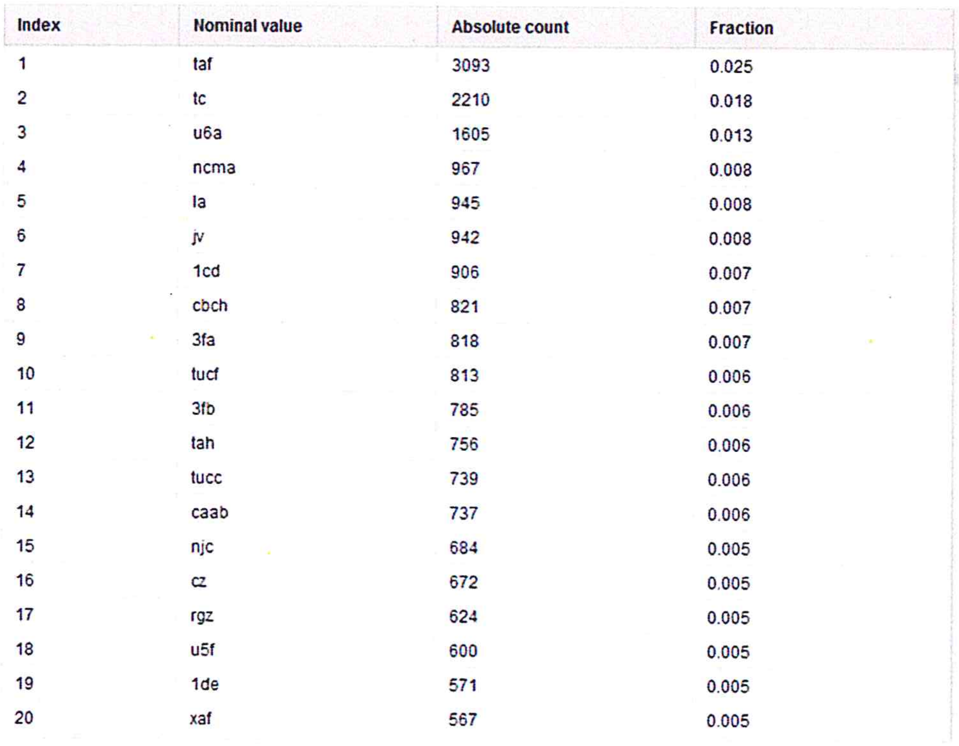
Index	Nominal value	Absolute count	Fraction
1	CC	119760	0.955
2	ME	2522	0.020
3	BKM	1879	0.015
4	PP	767	0.006
5	GP	449	0.004

Table 3 : SMF0 payment types count

The categories of transactions are given in the dataset. Every letter/figure stands for a category and they are stacked hierarchically. The first letter/figure is the top category.

Table 4 shows the most common 20 categories and Figure 1 shows this distribution graphically.

Only the top hierarchical category is used in the model.



Index	Nominal value	Absolute count	Fraction
1	taf	3093	0.025
2	tc	2210	0.018
3	u6a	1605	0.013
4	ncma	967	0.008
5	la	945	0.008
6	jv	942	0.008
7	1cd	906	0.007
8	cbch	821	0.007
9	3fa	818	0.007
10	tucf	813	0.006
11	3fb	785	0.006
12	tah	756	0.006
13	tucc	739	0.006
14	caab	737	0.006
15	njc	684	0.005
16	cz	672	0.005
17	rgz	624	0.005
18	u5f	600	0.005
19	1de	571	0.005
20	xaf	567	0.005

Table 4 : The most common 20 categories

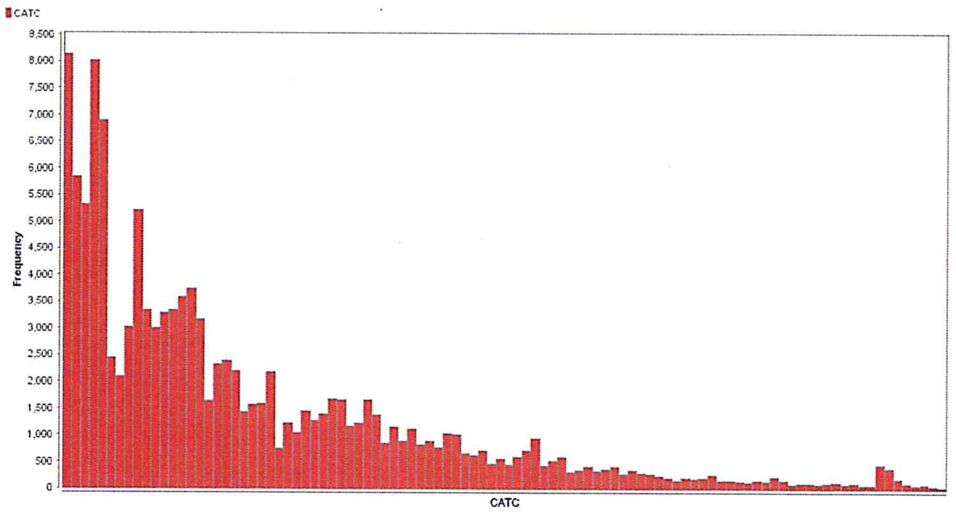


Figure 2 : Category distribution in SMF0

Figure 3 shows the distribution of the installment count, which is between 1 and 12 with an average of 2.87 as shown in Table 5.

SMF Type	Min Installment	Average Installment	Max Installment
SMF0	1	2.871	12

Table 5 : Installment count

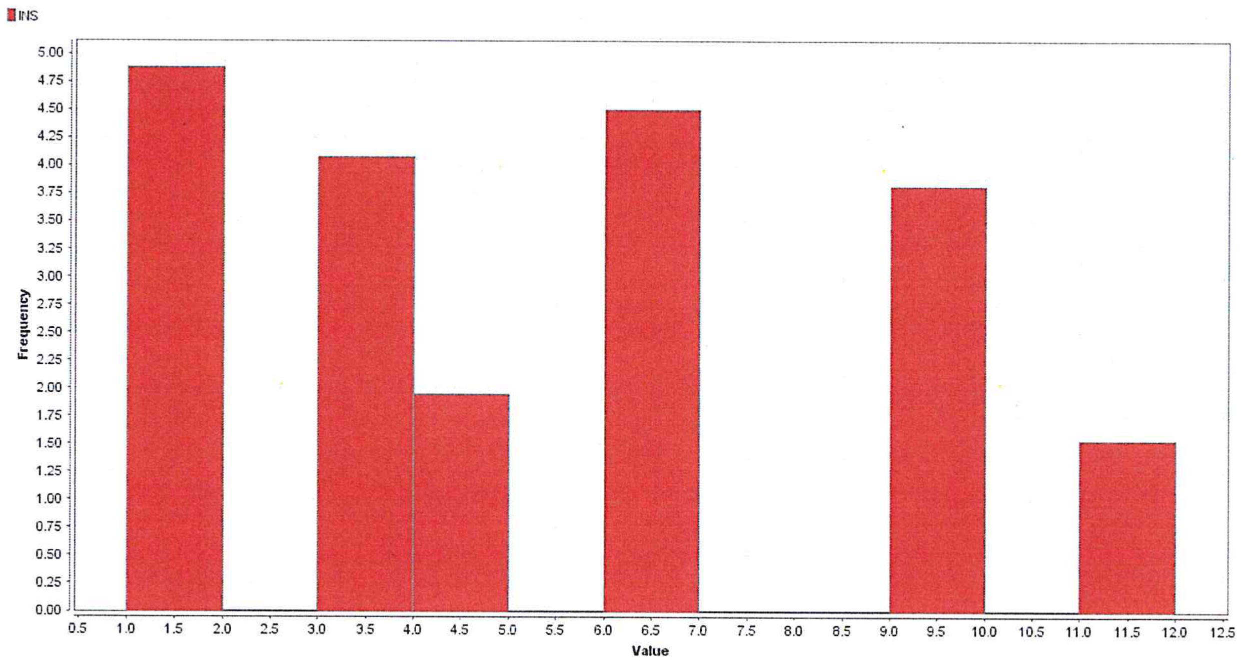


Figure 3 : Installment distribution

Amount of purchases gives an important information about churn probability of a customer. **Hata! Başvuru kaynağı bulunamadı.** gives the statistical values of purchase amounts and Figure 4 shows the distribution graphically. Amount is one of the most prominent features that will be used in the model.

SMF Type	Min	Average	Max.	Deviation
SMF0	0.020	246.093	51999.600	685.211

Table 6: Amount statistical values

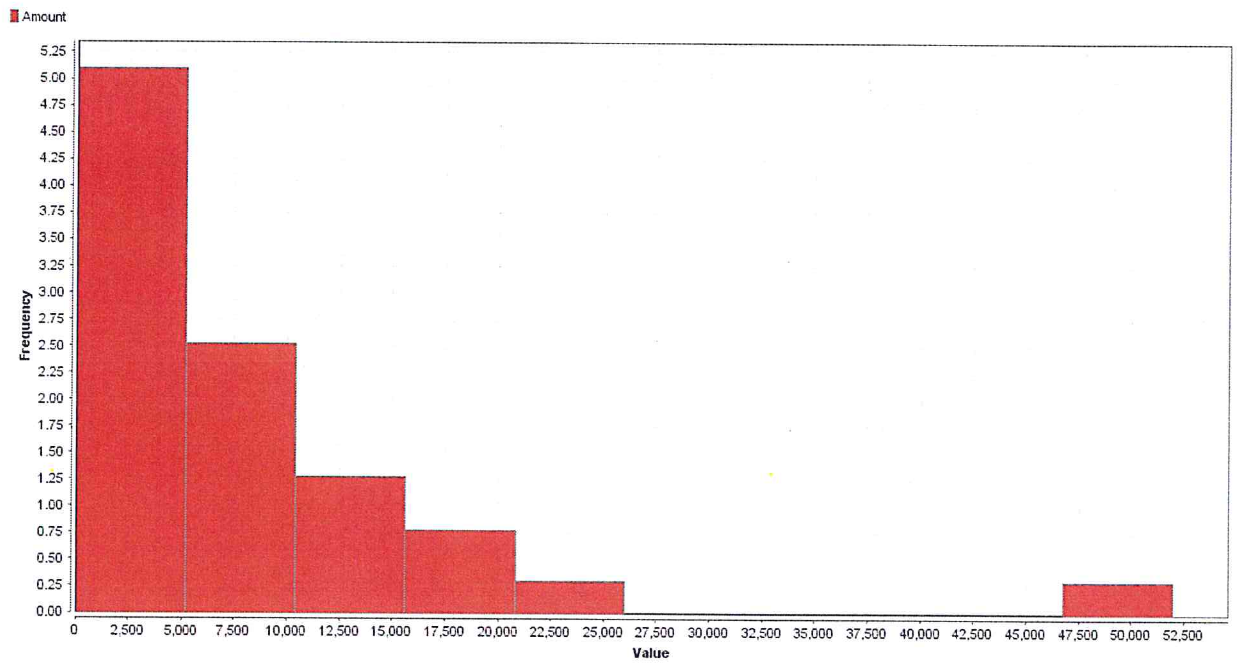


Figure 4: Amount distribution

Data Preprocessing

Transforming the Data Frame

SMF0 is transaction based. The aim of this study is predicting the customers that will churn, thus we need to convert the data to member based. Pandas's pivot table will be used for this transformation.

Every customer may have multiple rows. What is desired is having 1 single row for each customer and still keep the information. In this section, it will be investigated how to collect the information like shopping dates, bad experiences, categories, product IDs together in a single row.

The data is from a range of time. The trained model will be tested for different time periods. So, it's required to handle dates easily. In the code, it's enough to set TODAY and DAYS_TO_PREDICT variables. The remaining of the code assumes that the date of execution is TODAY and it tries to predict the statuses of active customers DAYS_TO_PREDICT days later.

For training, TODAY was set to 2016-04-28. The model was trained to predict 60 days later and the trained model was tested after setting TODAY to 2017-02-28.

There are 2 separate code files for training and testing. Training code saves the trained model to the disk, testing code reads the trained model from the disk and tests it for a different time period.

Pivot tables were created starting from dates. Minimum and maximum of transaction dates for each customer was taken and these were called “LAST PURCHASE” and “FIRST_PURCHASE” in the new data frame.

```
# first and last purchase dates of each member
piv2 = df.pivot_table(index=["member_id"], values=["DATE"], aggfunc=[max, min])
piv2.fillna(0, inplace=True)
piv2.reset_index( inplace=True)
piv2.columns = ['member_id', 'LAST_PURCHASE', 'FIRST_PURCHASE']
```

Listing 1: Finding first and last purchase of customers

GittiGidiyor defines churn of a customer as not having a transaction in the last 12 months. The status of each customer DAYS_TO_PREDICT days after TODAY was found and added into the new data frame with a column name “FUTURE_STATUS”. FUTURE_STATUS is set to 1 if the customer is still active on the prediction date, and to 0 if he churns.

```
PREDICTION_DATE = TODAY + timedelta(days=DAYS_TO_PREDICT)
ONE_YEAR_BEFORE_PREDICTION = PREDICTION_DATE - timedelta(days=365)


statuses['FUTURE_STATUS'] = np.where(statuses['LAST_PURCHASE'] >
pd.to_datetime(ONE_YEAR_BEFORE_PREDICTION), 1, 0 )
```

Listing 2: Detecting statuses of customers on prediction date

Every PAYMENT_ID stands for a separate shopping cart. Number of unique carts for each customer was found and put into the new column called “NUMBER_CARTS” with the code in Listing 3.

```
carts = pd.DataFrame()
carts = df.groupby("member_id").PAYMENT_ID.nunique().to_frame().reset_index()
carts.columns = ['member_id', 'NUMBER_CARTS']
```

Listing 3: Finding number of carts for each customer

Number of items each customer purchased is shown in column named “NUMBER_ITEMS”.

```
# how many items did each member buy?
piv1 = df.pivot_table(index=["member_id"], values=["Amount"],
aggfunc=[np.sum,np.mean,len])
piv1.fillna(0, inplace=True)
piv1.reset_index(inplace=True)
piv1.columns = ['member_id', 'TOTAL_AMOUNT', 'AVERAGE_AMOUNT_PER_ITEM',
'NUMBER_ITEMS']
```

Listing 4: Finding how many items each member purchased

The number of unique sellers each customer purchased from is stored in column called "NUMBER_SELLERS".

```
# number of unique sellers count per user
sellers = pd.DataFrame()
sellers = df.groupby("member_id").s_member_id.nunique().to_frame().reset_index()
sellers.columns = ['member_id', 'NUMBER_SELLERS']
```

Listing 5: Finding how many sellers each customer purchased from

The number of unique items each customer purchased is calculated by counting unique URUN_IDs for each member and is stored in column called "NUMBER_UNIQUE_ITEMS" as show in Listing 6:

```
# number of unique product ids per user
products = pd.DataFrame()
products = df.groupby("member_id").URUN_ID.nunique().to_frame().reset_index()
products.columns = ['member_id', 'NUMBER_UNIQUE_ITEMS']
```

Listing 6:

The number of items each customer purchased is calculated by counting rows for each customer and is stored in column called "NUMBER_ITEMS", as shown in Listing 7.

The total of amounts and average amount per item are also calculated in Listing 7.

```
# how many items did each member buy?
piv1 = df.pivot_table(index=["member_id"], values=["Amount"],
aggfunc=[np.sum,np.mean,len])
piv1.fillna(0, inplace=True)
piv1.reset_index(inplace=True)
piv1.columns = ['member_id', 'TOTAL_AMOUNT', 'AVERAGE_AMOUNT_PER_ITEM',
'NUMBER_ITEMS']
```

Listing 7:

Since the transactions are from many number of categories, the number of transactions per category is very low. Thus, only the top category is used for classification. The first letter/figure is the top category.

New columns for each top category were added to the data frame, so it's possible to store how many purchases each customer made from each category.

```
# get only the top category
df['TOP_CATEGORY'] = df['CATC'].astype(str).str[0]
# how many items did each member buy from each top category?
piv4 = df.pivot_table(index=["member_id", "TOP_CATEGORY"], values=["PAYMENT_ID"],
aggfunc=[len])
piv4.columns = piv4.columns.map('|'.join)
piv4.reset_index(inplace=True)

# convert top categories index into columns
piv4 = piv4.pivot_table(index='member_id', values='len|PAYMENT_ID',
columns='TOP_CATEGORY')
piv4.fillna(0, inplace=True)
piv4.reset_index(inplace=True)
```

Listing 8:

The number of unique top categories is stored in another column called "NUMBER_DIFFERENT_TOPCATEGORY".

```
def count_nonzero(*args):
    cnt=0
    for a in args:
        if a>0:
            cnt=cnt+1
    return cnt

piv4['NUMBER_DIFFERENT_TOPCATEGORY'] = piv4.apply(lambda row:
count_nonzero(row['1'],row['2'],row['3'],row['a'],row['b'],row['c'],row['e'],row['f
'],row['g'],row['h'],row['i'],row['j'],row['k'],row['l'],row['m'],row['n'],row['p']
,row['r'],row['t'],row['u'],row['v'],row['x'],row['y'],row['z']), axis=1)
```

Listing 9:

The number of times each member had a bad experience is stored in the column called "NUMBER_BAD_EXPERIENCE".

```
# how many times did each member have a bad experience?
piv5 = df.pivot_table(index=["member_id"], values=["Bad_Experience"],
aggfunc=[sum])
piv5.fillna(0, inplace=True)
piv5.reset_index( inplace=True)
piv5.columns = ['member_id', 'NUMBER_BAD_EXPERIENCE']
```

Listing 10:

In order to find in how many 3-months periods each customer made purchases, the transactions were handled in 4 separate periods ending in TODAY. This number is stored in a column called "NUMBER_PURCHASE_PERIODS".

```
# divide transactions in the last year into 4 periods
PERIOD1 = TODAY - timedelta(days=365)
PERIOD2 = TODAY - timedelta(days=273)
PERIOD3 = TODAY - timedelta(days=182)
PERIOD4 = TODAY - timedelta(days=91)

df_period1 = df[ (df['DATE'] <= pd.to_datetime(PERIOD2)) ]
df_period2 = df[ ((df['DATE'] <= pd.to_datetime(PERIOD3)) & (df['DATE'] >
pd.to_datetime(PERIOD2))  ) ]
df_period3 = df[ ((df['DATE'] <= pd.to_datetime(PERIOD4)) & (df['DATE'] >
pd.to_datetime(PERIOD3))  ) ]
df_period4 = df[ (df['DATE'] > pd.to_datetime(PERIOD4)) ]

# how many items did each member buy in each period
piv6a = df_period1.pivot_table(index=["member_id"], values=["URUN_ID"],
aggfunc=[len])
piv6a.fillna(0, inplace=True)
piv6a.reset_index(inplace=True)
piv6a.columns = ['member_id', 'ITEMS_PERIOD1']

piv6b = df_period2.pivot_table(index=["member_id"], values=["URUN_ID"],
aggfunc=[len])
piv6b.fillna(0, inplace=True)
piv6b.reset_index(inplace=True)
piv6b.columns = ['member_id', 'ITEMS_PERIOD2']

piv6c = df_period3.pivot_table(index=["member_id"], values=["URUN_ID"],
aggfunc=[len])
piv6c.fillna(0, inplace=True)
piv6c.reset_index(inplace=True)
piv6c.columns = ['member_id', 'ITEMS_PERIOD3']

piv6d = df_period4.pivot_table(index=["member_id"], values=["URUN_ID"],
aggfunc=[len])
piv6d.fillna(0, inplace=True)
piv6d.reset_index(inplace=True)
piv6d.columns = ['member_id', 'ITEMS_PERIOD4']

piv6 = pd.merge(piv6a, piv6b, 'outer', on=['member_id'])
piv6 = pd.merge(piv6, piv6c, 'outer', on=['member_id'])
piv6 = pd.merge(piv6, piv6d, 'outer', on=['member_id'])

piv6.fillna(0, inplace=True)
```

Listing 11:

After merging these pivot tables, some new columns were added:

- DATE_RANGE: number of days from the first transaction date to the last transaction date
- SINCE_FIRST: number of days from the first transaction date to TODAY
- SINCE_LAST: number of days from the last transaction date to TODAY
- AVERAGE_PERIOD: average number of days between creating carts
- AVERAGE_AMOUNT_PER_CART: average amount per cart

```
piv['DATE_RANGE'] = piv['LAST_PURCHASE'] - piv['FIRST_PURCHASE']
piv['DATE_RANGE'] = piv['DATE_RANGE'].dt.days

piv['SINCE_FIRST'] = pd.to_datetime(TODAY) - piv['FIRST_PURCHASE']
piv['SINCE_FIRST'] = piv['SINCE_FIRST'].dt.days

piv['SINCE_LAST'] = pd.to_datetime(TODAY) - piv['LAST_PURCHASE']
piv['SINCE_LAST'] = piv['SINCE_LAST'].dt.days

# average time between each purchase
members['AVERAGE_PERIOD'] = members['DATE_RANGE'] / members['NUMBER_CARTS']
members['AVERAGE_AMOUNT_PER_CART'] = members['TOTAL_AMOUNT'] /
members['NUMBER_CARTS']
```

Listing 12:

An extra column called “ANALYTICAL_DISTANCE” was created, but not used in the final model because it was causing overfitting.

$$|AB|=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

Figure 5:

```
# analytical distance between first and last purchase
members['ANALYTICAL_DISTANCE'] = ((TODAY - members['LAST_PURCHASE']).dt.days**2 +
(TODAY - members['FIRST_PURCHASE']).dt.days**2)**0.5
```

Listing 13:

Data Cleansing

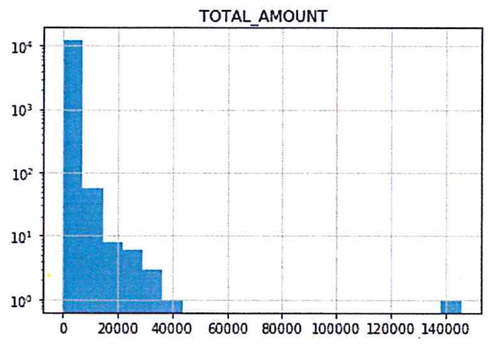
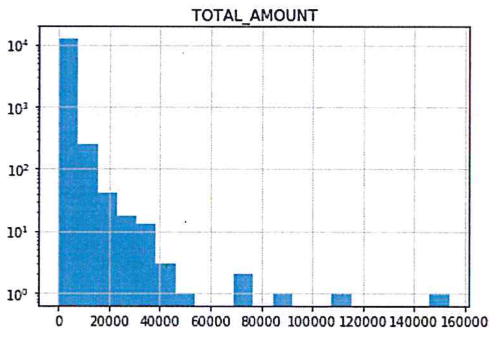
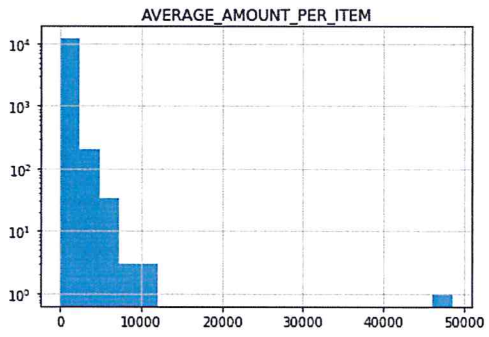
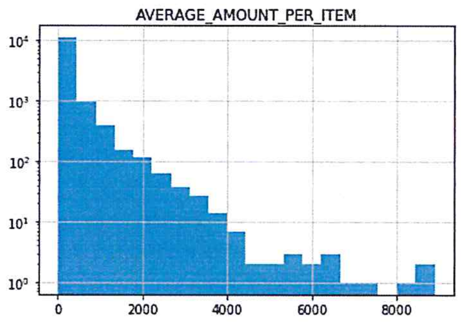
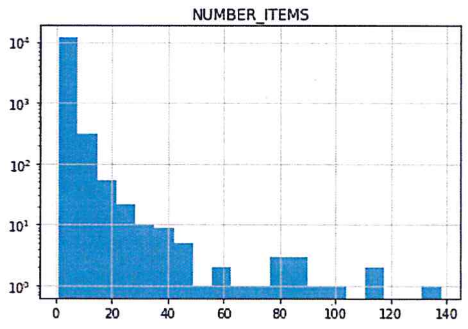
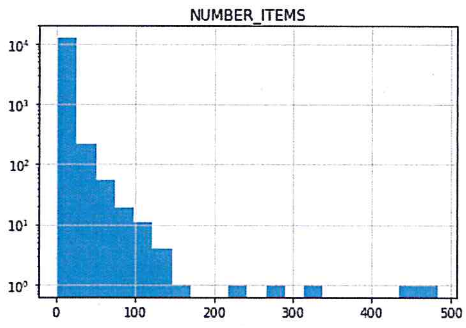
Data cleansing was required before training the model.

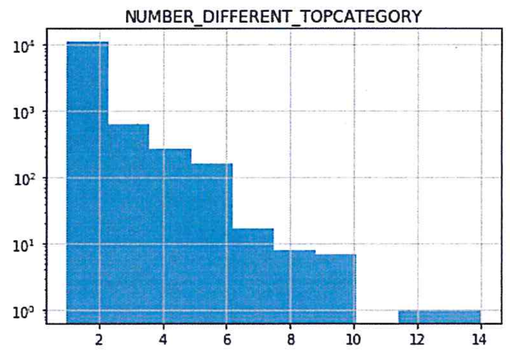
After determining the status of members on PREDICTION_DATE, the transactions after TODAY were discarded.

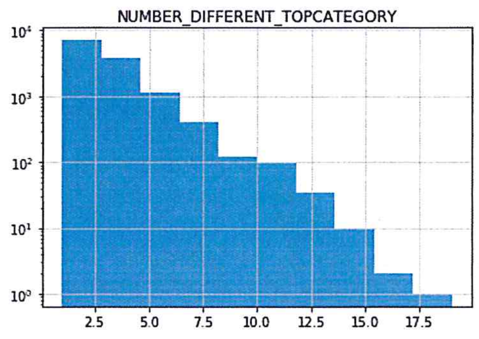
The customers who made only 1 transaction were also discarded, because they can’t be regarded as permanent GittiGidiyor customers.

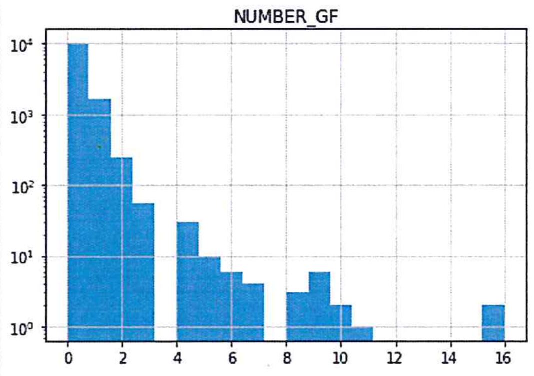
Proof of Method Validity

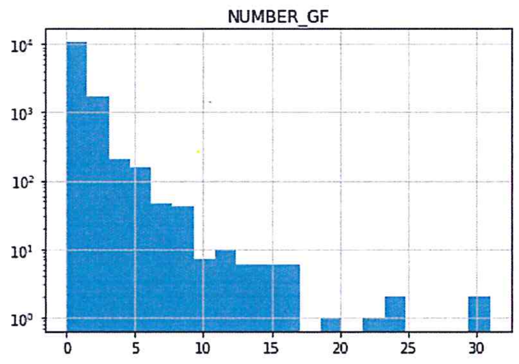
The model was trained by using the transactions between 28.04.2015 and 28.04.2016, then the model was tested with the transactions between 29.04.2016 and 30.04.2017. Comparison of extracted features prove that the distribution is similar in these two periods. Please note that Table 7 is in log scale, while Table 8 is not. The distribution of most features is very similar when some outliers are discarded.

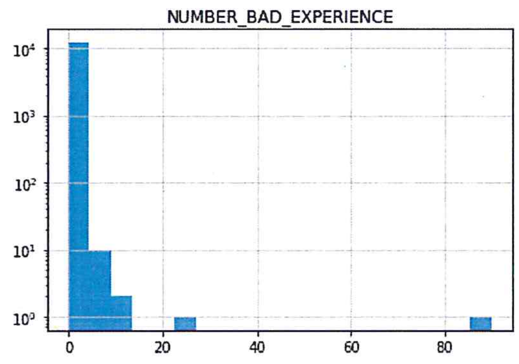
Train & Validation with Log Scale (28.04.2015 and 28.04.2016)	Test with Log Scale (29.04.2016 and 30.04.2017)
	
	
	

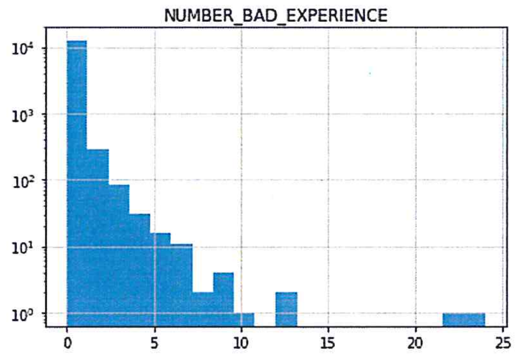

NUMBER_DIFFERENT_TOPCATEGORY
10⁴
10³
10²
10¹
10⁰
2 4 6 8 10 12 14

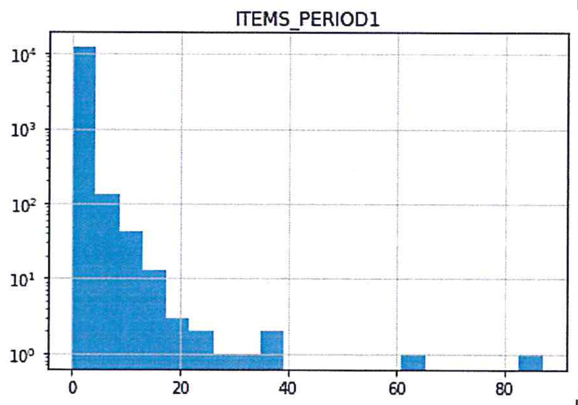

NUMBER_DIFFERENT_TOPCATEGORY
10⁴
10³
10²
10¹
10⁰
2.5 5.0 7.5 10.0 12.5 15.0 17.5

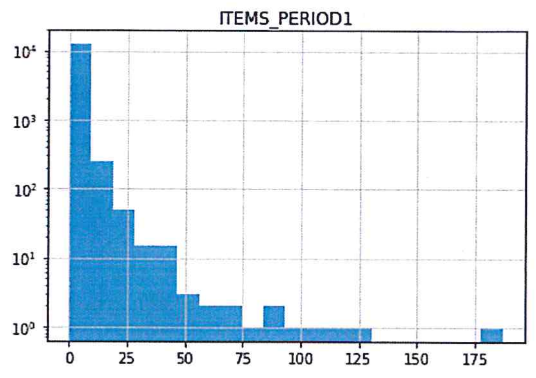

NUMBER_GF
10⁴
10³
10²
10¹
10⁰
0 2 4 6 8 10 12 14 16

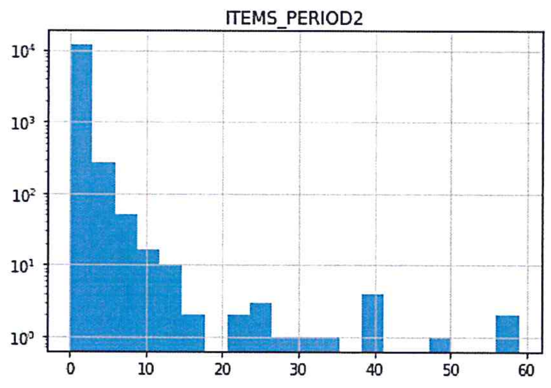

NUMBER_GF
10⁴
10³
10²
10¹
10⁰
0 5 10 15 20 25 30

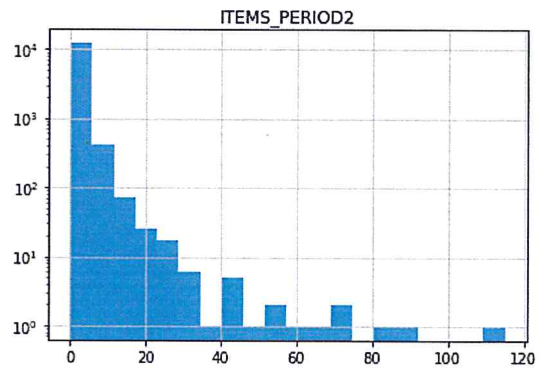

NUMBER_BAD_EXPERIENCE
10⁴
10³
10²
10¹
10⁰
0 20 40 60 80

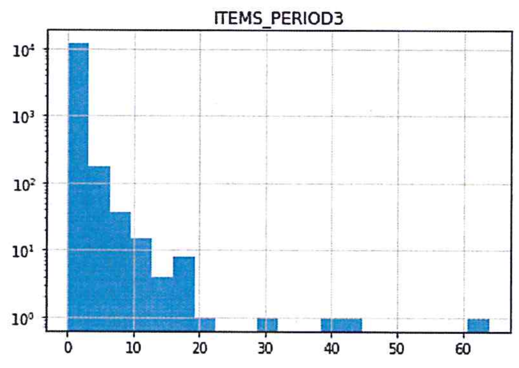

NUMBER_BAD_EXPERIENCE
10⁴
10³
10²
10¹
10⁰
0 5 10 15 20 25

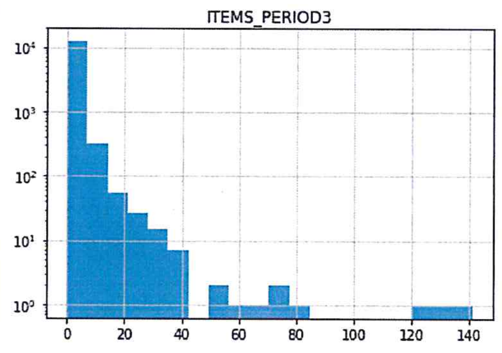

ITEMS_PERIOD1
10⁴
10³
10²
10¹
10⁰
0
20
40
60
80

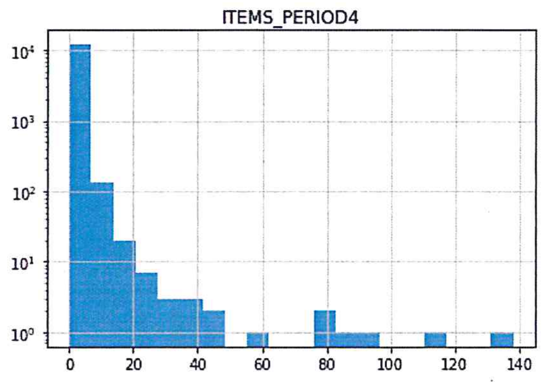

ITEMS_PERIOD1
10⁴
10³
10²
10¹
10⁰
0
25
50
75
100
125
150
175

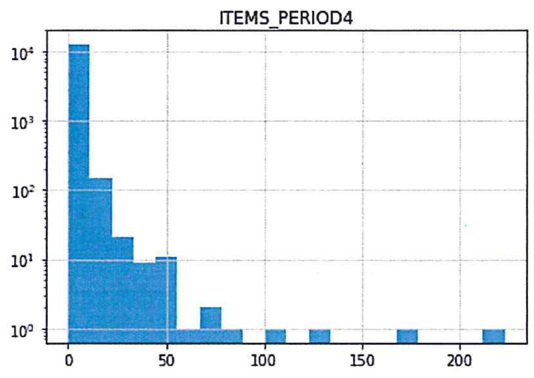

ITEMS_PERIOD2
10⁴
10³
10²
10¹
10⁰
0
10
20
30
40
50
60

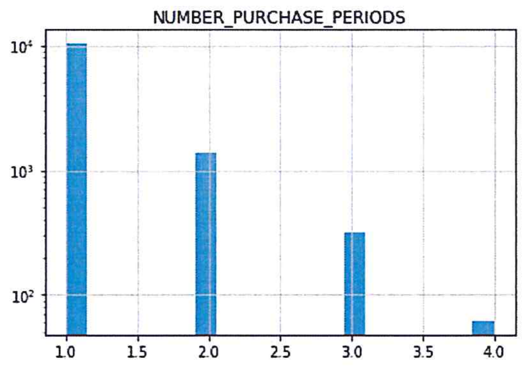

ITEMS_PERIOD2
10⁴
10³
10²
10¹
10⁰
0
20
40
60
80
100
120

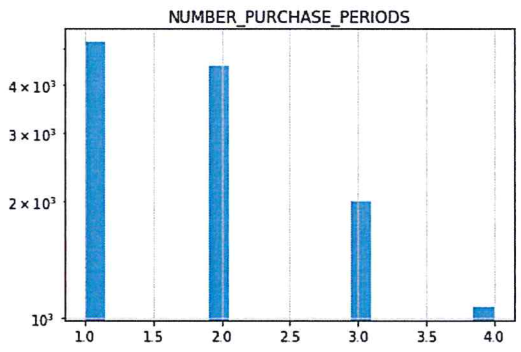

ITEMS_PERIOD3
10⁴
10³
10²
10¹
10⁰
0
10
20
30
40
50
60

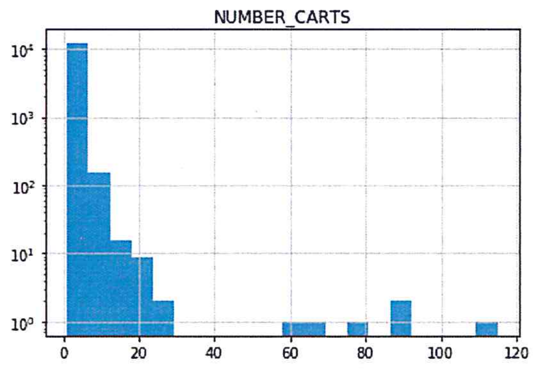

ITEMS_PERIOD3
10⁴
10³
10²
10¹
10⁰
0
20
40
60
80
100
120
140

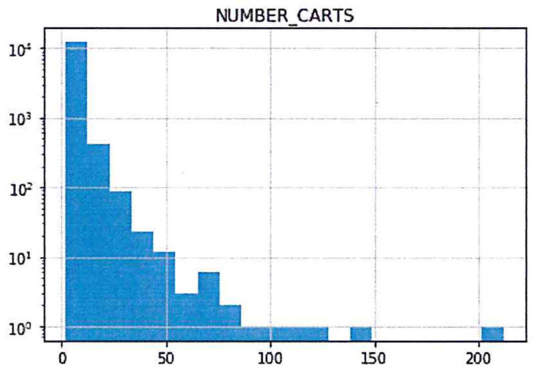

ITEMS_PERIOD4
10⁰ 10¹ 10² 10³ 10⁴
0 20 40 60 80 100 120 140

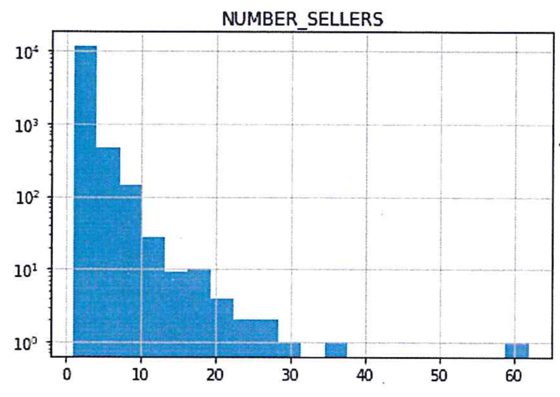

ITEMS_PERIOD4
10⁰ 10¹ 10² 10³ 10⁴
0 50 100 150 200

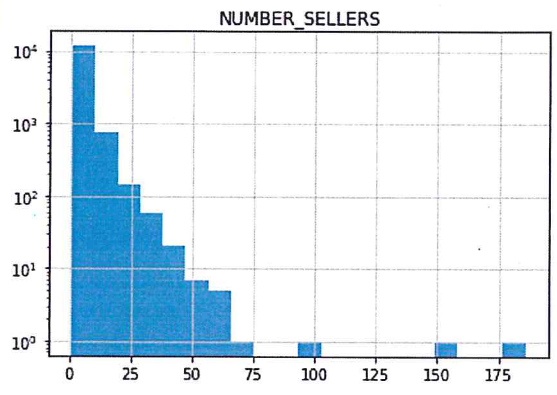

NUMBER_PURCHASE_PERIODS
10² 10³ 10⁴
1.0 1.5 2.0 2.5 3.0 3.5 4.0

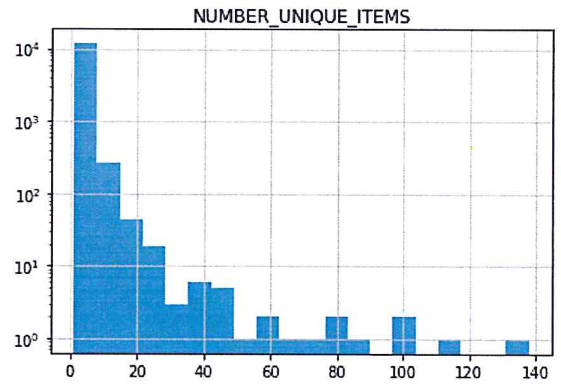

NUMBER_PURCHASE_PERIODS
10³ 2 × 10³ 3 × 10³ 4 × 10³
1.0 1.5 2.0 2.5 3.0 3.5 4.0

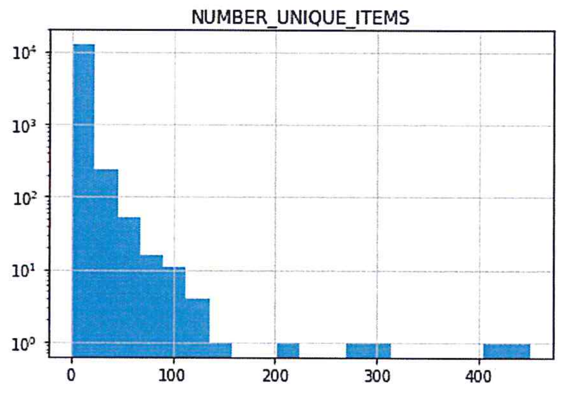

NUMBER_CARTS
10⁰ 10¹ 10² 10³ 10⁴
0 20 40 60 80 100 120

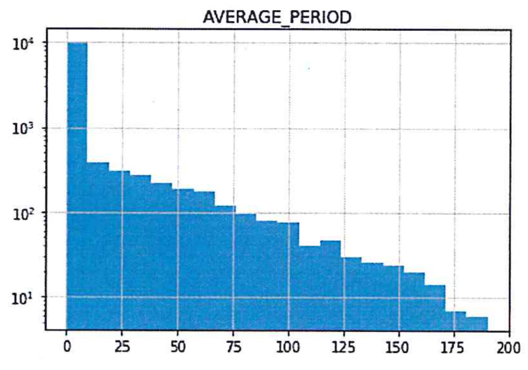

NUMBER_CARTS
10⁰ 10¹ 10² 10³ 10⁴
0 50 100 150 200

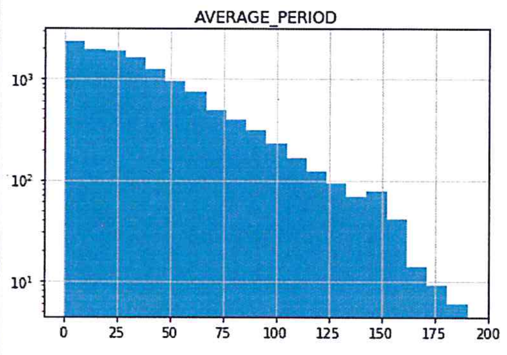

NUMBER_SELLERS
10⁴
10³
10²
10¹
10⁰
0 10 20 30 40 50 60


NUMBER_SELLERS
10⁴
10³
10²
10¹
10⁰
0 25 50 75 100 125 150 175


NUMBER_UNIQUE_ITEMS
10⁴
10³
10²
10¹
10⁰
0 20 40 60 80 100 120 140


NUMBER_UNIQUE_ITEMS
10⁴
10³
10²
10¹
10⁰
0 100 200 300 400


AVERAGE_PERIOD
10⁴
10³
10²
10¹
0 25 50 75 100 125 150 175 200


AVERAGE_PERIOD
10³
10²
10¹
0 25 50 75 100 125 150 175 200

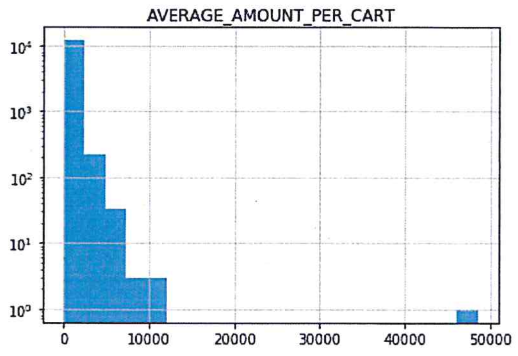
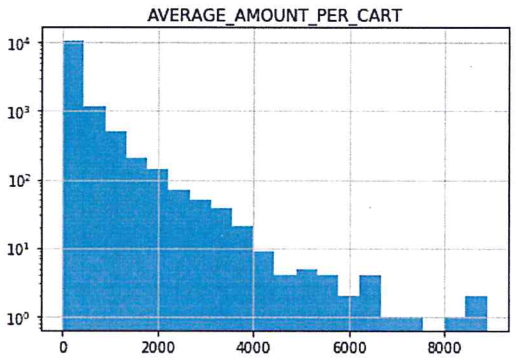
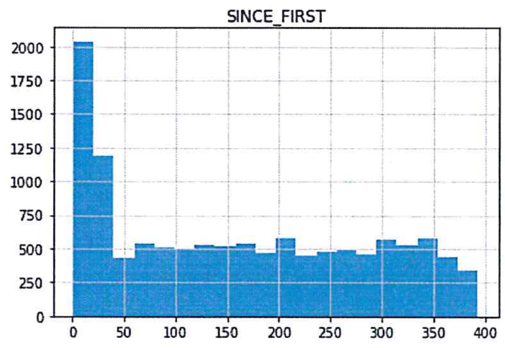
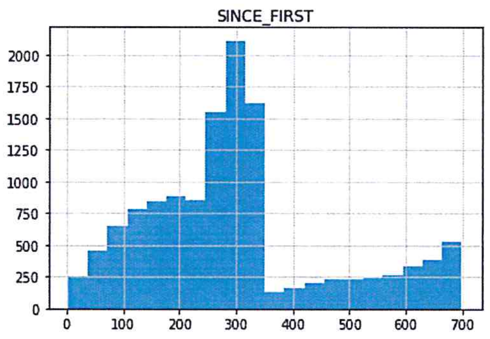
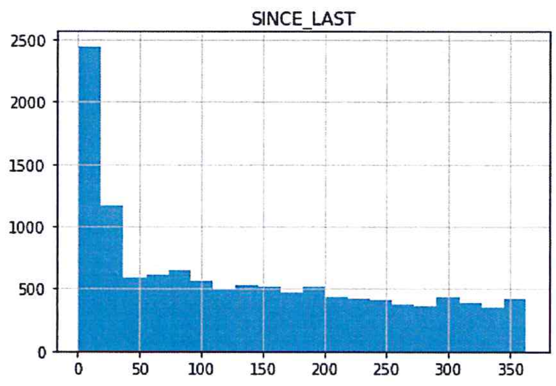
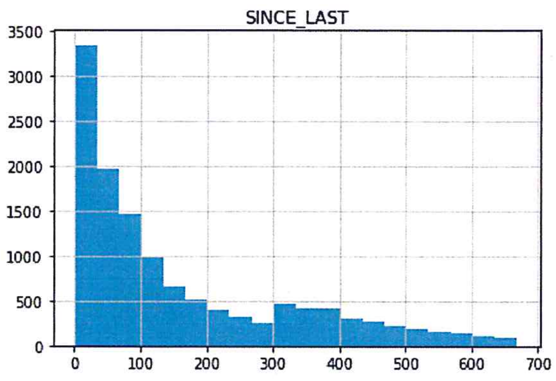
	

Table 7: Feature Comparison between Train and Test Time Period (Log Scale)

Train & Validation (28.04.2015 and 28.04.2016)	**Test (29.04.2016 and 30.04.2017)**
	
	

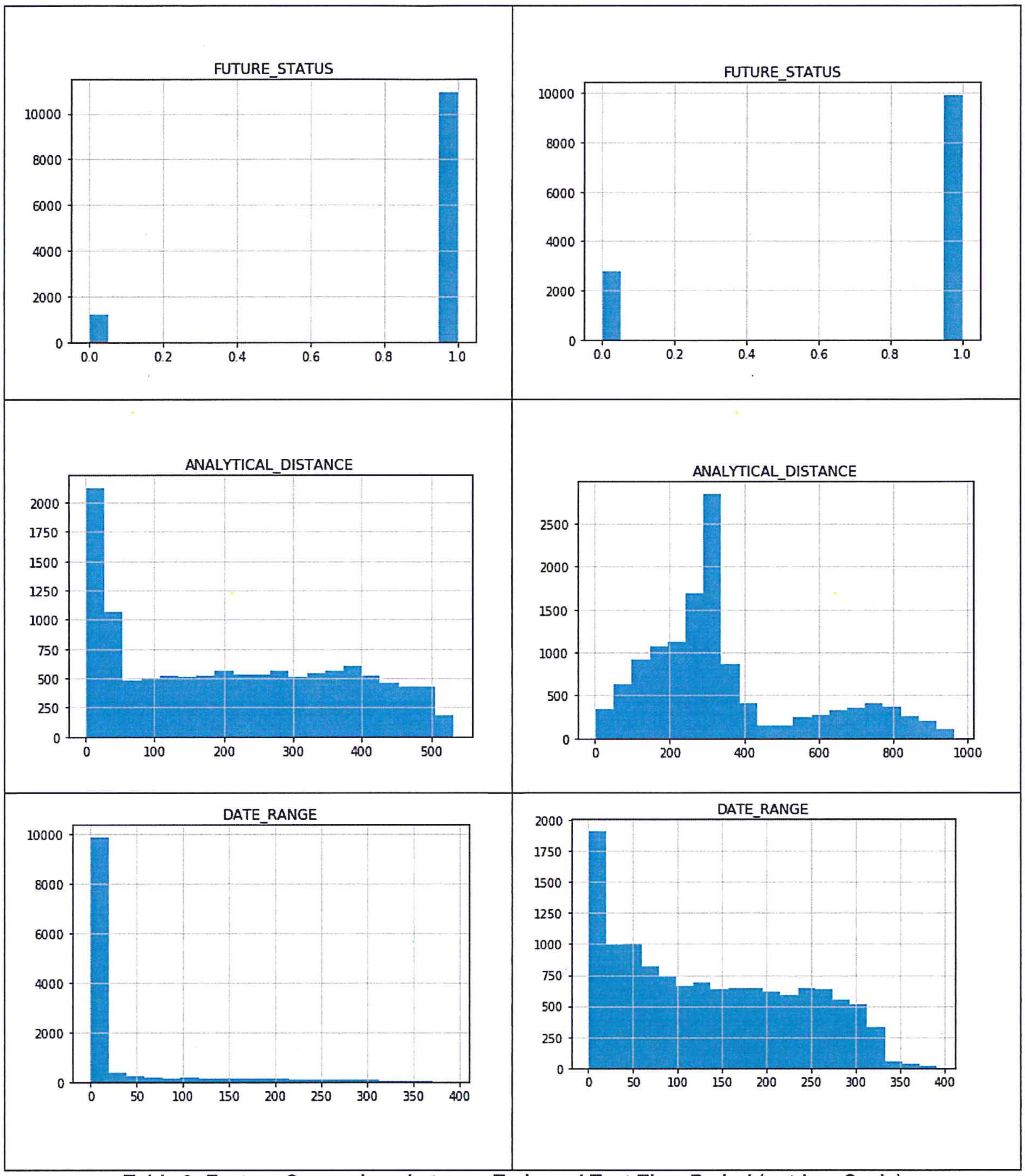


Table 8: Feature Comparison between Train and Test Time Period (not Log Scale)

Class Imbalance Problem

When the number of churned/not churned customers are compared a problem is observed. The number of active customers is much more than the number of passive customers. This problem is called "Class Imbalance Problem". Since the number of samples from these two classes are not close to each other, the trained model may be biased.

```
members_active  = members[members['FUTURE_STATUS'] == 1]
members_passive = members[members['FUTURE_STATUS'] == 0]


# members who made more than 1 transaction (not oneshot members)
members_active  = members_active[members_active['NUMBER_CARTS'] > 1 ]
members_active  = members_active[members_active['DATE_RANGE']>0]

members_passive = members_passive[members_passive['NUMBER_CARTS'] > 1 ]
members_passive = members_passive[members_passive['DATE_RANGE'] > 0 ]
```

Listing 14:

	Active	**Passive**
Number of Customers	3149	175

Table 9: Number of Active and Passive Customers in Test Dataset

In order to balance samples of these 2 classes, some rows of passive customers are duplicated. The code snippet in Listing 15 duplicates the rows of smaller class to balance them.

```
# add extra rows to balance
active_count = len( members_active.index )
passive_count =  len( members_passive.index )

if active_count > passive_count:
    diff = active_count - passive_count
    members = members_passive
    while diff > passive_count:
        members = pd.concat([members, members_passive])
        diff = diff - passive_count
    members = pd.concat([members, members_passive.sample(diff)])
    members = pd.concat([members, members_active])

elif passive_count > active_count:
    diff = passive_count - active_count
    members = members_active
    while diff > active_count:
        members = pd.concat([members, members_active])
        diff = diff - active_count
    members = pd.concat([members, members_active.sample(diff)])
    members = pd.concat([members, members_passive])

else:
    members = pd.concat([members_active, members_passive])
```

Listing 15: Balancing classes

Please note that, the classes are balanced only for training and validation datasets. Classes in the test dataset are not balanced.

Feature Selection

Many tests were run for feature selection. The most prominent features were selected by investigating the correlation between them and their importances in trained models.

The correlation of columns of the new data frame is shown as a heat map in Figure 6.

For example, AVERAGE_AMOUNT_PER_CART, TOTAL_AMOUNT and NUMBER_CARTS columns seem to be highly correlated, using all of them does not improve the accuracy, and thus AVERAGE_AMOUNT_PER_CART is not included in training.

The top categories seem to have very little correlation between them, however, their importance's are low in the trained models, thus they are not included in training too. Instead a new feature is defined as the number of different top categories and used as a feature.

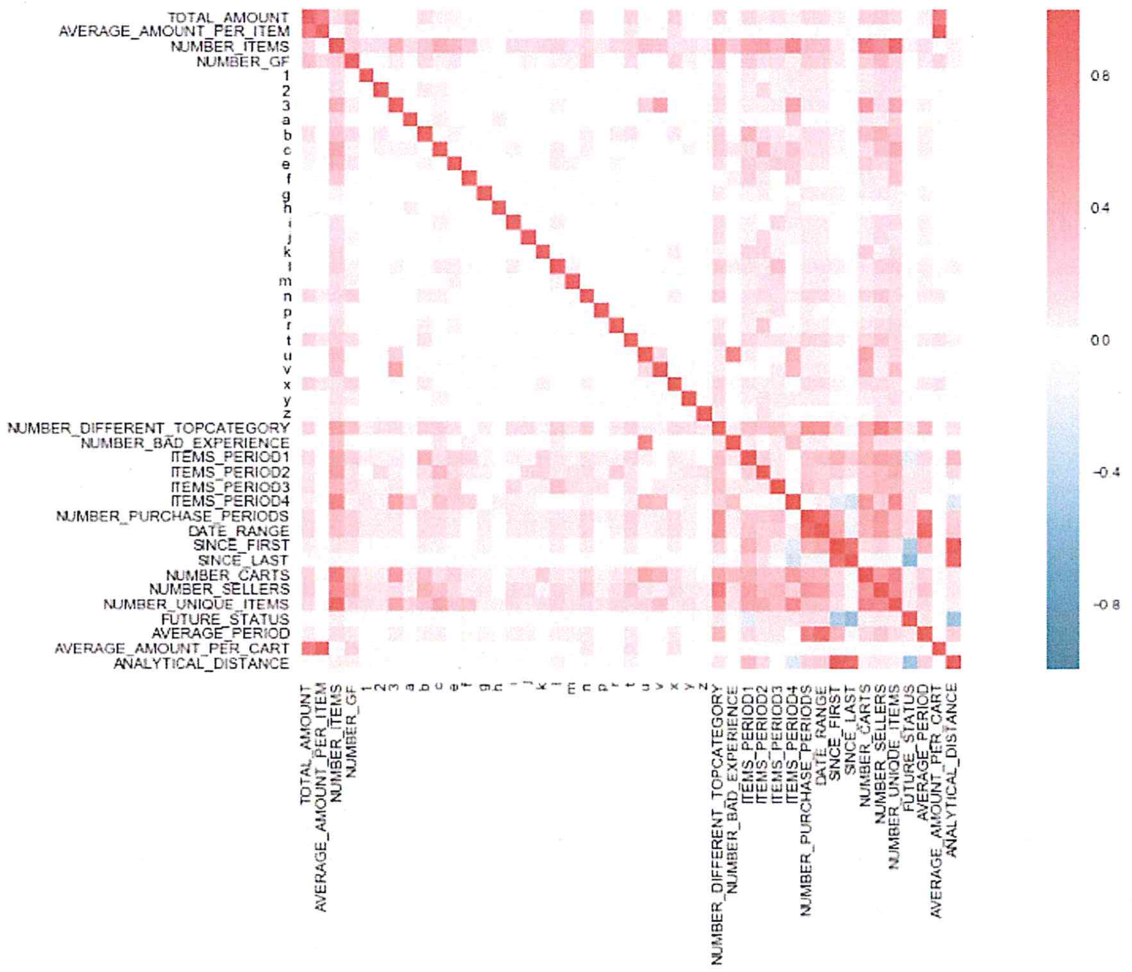


Figure 6: Feature Correlation Heat Map

Columns which depend on the date of purchase directly cause overfitting due to the definition of churn. They give too much hint for predicting and the accuracy increases unfairly. Thus LAST_PURCHASE and ANALYTICAL_DISTANCE are excluded too.

Figure 7 shows the heat map of the features that were decided to be used. All combinations of features with high and low correlation were tested one by one and these are found to be the optimal set of features.

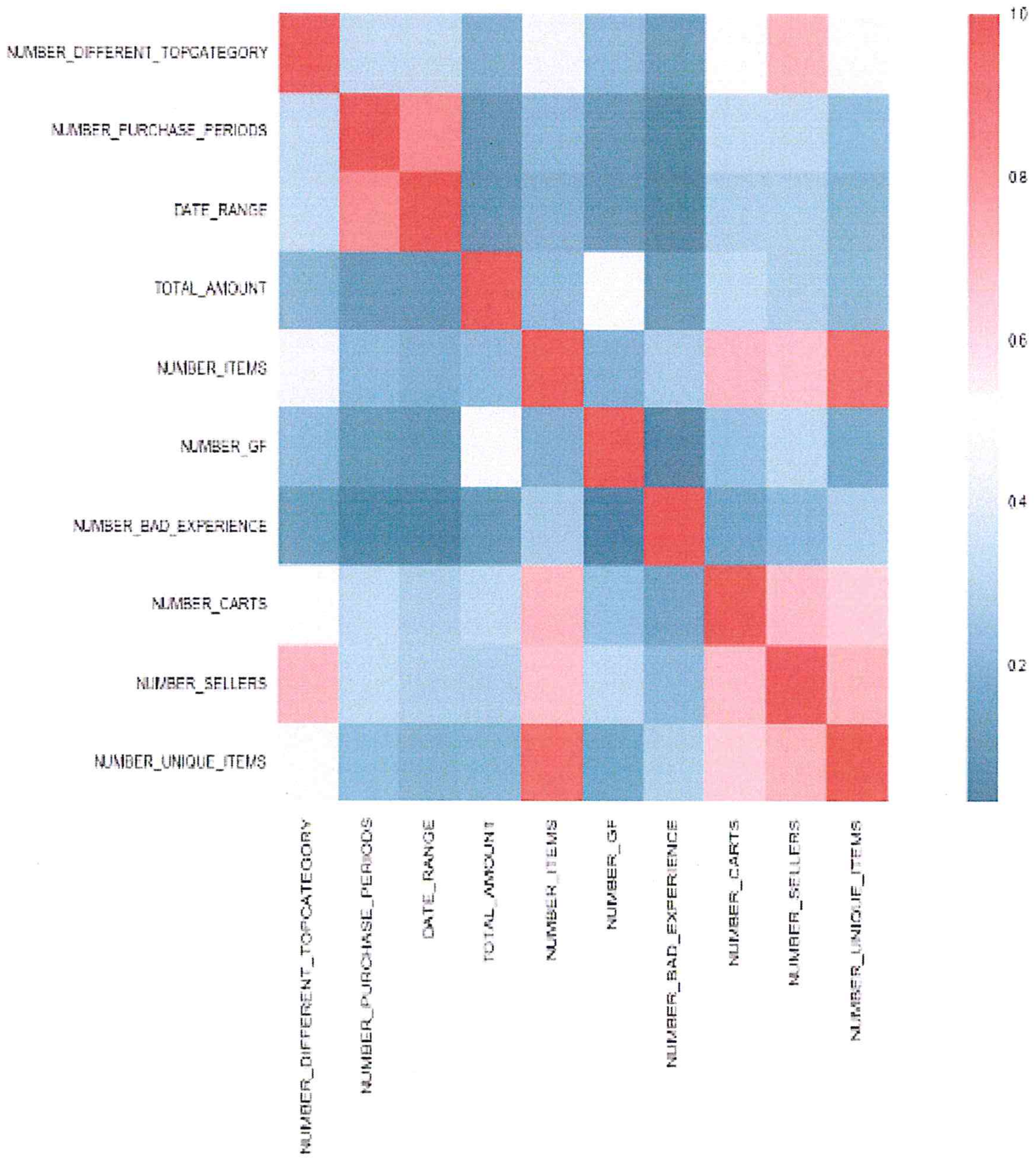


Figure 7 : Correlation Heat Map of Used Features

Table 10 gives the description of the features that are used in the final model.

Feature Name	Feature Description
'NUMBER_DIFFERENT_TOPCATEGORY'	how many different main categories a customer purchased from
'NUMBER_PURCHASE_PERIODS'	how many time periods a customer purchased in
'DATE_RANGE'	number of days between the last and the first purchases
'TOTAL_AMOUNT'	total amount of a customer's purchases
'NUMBER_ITEMS'	total number of a customer's purchases
'NUMBER_GF'	how many times a customer used the opportunity of the day
'NUMBER_BAD_EXPERIENCE'	how many times a customer had bad experiences
'NUMBER_CARTS'	how many different carts a customer has created
'NUMBER_SELLERS'	how many different sellers a customer purchased from
'NUMBER_UNIQUE_ITEMS'	How many different products a customer purchased

Table 10: Description of Features

Training and Algorithms

The model was trained using the transactions between 28.04.2015 and 28.04.2016. After the member based data frame was created, the data frame was split into train and validation datasets with the ratios of 80% and 20%.

The model was tested using the transactions between 29.04.2016 and 30.04.2017. The data was transformed into member-based format and the whole data frame was used for testing without splitting.

Many Machine Learning algorithms were tested to find the best accuracy. Feature importances, partial dependence and boosting iterations were examined while comparing the algorithms.

Gradient Boosting

Illustration of the effect of different regularization strategies for Gradient Boosting. The loss function used is binomial deviance. Regularization via shrinkage (learning_rate < 1.0) improves performance considerably. In combination with shrinkage, stochastic gradient boosting (subsample < 1.0) can produce more accurate models by reducing the variance via bagging. Subsampling without shrinkage usually does. In Figure 8, we can see boosting iterations validation set deviance

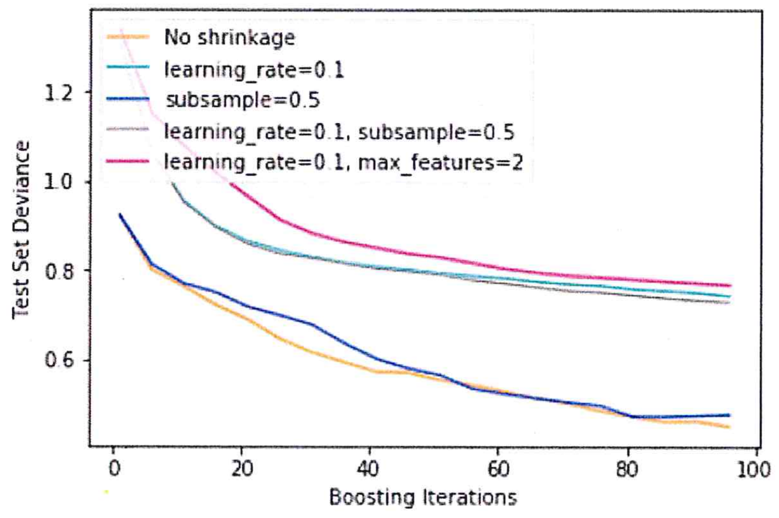


Figure 8: Boosting iterations

Table 12 shows that this model predicts 66% of the active customers and 82% of the passive customers correctly. Its performance for predicting churned customers is quite good.

However, Table 13 shows that its performance on test dataset is not so good. When the time window is moved to 29.04.2016 - 30.04.2017 for testing, it can predict only 38% of the active customers and 81% of the passive customers correctly.

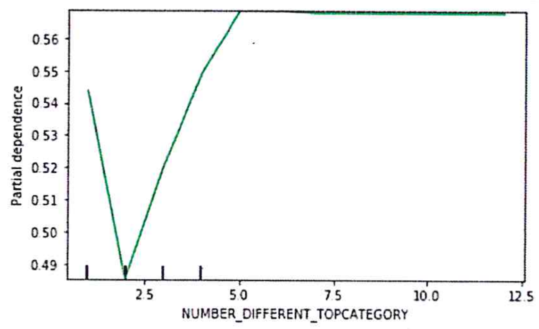
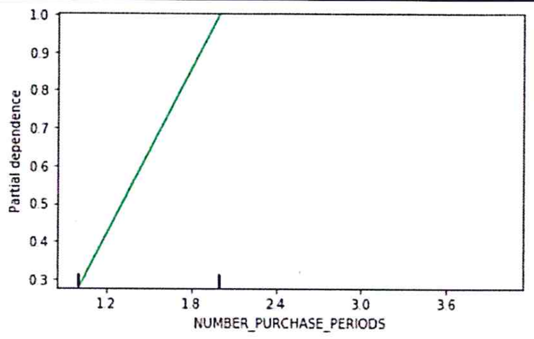
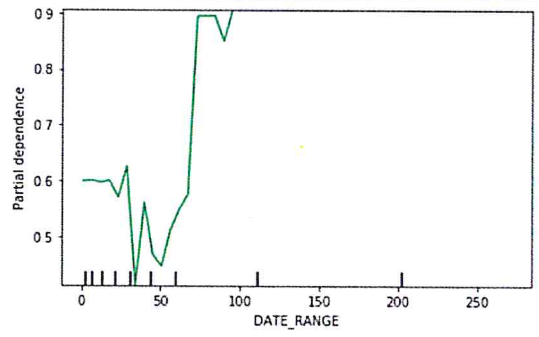
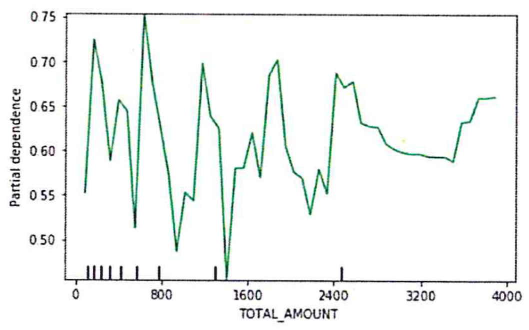
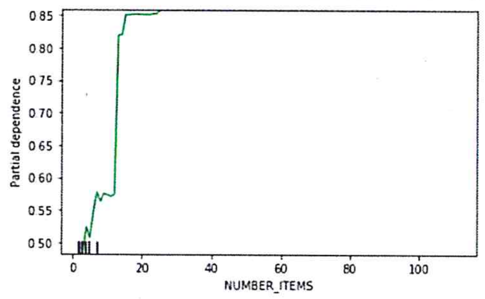
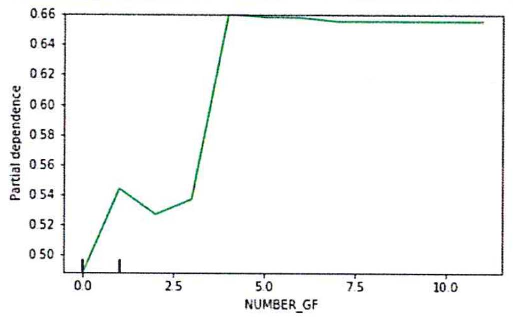
Moreover, Table 14 shows that this model takes longer time train than the models trained with other algorithms.

As a result, this model is not suitable for this use case since it can not predict churned customers with a good accuracy.

Gradient Boosting Regressor

Partial dependence plots show the dependence between the target function and a set of 'target' features, marginalizing over the values of all other features (the complement features).

In Table 11, the effects of selected features over the churn prediction accuracy is investigates. (1 stands for active customers and 0 stands for passive customers)

	
Customers that purchased from more than 2.5 categories are less likely to churn.	Probability of customers that make purchases in more than 2 periods to churn is less. Customers that make purchases in more than 2 periods will definitely not churn.
	
If the date range between the first and the last puchases is between 50 and 100, the customer is less likely to churn. If the range is less than 50, the probability is higher.	It's not possible to make a simple conclusion with the total amount feature alone.
	
Churn probability of a customer depends on the number of items she purchased. The probability is quite low if she purchased more than 20 items.	Customers who used Opportunity of the Day for more than 3 times are less likely to churn. Customers who did not use it are more likely to churn.

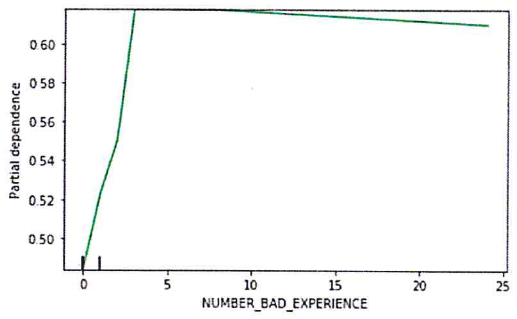
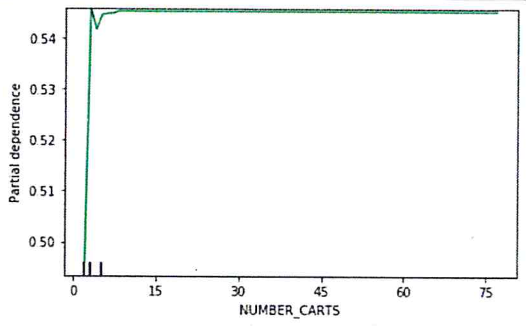
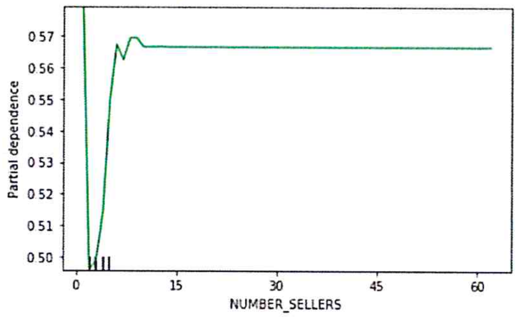
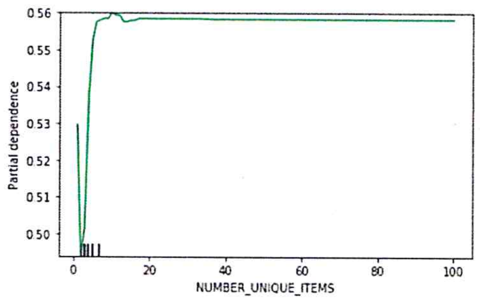
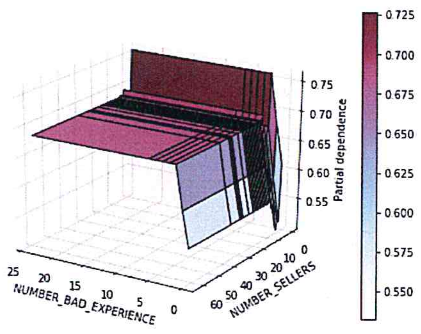
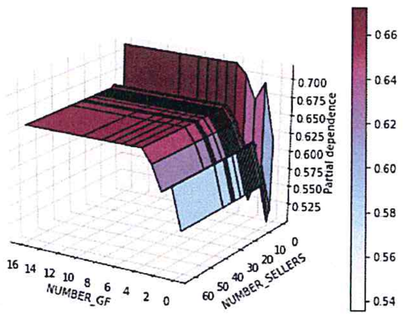
	
Cuustomers who did not file any bad experience complaints are more probably to churn. This may be interpretted as customers who report having bad experiences continue purchasing other items after their problems are solved, but many customers who have bad experiences don't report it and simply leave.	The probability of churn decreases if the number of carts exceeds 13.
	
Customers who purchase from 3 or less sellers are more loyal customers. It can be inferred that customers loyal to a single seller are also loyal to Gittigidiyor.	Customers who purchase more than 1 types of items are less likely to churn. If a customer purchased many different items, she is less likely to churn.
	

Table 11: Feature partial dependence

Decision Tree Classifier

max_depth parameter is one of the most critical parameters of Decision Trees. It directly controls over-fitting / under-fitting of the model. When it's too high, the model over-fits the data, in other words, it starts to memorize the samples. And when it's too low, the model under-fits the data, in other words, it can't learn the data well enough. 3 different max_depth values were tested: 5, 10 and unlimited.

max_depth=5

10 features were given. When max_depth was set to 5, the tree uses 7 features in total.

Figure 9 depicts the importance's of the features. The most important feature is 'NUMBER_PURCHASE_PERIODS.

Figure 10 shows the trained decision tree. It can be seen that it's a quite shallow and imbalanced tree, one feature is much more important than all others.

```
Feature ranking:
1. feature 1 (0.823396)
2. feature 2 (0.102383)
3. feature 4 (0.027617)
4. feature 7 (0.016520)
5. feature 3 (0.014385)
6. feature 9 (0.009600)
7. feature 0 (0.006099)
8. feature 8 (0.000000)
9. feature 6 (0.000000)
10. feature 5 (0.000000)
```

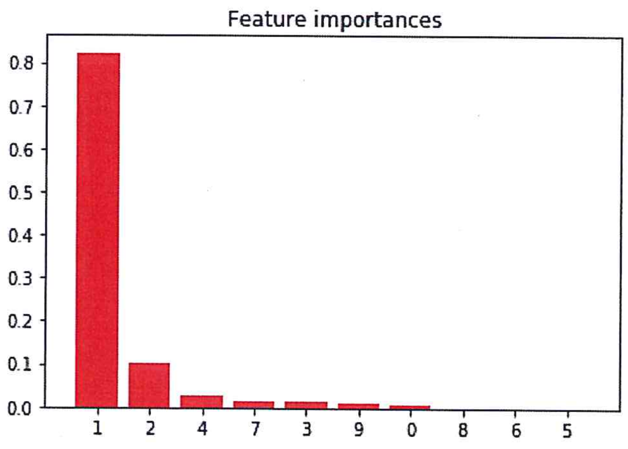


Figure 9 : Feature importance for max_depth=5

Visualizing a decision tree gives a lot of information about the model.

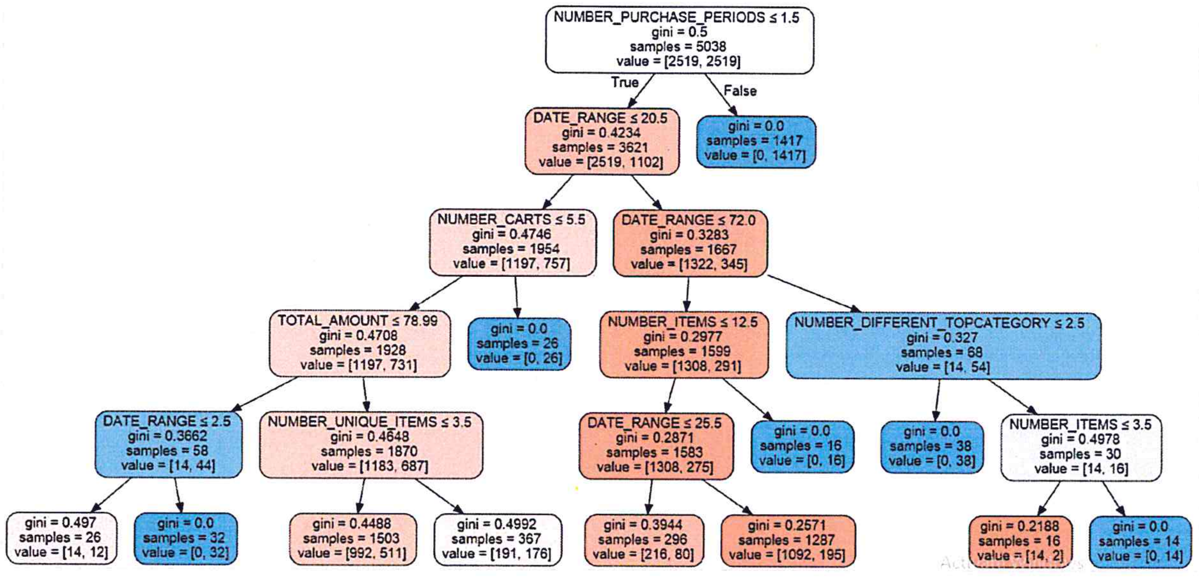


Figure 10 : Decision Tree Structure for max_depth=5

max_depth=10

When max_depth is set to 10, all features are used. NUMBER_PURCHASE_PERIODS is still the most important. When Figure 13 is investigated, it can be seen that it starts like the previous tree but the tree branches more with the increasing depth.

Scores increase when max_depth is increased to 10.

```
Feature ranking:
1. feature 1 (0.618000)
2. feature 3 (0.124404)
3. feature 2 (0.109252)
4. feature 7 (0.037144)
5. feature 4 (0.036683)
6. feature 0 (0.026766)
7. feature 8 (0.015500)
8. feature 5 (0.015212)
9. feature 9 (0.014373)
10. feature 6 (0.002666)
```

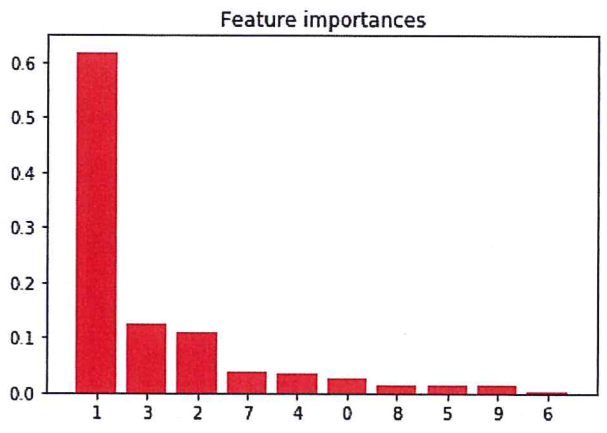


Figure 11 : Feature importance for max_depth=10

max_depth=None

When max_depth of the decision tree is set to unlimited, a tree with a depth of 25 is created. This can be said to be over-fitting.

Figure 12 shows that all features are used and the importances of features are so separated. This increases the reliability of the model.

```
Feature ranking:
1. feature 1 (0.391619)
2. feature 3 (0.240625)
3. feature 2 (0.160536)
4. feature 0 (0.047987)
5. feature 4 (0.035615)
6. feature 7 (0.029212)
7. feature 5 (0.028281)
8. feature 8 (0.026791)
9. feature 9 (0.019934)
10. feature 6 (0.019401)
```

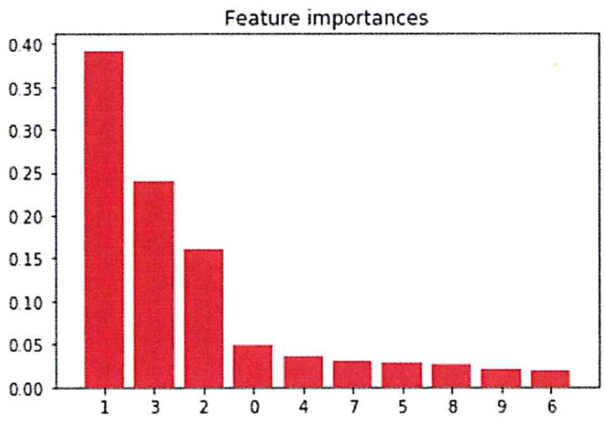


Figure 12 : Feature importance for max_depth=None

After investigating tree structures, validation and test scores need to be investigated too.

As seen in Table 12 a decision tree's performance increases as its max_depth increases. For validation dataset, a decision tree can predict 85% if its max_depth is 5, 87% if its max_depth is 10 and 97% if its max_depth is unlimited. However, this is not the only metric that should be considered, a more careful investigation is needed.Figure14 shows the trained decision tree.

Table 13 shows the performances on test dataset, which consists of the transactions in a different time range (between 29.04.2016 and 30.04.2017). The accuracy of a decision tree with max_depth 5 and 10 is 72% and its accuracy rises only 5% if its max depth is unlimited. Furthermore, its performance of predicting passive customers is too low only 9% for an unlimited depth decision tree. Thus a Decision Tree is not suitable for this use case.

Naive Bayes Classifier

Table 12 and Table 13 show that this algorithm performs badly on both validation and test datasets. Using different parameters did not improve its performance too. Thus, this algorithm is not suitable.

Multi-layer Perceptron Classifier

MLP Classifier is the basic deep neural network with only feed forward data flow. This classifier can predict only half of the customers that will churn. Thus, this algorithm is not suitable.

Support Vector Machines

As it can be seen on Table 14, this algorithms training time is longer than others.

In validation dataset it can predict almost all passive customers correctly. However, its performance on test dataset is only 7%.

Linear Discriminant Analysis

The model trained with LDA algorithm can predict almost all the passive customers correctly on the validation dataset. Moreover, its performance on test dataset is also better than other algorithms. As shown in Table 13, it can predict all of the passive customers and 76% of the active customers correctly.

Furthermore, its training time is good when compared with other algorithms as seen in Table 14.

Normalization should not be forgotten and feature selection should be done depending on well-founded investigation instead of instincts. Correlation of features does not spoil the accuracy always.

VALIDATION	Confusion Matrix	Precision			Recall			F1-score			Support		
		Pas.	Act.	Tot.	Pas.	Act.	Tot.	Pas.	Act.	Tot.	Pas.	Act.	Tot.
Gradient Boosting Classifier	[[612, 4], [218, 426]]	0.74	0.99	0.87	0.99	0.66	0.82	0.85	0.79	0.82	616	644	1260
Decision Tree Classifier(max_dept=5)	[[600, 16], [203, 441]]	0.75	0.96	0.86	0.97	0.68	0.83	0.85	0.8	0.82	616	644	1260
Decision Tree Classifier (max_dept=10)	[[602, 28], [132, 498]]	0.82	0.95	0.88	0.96	0.79	0.87	0.88	0.86	0.87	630	630	1260
Decision Tree Classifier (max_dept=None)	[[630, 0], [38, 592]]	0.94	1	0.97	1	0.94	0.97	0.97	0.97	0.97	630	630	1260
Naive Bayes Classifier	[[584, 32], [251, 393]]	0.7	0.92	0.81	0.95	0.61	0.78	0.8	0.74	0.77	616	644	1260
Linear Discriminant Analysis	[[612, 4], [280, 364]]	0.69	0.99	0.84	0.99	0.57	0.77	0.81	0.72	0.76	616	644	1260
Multi-layer Perceptron Classifier	[[603, 13], [364, 280]]	0.62	0.96	0.79	0.98	0.43	0.7	0.76	0.6	0.68	616	644	1260
Support Vector Machines	[[616, 0], [6, 638]]	0.99	1	1	1	0.99	1	1	1	1	616	644	1260

Table 12 : Performances of Algorithms Validation Dataset

TEST	Confusion Matrix	Precision			Recall			F1-score			Support		
		Pas.	Act.	Tot.	Pas.	Act.	Tot.	Pas.	Act.	Tot.	Pas.	Act.	Tot.
Gradient Boosting Classifier	[[1067, 1721], [2032, 7909]]	0.34	0.82	0.72	0.38	0.8	0.71	0.36	0.81	0.71	2788	9941	12729
Decision Tree Classifier (max_depth=5)	[[1182, 1606], [1987, 7954]]	0.37	0.83	0.73	0.42	0.8	0.72	0.4	0.82	0.72	2788	9941	12729
Decision Tree Classifier (max_dept=10)	[[879, 1909], [1637, 8304]]	0.36	0.82	0.71	0.32	0.84	0.72	0.34	0.83	0.72	2788	9941	12729
Decision Tree Classifier (max_dept=None)	[[260, 2528], [403, 9538]]	0.39	0.79	0.70	0.09	0.96	0.77	0.15	0.87	0.71	2788	9941	12729
Naive Bayes Classifier	[[1369, 1419], [2241, 7700]]	0.38	0.84	0.74	0.49	0.77	0.71	0.43	0.81	0.72	2788	9941	12729
Linear Discriminant Analysis	[[2782,6], [2384, 7557]]	0.54	1	0.9	1	0.76	0.81	0.7	0.86	0.83	2788	9941	12729
Multi-layer Perceptron Classifier	[[1494, 1294], [5254, 4687]]	0.22	0.78	0.66	0.54	0.47	0.49	0.31	0.59	0.53	2788	9941	12729
Support Vector Machines	[[187, 2601], [79, 9862]]	0.7	0.79	0.77	0.07	0.99	0.79	0.12	0.88	0.71	2788	9941	12729

Table 13 : Performances of Algorithms Test Dataset

Training time (seconds)	
Gradient Boosting Classifier	2.285000086
Decision Tree Classifier(max_depth=5)	0.005000114
Decision Tree Classifier(max_depth=10)	0.0110001564026
Decision Tree Classifier(max_depth=None)	0.010999917984
Naive Bayes Classifier	0.002000093
Linear Discriminant Analysis	0.006999969
Multi-layer Perceptron Classifier	0.498000145
Support Vector Machines	1.430999994

Table 14: Model training time

Conclusion

The data set was divided into training, validation and test and different algorithms were used. One of the innovative approaches was training and validating models in an earlier time window and testing the model with samples from a later time window. As a result of the studies, it was decided to use "Linear Discriminant Analysis" considering its short training time and especially the success of predicting passive customers.

In the dataset studied, it was seen that the number of active customers is much higher than that of passive customers. Several methods have been tried to solve this "Class imbalance" problem and it has been decided to replicate some lines of passive customers. Rows of smaller classes are duplicated to compensate classes with generated code.

In churn prediction, it is important to predict the customers that will churn, but it's also required to make a good prediction for customers that will not churn as well.

LDA algorithm gives a satisfactory result for both groups. It can predict almost all the customers that will churn, which is a very essential.

References

- https://academic.oup.com/bioinformatics/article/27/14/1986/194387/Classification-with-correlated-features
- https://ragulpr.github.io/assets/draft_master_thesis_martinsson_egil_wtte_rnn_2016.pdf
- https://ragulpr.github.io/2016/12/22/WTTE-RNN-Hackless-churn-modeling
- http://rstudio-pubs-static.s3.amazonaws.com/63125_2c67c7ed547946b7b1682379f1f1b2f4.html
- https://arxiv.org/ftp/arxiv/papers/1607/1607.07792.pdf
- https://www.researchgate.net/profile/Adnan_Amin/publication/280745147_Churn_Prediction_in_Telecommunication_Industry_Using_Rough_Set_Approach/links/562daddc08aef25a24431bbf.pdf
- https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python
- http://sebastianraschka.com/Articles/2014_python_lda.html
- http://journals.ama.org/doi/abs/10.1509/jmkr.43.2.276?code=amma-site
- http://www.sciencedirect.com.sci-hub.cc/science/article/pii/S0957417410006779
- https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/retail-big-data-analytics-solution-blueprint.pdf
- https://www.forbes.com/sites/groupthink/2013/05/14/can-big-data-cure-your-churn-rate/#20be06fc56c4
- https://www.ijarcce.com/upload/2016/si/ICRITCSA-16/IJARCCE-ICRITCSA%2028.pdf
- https://techwave.net/churn-analytics
- https://blog.kissmetrics.com/improve-by-predicting-churn
- http://www.aungz.com/PDF/Handling%20Class%20Imbalance%20in%20Customer%20Behavior%20Prediction.pdf

Appendix A

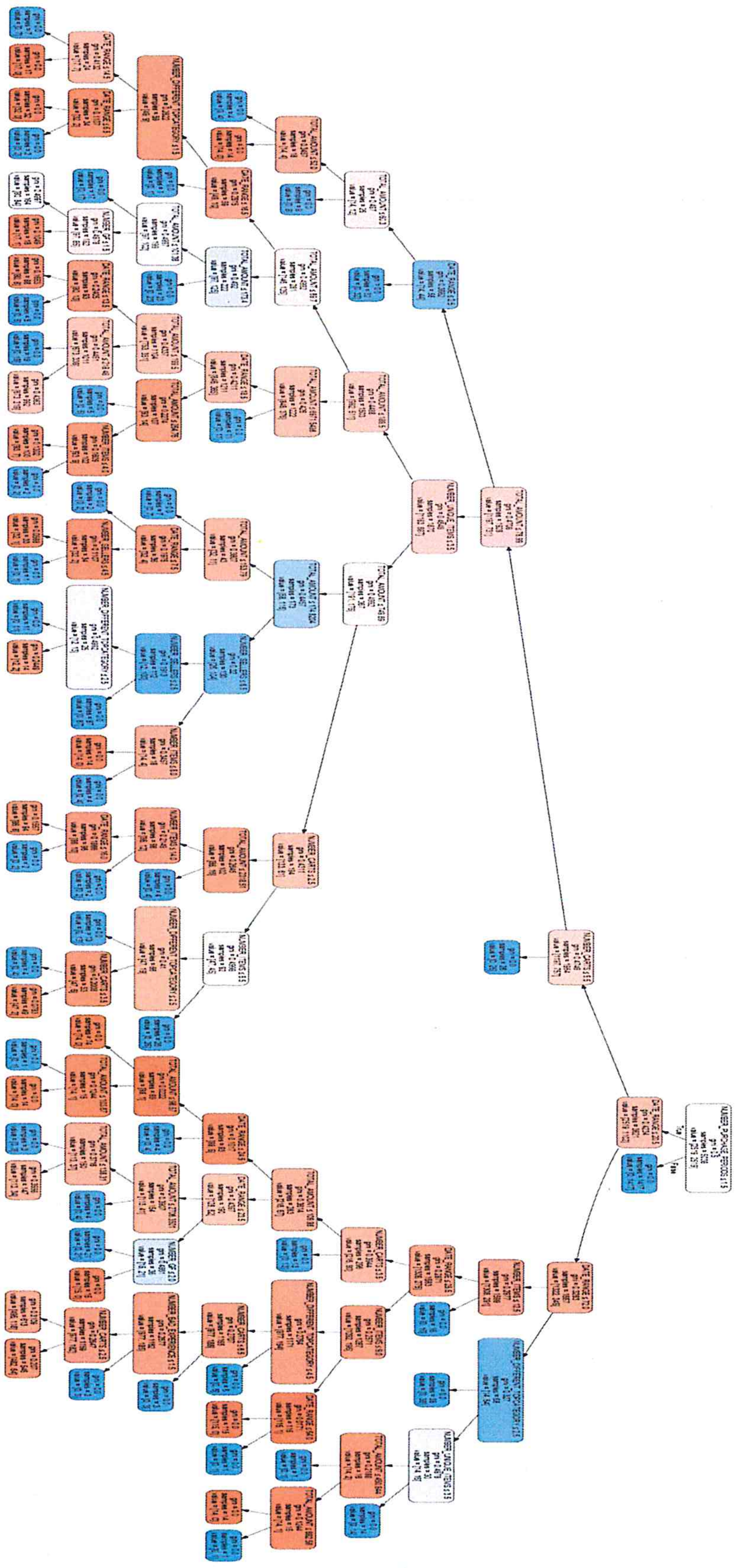


Figure 13 : Decision Tree Structure for max_depth=10

Appendix B

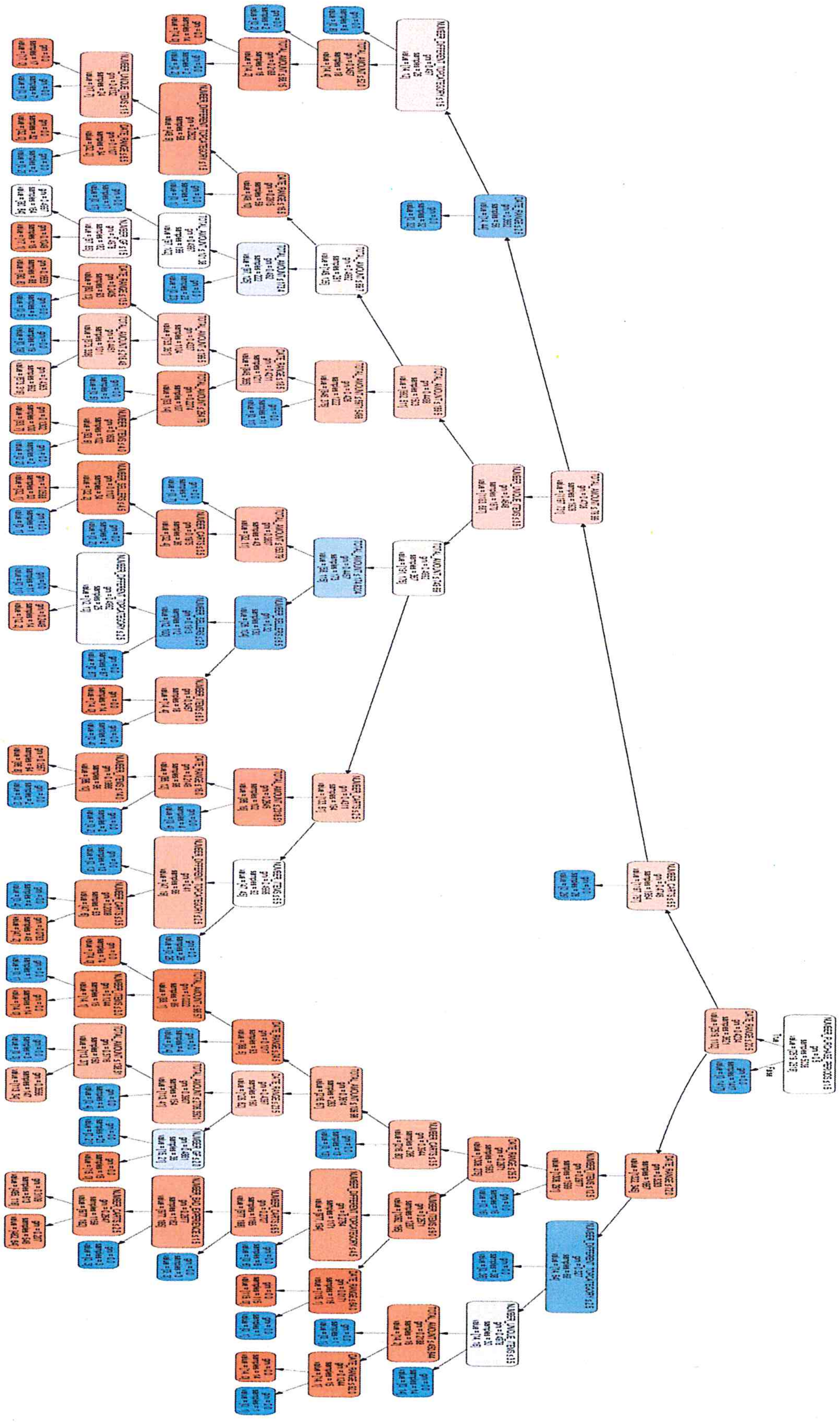


Figure 14: Decision Tree Structure for max_depth =None