Adama_Capstone_project_final

May 25, 2023

1 Guiding Customers Towards Product Subscription Through App Behavior Analysis

Imagine your company introduces an innovative app features. The first crucial step is finding customers. Offering a 24-hour free trial of the app can generate valuable user behavior data for marketing. This Capstone project aims to develop an ML model that classifies customers based on their app interactions.

Market: The target audience consists of customers who have downloaded the free version of our app services.

Product: The paid version offers enhanced features.

Goal: The objective of this Capstone project is to create a classification model that predicts which users are unlikely to subscribe to the paid membership, allowing for targeted marketing efforts. As a data scientist, my role is to identify users least likely to enroll in the paid version, to ensure efficient resource allocation. We assume that the free version access expires after 24 hours.

The data utilized in this case study originates from a fintech company aiming to offer its customers a paid mobile app subscription, enabling them to conveniently manage all their finances in a single location.

[ ]:

```
# Tools needed to installed to google colab as they are not part of main tools␣
  ↪installed by default
!pip install sweetviz # for intaractive report generation and visualization
# Tools for stat summary Table and feature selection
!pip install tableone
!pip install boruta
# Tool to get pdf file of Notebook
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
  ↪texlive-plain-generic
```

[2]:

```
# file path tools
from google.colab import drive #for Connecting to Google Drive
from IPython.display import Image #for image display
# EDA tools
import pandas as pd # for data manipulation using dataframes
import numpy as np # for data statistical analysis
import matplotlib.pyplot as plt # Import matplotlib for data visualisation
import seaborn as sns # Statistical data visualization
```

```
from dateutil import parser # to parse dates
%matplotlib inline
# Data preprocessing tools
from tableone import TableOne # stat summary table
from boruta import BorutaPy # for feature selection
from sklearn.ensemble import RandomForestClassifier #for feature selection
import sweetviz as sw #for interactive report generation

# Feature scaling, classification ML development and evaluation tools
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,␣
 ↪recall_score, classification_report
from sklearn.metrics import roc_curve, roc_auc_score
import pickle
# System error handling tools
import warnings
warnings.filterwarnings('ignore')
print('Libraries Import Successful')
```

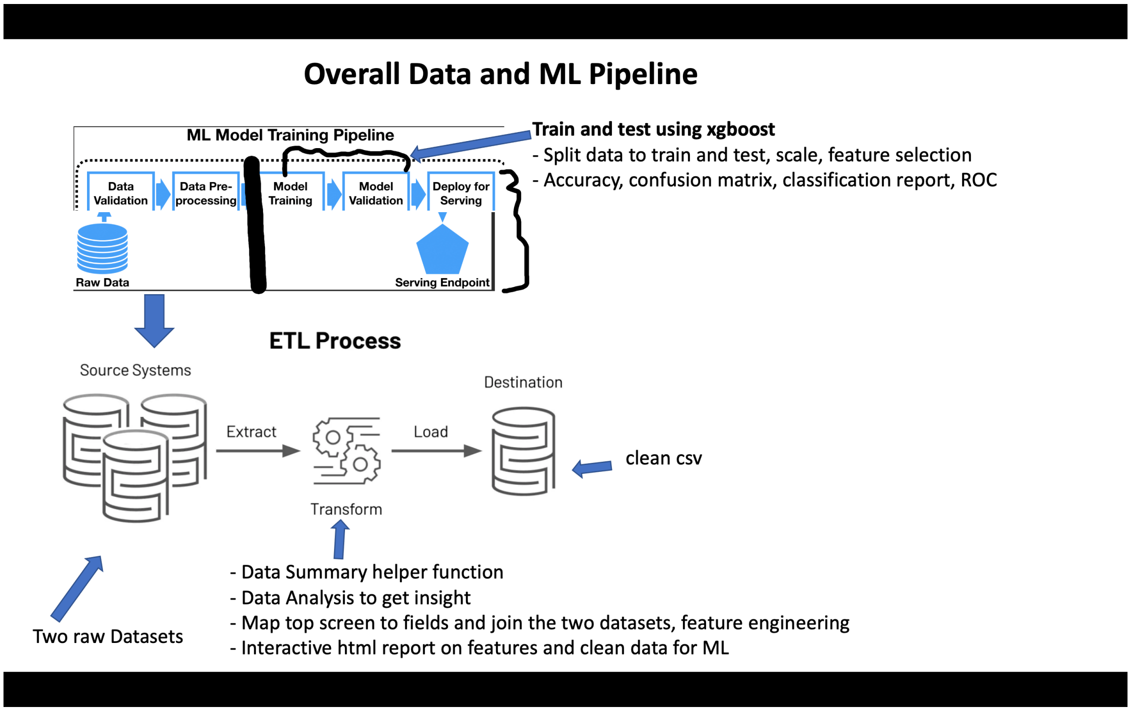
```
Libraries Import Successful
```

```
[3]: drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
[60]: Image('/content/drive/MyDrive/Data_incubator/data_sets/
 ↪adama_capstoneETL-ML-pipeline.png')
```

```
[60]:
```



Overall Data and ML Pipeline

Train and test using xgboost
- Split data to train and test, scale, feature selection
- Accuracy, confusion matrix, classification report, ROC

ETL Process

clean csv

Two raw Datasets

- Data Summary helper function
- Data Analysis to get insight
- Map top screen to fields and join the two datasets, feature engineering
- Interactive html report on features and clean data for ML

```
[5]: #importing data
     dataset = pd.read_csv('/content/drive/MyDrive/Data_incubator/data_sets/
      ↪appdata10.csv')
     dataset.head(10)
     # user is ID of each user (could be email)
     #first_open is the day they first used the cancer classifier
     #daysoftheweek (Sunday = 0 and Sat = 6)
     #hour is time they first used the app
     # screen_list is buttons (feature within app) they pushed within app
     #numscreens is number of screen_list visited
     # minigame here it specific games they payed (0 means no play, 1 played)
     #used_premium_feature if they tried paid version during free trial 0 = no, 1 =␣
      ↪yes
     #enrolled = 1 if the become member, 0 if not
     #enrolled_date is when they become a member (it can be any time)
     # like is whether or not they like the product
```

```
[5]:      user               first_open  dayofweek       hour  age  \
     0  235136  2012-12-27 02:14:51.273          3   02:00:00   23
     1  333588  2012-12-02 01:16:00.905          6   01:00:00   24
     2  254414  2013-03-19 19:19:09.157          1   19:00:00   23
     3  234192  2013-07-05 16:08:46.354          4   16:00:00   28
     4   51549  2013-02-26 18:50:48.661          1   18:00:00   31
     5   56480  2013-04-03 09:58:15.752          2   09:00:00   20
```

```
6  144649  2012-12-25 02:33:18.461          1   02:00:00   35
7  249366  2012-12-11 03:07:49.875          1   03:00:00   26
8  372004  2013-03-20 14:22:01.569          2   14:00:00   29
9  338013  2013-04-26 18:22:16.013          4   18:00:00   26

                                       screen_list  numscreens  minigame  \
0  idscreen,joinscreen,Cycle,product_review,ScanP…          15         0
1  joinscreen,product_review,product_review2,Scan…          13         0
2                                 Splash,Cycle,Loan            3         0
3  product_review,Home,product_review,Loan3,Finan…          40         0
4  idscreen,joinscreen,Cycle,Credit3Container,Sca…          32         0
5  idscreen,Cycle,Home,ScanPreview,VerifyPhone,Ve…          14         0
6         product_review,product_review2,ScanPreview            3         0
7  Splash,Cycle,Home,Credit3Container,Credit3Dash…          41         0
8  product_review,product_review2,ScanPreview,Ver…          33         1
9  Home,Loan2,product_review,product_review,produ…          19         0

   used_premium_feature  enrolled            enrolled_date  liked
0                     0         0                      NaN      0
1                     0         0                      NaN      0
2                     1         0                      NaN      1
3                     0         1  2013-07-05 16:11:49.513      0
4                     0         1  2013-02-26 18:56:37.841      1
5                     0         1  2013-04-03 09:59:03.291      0
6                     0         0                      NaN      0
7                     1         0                      NaN      0
8                     1         1  2013-04-27 22:24:54.542      0
9                     0         1  2013-04-26 18:31:58.923      0
```

[58]:
```
dataset.columns
```

[58]:
```
Index(['user', 'hour', 'age', 'numscreens', 'minigame', 'used_premium_feature',
       'location', 'Institutions', 'VerifyPhone', 'BankVerification',
       'VerifyDateOfBirth', 'ProfilePage', 'VerifyCountry', 'Cycle',
       'idscreen', 'Splash', 'Finances', 'Alerts', 'VerifyMobile',
       'VerifyHousing', 'VerifyHousingAmount', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'Login', 'WebView', 'SecurityModal',
       'ResendToken', 'TransactionList', 'Other', 'SavingCount', 'CMCount',
       'CCCount', 'LoansCount'],
      dtype='object')
```

[6]:
```
# Function to find missing values ,unique values ,data types  --> EDA
def df_summary(df):
    df_U = df.nunique()
    df_M = df.isnull().sum()
    df_I = df.dtypes
    ## converting all data to dataframe
    df_U = df_U.to_frame().reset_index()
    df_M = df_M.to_frame().reset_index()
    df_I = df_I.to_frame().reset_index()

    ## renaming columns to default 0 to some sensible name
    df_U = df_U.rename(columns= {0: 'Unique Data'})
    df_M = df_M.rename(columns= {0: 'Missing Data'})
    df_I = df_I.rename(columns= {0: 'Data Types'})

    ## concatting the 3 dataframes. Remember pd.merge can merge only 2 df at a␣
↪time
    output = pd.merge(pd.merge(df_M,df_U,on='index'),df_I,on='index')

    return output;
```

[7]:

```
df_summary(dataset)
```

[7]:

```
                   index  Missing Data  Unique Data Data Types
0                   user             0        49874      int64
1             first_open             0        49747     object
2              dayofweek             0            7      int64
3                   hour             0           24     object
4                    age             0           78      int64
5            screen_list             0        38799     object
6             numscreens             0          151      int64
7               minigame             0            2      int64
8   used_premium_feature             0            2      int64
9               enrolled             0            2      int64
10         enrolled_date         18926        31001     object
11                 liked             0            2      int64
```

[8]:

```
dataset.describe()
# about 62 % enrolled; about 17 % liked
```

[8]:

```
                user     dayofweek          age    numscreens      minigame  \
count   50000.000000  50000.000000  50000.00000  50000.000000  50000.000000
mean   186889.729900      3.029860     31.72436     21.095900      0.107820
std    107768.520361      2.031997     10.80331     15.728812      0.310156
min        13.000000      0.000000     16.00000      1.000000      0.000000
25%     93526.750000      1.000000     24.00000     10.000000      0.000000
50%    187193.500000      3.000000     29.00000     18.000000      0.000000
75%    279984.250000      5.000000     37.00000     28.000000      0.000000
max    373662.000000      6.000000    101.00000    325.000000      1.000000

       used_premium_feature      enrolled         liked
count          50000.000000  50000.000000  50000.000000
```

```
mean                   0.172020      0.621480      0.165000
std                    0.377402      0.485023      0.371184
min                    0.000000      0.000000      0.000000
25%                    0.000000      0.000000      0.000000
50%                    0.000000      1.000000      0.000000
75%                    0.000000      1.000000      0.000000
max                    1.000000      1.000000      1.000000
```

[9]:
```
# First set of Feature cleaning
dataset['hour'] = dataset.hour.str.slice(1, 3).astype(int)  #to convert hour␣
 ↪col to int, 1st and 2nd index of hour col, in python last index not included␣
 ↪so 3 used
```

[10]:
```
dataset.describe()
```

[10]:
```
                user     dayofweek          hour          age    numscreens  \
count   50000.000000  50000.000000  50000.000000  50000.00000  50000.000000
mean   186889.729900      3.029860     12.557220     31.72436     21.095900
std    107768.520361      2.031997      7.438072     10.80331     15.728812
min        13.000000      0.000000      0.000000     16.00000      1.000000
25%     93526.750000      1.000000      5.000000     24.00000     10.000000
50%    187193.500000      3.000000     14.000000     29.00000     18.000000
75%    279984.250000      5.000000     19.000000     37.00000     28.000000
max    373662.000000      6.000000     23.000000    101.00000    325.000000

           minigame  used_premium_feature      enrolled         liked
count  50000.000000          50000.000000  50000.000000  50000.000000
mean       0.107820              0.172020      0.621480      0.165000
std        0.310156              0.377402      0.485023      0.371184
min        0.000000              0.000000      0.000000      0.000000
25%        0.000000              0.000000      0.000000      0.000000
50%        0.000000              0.000000      1.000000      0.000000
75%        0.000000              0.000000      1.000000      0.000000
max        1.000000              1.000000      1.000000      1.000000
```

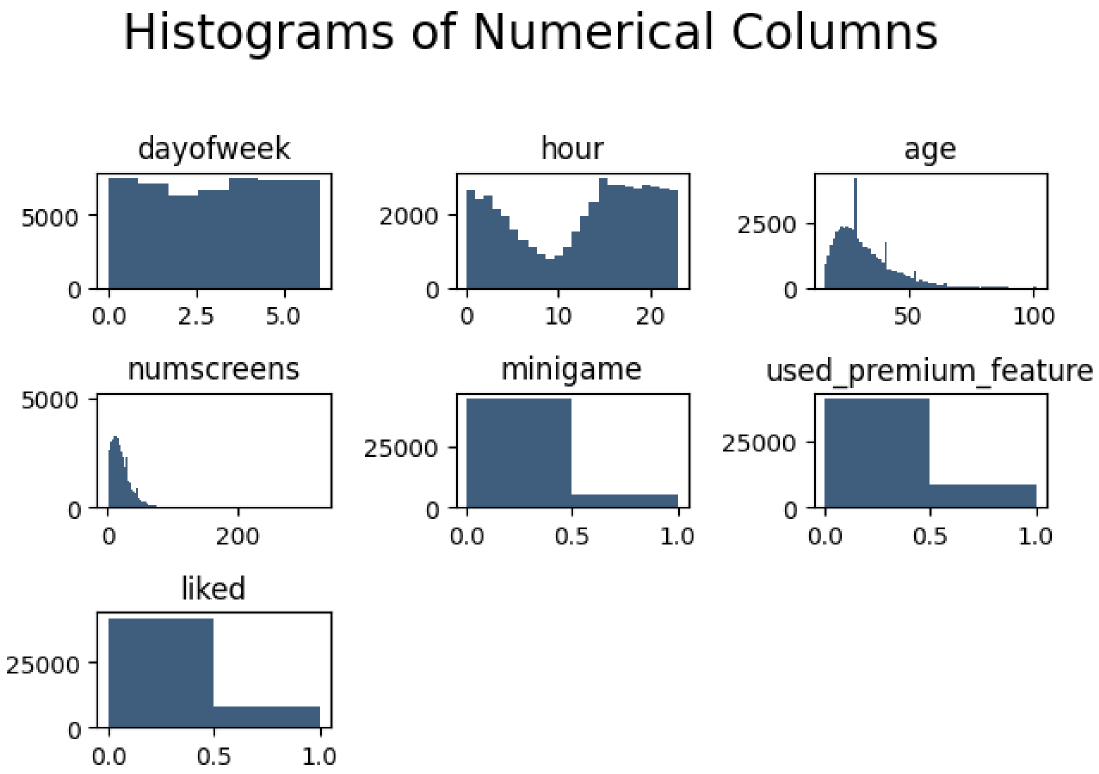
[11]:
```
# dropping less infomative, response valriable and non numeric columns to plot␣
 ↪hostogram
dataset2 = dataset.copy().drop(columns = ['user', 'screen_list',␣
 ↪'enrolled_date',
                                          'first_open', 'enrolled'])
dataset2.head(10)
```

[11]:
```
   dayofweek  hour  age  numscreens  minigame  used_premium_feature  liked
0          3     2   23          15         0                     0      0
1          6     1   24          13         0                     0      0
2          1    19   23           3         0                     1      1
3          4    16   28          40         0                     0      0
```

```
4          1   18   31          32      0                   0      1
5          2    9   20          14      0                   0      0
6          1    2   35           3      0                   0      0
7          1    3   26          41      0                   1      0
8          2   14   29          33      1                   1      0
9          4   18   26          19      0                   0      0
```

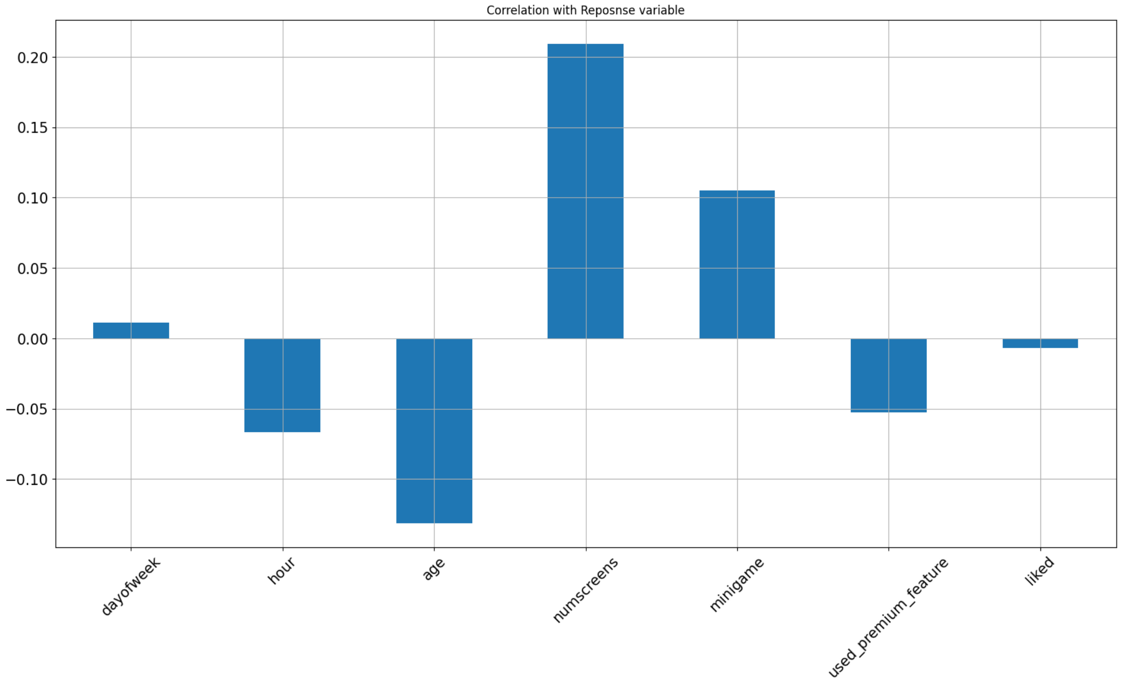
[12]:

```python
## Histograms to know data features distribution
plt.suptitle('Histograms of Numerical Columns', fontsize=20)
for i in range(1, dataset2.shape[1] + 1): #dataset 2 shape + as last index is not included in python
    plt.subplot(3, 3, i)
    f = plt.gca() #to print everything
#    f.axes.get_yaxis().set_visible(False)
    f.set_title(dataset2.columns.values[i - 1])

    vals = np.size(dataset2.iloc[:, i - 1].unique()) #to give bins based on data in its cols

    plt.hist(dataset2.iloc[:, i - 1], bins=vals, color='#3F5D7D')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
#plt.savefig('app_data_hist.jpg')

# from hist things like alright. deep in hour col is late night, minigame, used_premium, liked are heavy to zero side. Meaning not many people used
```



```
[13]: ## Correlation with Response Variable
      dataset2.corrwith(dataset.enrolled).plot.bar(figsize=(20,10),
                        title = 'Correlation with Reposnse variable',
                        fontsize = 15, rot = 45,
                        grid = True)
```

```
[13]: <Axes: title={'center': 'Correlation with Reposnse variable'}>
```

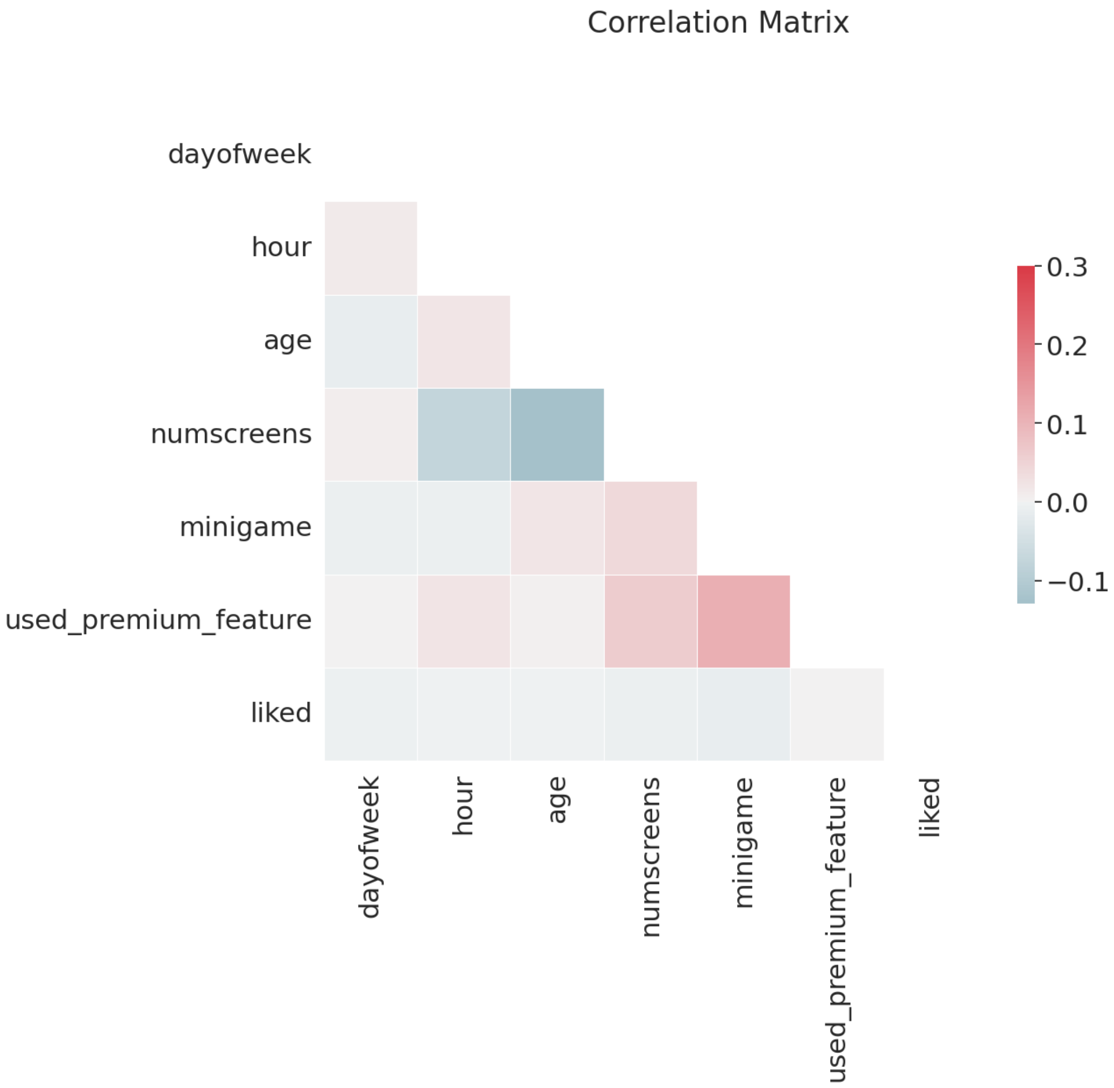


Note correlation plot: minigame and numscreens have +v correlation with respnse (enrolled) variable. The rest of freatures are in -ve correlation.

[14]:

```
## When we build a model we assume that all features are independant. So we need to check that. There should be no feature that depend on the other.
## Correlation Matrix
sns.set(style="white", font_scale=2)

# Compute the correlation matrix
corr = dataset2.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(12, 10))
f.suptitle("Correlation Matrix", fontsize = 24)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
[14]: <Axes: >
```



Note from correlation metxrix: There is no intense (light blue i.e -ve correlation red i.e +ve correlation) observed. So we can all features are independent. age and numscreens have slight -ve correlation, used_premium_feature have slight +ve correlation. But nothing too intense in general.

```
[15]: dataset.dtypes
```

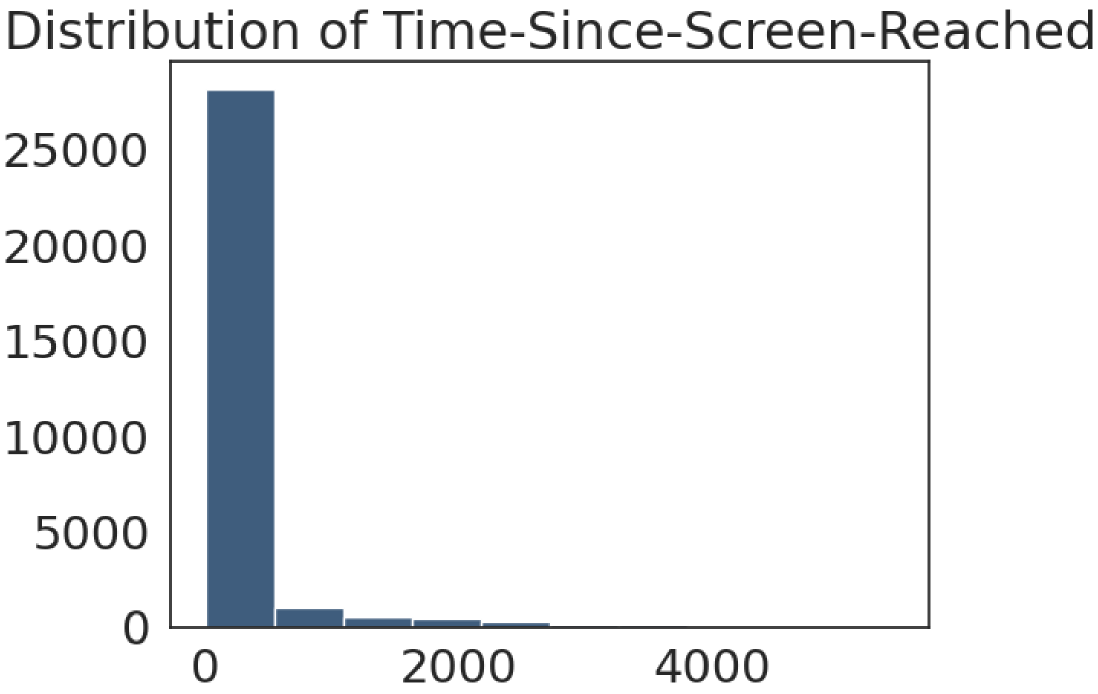
```
[15]: user                    int64
      first_open             object
      dayofweek               int64
      hour                    int64
      age                     int64
      screen_list            object
```

```
numscreens               int64
minigame                 int64
used_premium_feature     int64
enrolled                 int64
enrolled_date           object
liked                    int64
dtype: object
```

```
[16]: # Formatting Date Columns. We need to know how to evaluste our model. For this␣
       ↪we need to know the difference between
      # first_open and erolled so we know how our model perform by plotting the␣
       ↪distribution. To do so we need to perse our date cols.
      dataset["first_open"] = [parser.parse(row_date) for row_date in␣
       ↪dataset["first_open"]] #row_date can be any thing, here we are converting to␣
       ↪date time object;
      dataset["enrolled_date"] = [parser.parse(row_date) if isinstance(row_date, str)␣
       ↪else row_date for row_date in dataset["enrolled_date"]] #if else is to apply␣
       ↪only to str if not return row_data itseld
      dataset.dtypes
```

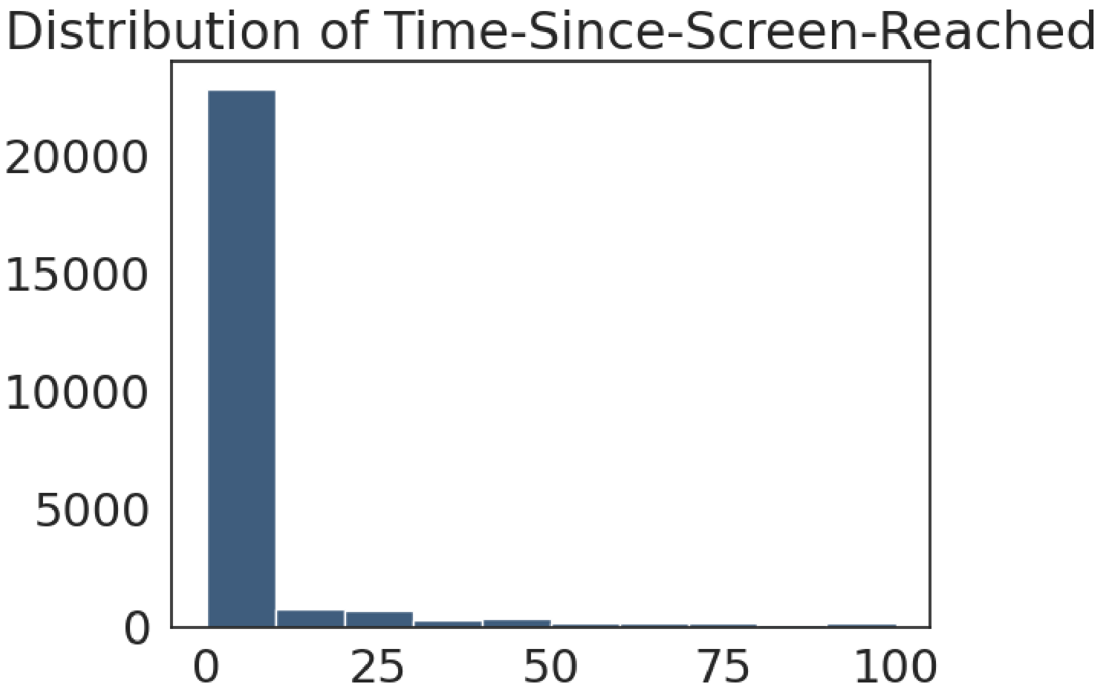
```
[16]: user                                int64
      first_open                 datetime64[ns]
      dayofweek                           int64
      hour                                int64
      age                                 int64
      screen_list                        object
      numscreens                          int64
      minigame                            int64
      used_premium_feature                int64
      enrolled                            int64
      enrolled_date              datetime64[ns]
      liked                               int64
      dtype: object
```

```
[17]: # Selecting Time For Response. Added column called difference by sutracting␣
       ↪first_open from enrolled_date. timedelta64[h] is make time difference
      # Most of the enrollemnt hapen in the first 500 hrs.
      dataset["difference"] = (dataset.enrolled_date-dataset.first_open).
       ↪astype('timedelta64[h]')
      response_hist = plt.hist(dataset["difference"].dropna(), color='#3F5D7D')
      plt.title('Distribution of Time-Since-Screen-Reached')
      plt.show()
```



Note from above plot: Most people enrolled in the first 500 hours. Let us see what happen in the first 100 hours below.

[18]:
```
#modify to first 100 hours; indeed first 100 hour is even too long. so let us␣
↪cut to 48 (two days) see below cell
plt.hist(dataset["difference"].dropna(), color='#3F5D7D', range = [0, 100])
plt.title('Distribution of Time-Since-Screen-Reached')
plt.show()
```



[19]:
```
dataset.loc[dataset.difference > 48, 'enrolled'] = 0 # so 48 (two days) might be good cut, so  if not enrolled in 48 hrs we consider them as not enrolled (set to 0)
dataset = dataset.drop(columns=['enrolled_date', 'difference', 'first_open']) #remove cols we do not need anymore
```

[20]:
```
dataset.head(10)
```

[20]:
```
     user  dayofweek  hour  age  \
0  235136          3     2   23
1  333588          6     1   24
2  254414          1    19   23
3  234192          4    16   28
4   51549          1    18   31
5   56480          2     9   20
6  144649          1     2   35
7  249366          1     3   26
8  372004          2    14   29
9  338013          4    18   26

                                         screen_list  numscreens  minigame  \
0  idscreen,joinscreen,Cycle,product_review,ScanP…          15         0
1  joinscreen,product_review,product_review2,Scan…          13         0
2                                  Splash,Cycle,Loan           3         0
```

```
3  product_review,Home,product_review,Loan3,Finan…          40          0
4  idscreen,joinscreen,Cycle,Credit3Container,Sca…          32          0
5  idscreen,Cycle,Home,ScanPreview,VerifyPhone,Ve…          14          0
6         product_review,product_review2,ScanPreview           3          0
7  Splash,Cycle,Home,Credit3Container,Credit3Dash…          41          0
8  product_review,product_review2,ScanPreview,Ver…          33          1
9  Home,Loan2,product_review,product_review,produ…          19          0

   used_premium_feature  enrolled  liked
0                     0         0      0
1                     0         0      0
2                     1         0      1
3                     0         1      0
4                     0         1      1
5                     0         1      0
6                     0         0      0
7                     1         0      0
8                     1         0      0
9                     0         1      0
```

[21]:

```
# Let load top screen data set. It is the top used screens
## Formatting the screen_list Field
## we had information regarding the to most used screens
# Load Top Screens
top_screens = pd.read_csv('/content/drive/MyDrive/Data_incubator/data_sets/
 ↪top_screens.csv').top_screens.values # we select only values of top_sceens␣
 ↪columns to have them as list
top_screens
```

[21]:

```
array(['Loan2', 'location', 'Institutions', 'Credit3Container',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth',
       'ProfilePage', 'VerifyCountry', 'Cycle', 'idscreen',
       'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash', 'Loan',
       'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts',
       'Saving8', 'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'Saving7', 'ProfileEducationMajor', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'VerifyIncomeType', 'Saving2', 'Saving6',
       'Saving2Amount', 'Saving5', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'Loan4',
       'ResendToken', 'TransactionList', 'NetworkFailure', 'ListPicker'],
      dtype=object)
```

[22]:

```
# Mapping Screens to Fields
```

```
dataset["screen_list"] = dataset.screen_list.astype(str) + ',' #to use comma to␣
 ↪count or separate as many as number of screens

for sc in top_screens:
    dataset[sc] = dataset.screen_list.str.contains(sc).astype(int)
    dataset['screen_list'] = dataset.screen_list.str.replace(sc+",", "")

#for screens not in top_screens, we catagorized them to other col
dataset['Other'] = dataset.screen_list.str.count(",")
dataset = dataset.drop(columns=['screen_list'])
```

[23]:
```
dataset.head(10)
```

[23]:
```
     user  dayofweek  hour  age  numscreens  minigame  used_premium_feature  \
0  235136          3     2   23          15         0                     0
1  333588          6     1   24          13         0                     0
2  254414          1    19   23           3         0                     1
3  234192          4    16   28          40         0                     0
4   51549          1    18   31          32         0                     0
5   56480          2     9   20          14         0                     0
6  144649          1     2   35           3         0                     0
7  249366          1     3   26          41         0                     1
8  372004          2    14   29          33         1                     1
9  338013          4    18   26          19         0                     0

   enrolled  liked  Loan2  …  Login  ProfileEmploymentLength  WebView  \
0         0      0      1  …      1                        0        0
1         0      0      1  …      0                        0        0
2         0      1      0  …      0                        0        0
3         1      0      0  …      0                        0        0
4         1      1      1  …      0                        0        0
5         1      0      1  …      0                        0        0
6         0      0      0  …      0                        0        0
7         0      0      1  …      0                        0        0
8         0      0      1  …      0                        0        0
9         1      0      1  …      0                        0        0

   SecurityModal  Loan4  ResendToken  TransactionList  NetworkFailure  \
0              0      0            0                0               0
1              0      0            0                0               0
2              0      0            0                0               0
3              0      0            0                0               0
4              0      0            0                0               0
5              0      0            0                0               0
6              0      0            0                0               0
7              0      0            0                1               0
8              0      1            0                0               0
```

```
9              0      0            0                0               0

   ListPicker  Other
0           0      7
1           0      5
2           0      0
3           0      6
4           0     10
5           0      6
6           0      3
7           0      8
8           0     19
9           0     11

[10 rows x 68 columns]
```

[24]:

```
# Funnels are screens that are related, by funneling we remove correlation but␣
 ↪still keep their value. That is to remove highly similar (correlated screens)
savings_screens = ["Saving1",
                    "Saving2",
                    "Saving2Amount",
                    "Saving4",
                    "Saving5",
                    "Saving6",
                    "Saving7",
                    "Saving8",
                    "Saving9",
                    "Saving10"]
dataset["SavingCount"] = dataset[savings_screens].sum(axis=1)
dataset = dataset.drop(columns=savings_screens) #drop cols as they are already␣
 ↪agregated to SavingCount col

cm_screens = ["Credit1",
               "Credit2",
               "Credit3",
               "Credit3Container",
               "Credit3Dashboard"]
dataset["CMCount"] = dataset[cm_screens].sum(axis=1)
dataset = dataset.drop(columns=cm_screens)

cc_screens = ["CC1",
                "CC1Category",
                "CC3"]
dataset["CCCount"] = dataset[cc_screens].sum(axis=1)
dataset = dataset.drop(columns=cc_screens)

loan_screens = ["Loan",
               "Loan2",
               "Loan3",
               "Loan4"]
dataset["LoansCount"] = dataset[loan_screens].sum(axis=1)
dataset = dataset.drop(columns=loan_screens)

#### Saving Results ####
dataset.head()
dataset.describe()
dataset.columns

dataset.to_csv('/content/drive/MyDrive/Data_incubator/data_sets/new_appdata10.
 ↪csv', index = False) # this is clean data to be used
```

[25]:

```
dataset.describe()
```

[25]:

```
                user     dayofweek          hour          age    numscreens  \
count   50000.000000  50000.000000  50000.000000  50000.00000  50000.000000
mean   186889.729900      3.029860     12.557220     31.72436     21.095900
std    107768.520361      2.031997      7.438072     10.80331     15.728812
min        13.000000      0.000000      0.000000     16.00000      1.000000
25%     93526.750000      1.000000      5.000000     24.00000     10.000000
50%    187193.500000      3.000000     14.000000     29.00000     18.000000
75%    279984.250000      5.000000     19.000000     37.00000     28.000000
max    373662.000000      6.000000     23.000000    101.00000    325.000000

           minigame  used_premium_feature      enrolled         liked  \
count  50000.000000          50000.000000  50000.000000  50000.000000
mean       0.107820              0.172020      0.497000      0.165000
std        0.310156              0.377402      0.499996      0.371184
min        0.000000              0.000000      0.000000      0.000000
25%        0.000000              0.000000      0.000000      0.000000
50%        0.000000              0.000000      0.000000      0.000000
75%        0.000000              0.000000      1.000000      0.000000
max        1.000000              1.000000      1.000000      1.000000

           location  …  SecurityModal  ResendToken  TransactionList  \
count  50000.000000  …   50000.000000 50000.000000     50000.000000
mean       0.517760  …       0.014220     0.013340         0.013400
std        0.499689  …       0.118398     0.114727         0.114981
min        0.000000  …       0.000000     0.000000         0.000000
25%        0.000000  …       0.000000     0.000000         0.000000
50%        1.000000  …       0.000000     0.000000         0.000000
75%        1.000000  …       0.000000     0.000000         0.000000
max        1.000000  …       1.000000     1.000000         1.000000

       NetworkFailure   ListPicker          Other   SavingCount      CMCount  \
count    50000.000000  50000.000000  50000.000000  50000.000000  50000.00000
mean         0.008200      0.007580      6.214260      0.365020      0.92776
std          0.090183      0.086733      3.672561      1.405511      1.21751
min          0.000000      0.000000      0.000000      0.000000      0.00000
25%          0.000000      0.000000      3.000000      0.000000      0.00000
50%          0.000000      0.000000      6.000000      0.000000      0.00000
75%          0.000000      0.000000      8.000000      0.000000      1.00000
max          1.000000      1.000000     35.000000     10.000000      5.00000

            CCCount     LoansCount
count  50000.000000  50000.000000
mean       0.176860      0.788400
std        0.612787      0.677462
min        0.000000      0.000000
25%        0.000000      0.000000
50%        0.000000      1.000000
75%        0.000000      1.000000
max        3.000000      3.000000

[8 rows x 50 columns]
```

[26]:
```python
dataset.columns
```

[26]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'location', 'Institutions',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth', 'ProfilePage',
       'VerifyCountry', 'Cycle', 'idscreen', 'Splash', 'RewardsContainer',
       'EditProfile', 'Finances', 'Alerts', 'Leaderboard', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'ResendToken',
       'TransactionList', 'NetworkFailure', 'ListPicker', 'Other',
       'SavingCount', 'CMCount', 'CCCount', 'LoansCount'],
      dtype='object')
```

2 Data Pre-processing

[27]:
```python
# Load the clean saved data
dataset = pd.read_csv('/content/drive/MyDrive/Data_incubator/data_sets/
 ↪new_appdata10.csv')
display(dataset.shape)
display(dataset.head(10))
display(dataset.tail(10))
```

```
(50000, 50)
     user  dayofweek  hour  age  numscreens  minigame  used_premium_feature  \
0  235136          3     2   23          15         0                     0
1  333588          6     1   24          13         0                     0
2  254414          1    19   23           3         0                     1
3  234192          4    16   28          40         0                     0
4   51549          1    18   31          32         0                     0
5   56480          2     9   20          14         0                     0
6  144649          1     2   35           3         0                     0
7  249366          1     3   26          41         0                     1
8  372004          2    14   29          33         1                     1
9  338013          4    18   26          19         0                     0

   enrolled  liked  location  …  SecurityModal  ResendToken  \
0         0      0         0  …              0            0
1         0      0         1  …              0            0
2         0      1         0  …              0            0
3         1      0         1  …              0            0
4         1      1         0  …              0            0
5         1      0         0  …              0            0
6         0      0         0  …              0            0
7         0      0         0  …              0            0
8         0      0         1  …              0            0
9         1      0         1  …              0            0

   TransactionList  NetworkFailure  ListPicker  Other  SavingCount  CMCount  \
0                0               0           0      7            0        0
1                0               0           0      5            0        0
2                0               0           0      0            0        0
3                0               0           0      6            0        3
4                0               0           0     10            0        2
5                0               0           0      6            0        2
6                0               0           0      3            0        0
7                1               0           0      8            0        2
8                0               0           0     19            0        0
9                0               0           0     11            0        0

   CCCount  LoansCount
0        0           1
1        0           1
2        0           1
3        0           1
4        0           1
5        0           1
6        0           0
7        0           1
8        0           3
```

```
9        0           1

[10 rows x 50 columns]
         user  dayofweek  hour  age  numscreens  minigame  \
49990  179308          5    17   20           8         0
49991   85532          4    22   45          30         1
49992   96155          6    15   50          28         0
49993  343026          5     2   28           4         0
49994   90813          0    19   36          25         0
49995  222774          3    13   32          13         0
49996  169179          1     0   35           4         0
49997  302367          2    22   39          25         0
49998  324905          6    12   27          26         0
49999   27047          4     1   25          26         0

       used_premium_feature  enrolled  liked  location  …  SecurityModal  \
49990                     0         1      1         0  …              0
49991                     1         1      0         1  …              0
49992                     0         1      0         1  …              1
49993                     0         0      1         0  …              0
49994                     0         1      0         1  …              0
49995                     0         1      0         0  …              0
49996                     1         0      0         0  …              0
49997                     0         0      0         1  …              0
49998                     0         1      0         1  …              0
49999                     0         0      1         0  …              0

       ResendToken  TransactionList  NetworkFailure  ListPicker  Other  \
49990            0                0               0           0      4
49991            0                0               0           0      3
49992            0                0               0           0      9
49993            0                0               0           0      4
49994            0                1               0           0      9
49995            0                0               0           0      6
49996            0                0               0           0      1
49997            0                0               0           0      6
49998            0                0               0           0     13
49999            0                0               0           0      5

       SavingCount  CMCount  CCCount  LoansCount
49990            0        0        0           0
49991            0        3        0           2
49992            0        1        0           1
49993            0        0        0           0
49994            0        0        0           1
49995            0        2        0           0
49996            0        0        0           0
```

```
49997            0        0        0           0
49998            0        0        0           0
49999            7        0        0           1

[10 rows x 50 columns]
```

[28]:

```
df_summary(dataset)
```

[28]:

```
                      index  Missing Data  Unique Data Data Types
0                      user             0        49874      int64
1                 dayofweek             0            7      int64
2                      hour             0           24      int64
3                       age             0           78      int64
4                numscreens             0          151      int64
5                  minigame             0            2      int64
6      used_premium_feature             0            2      int64
7                  enrolled             0            2      int64
8                     liked             0            2      int64
9                  location             0            2      int64
10             Institutions             0            2      int64
11              VerifyPhone             0            2      int64
12         BankVerification             0            2      int64
13        VerifyDateOfBirth             0            2      int64
14              ProfilePage             0            2      int64
15            VerifyCountry             0            2      int64
16                    Cycle             0            2      int64
17                 idscreen             0            2      int64
18                   Splash             0            2      int64
19         RewardsContainer             0            2      int64
20              EditProfile             0            2      int64
21                 Finances             0            2      int64
22                   Alerts             0            2      int64
23              Leaderboard             0            2      int64
24             VerifyMobile             0            2      int64
25            VerifyHousing             0            2      int64
26             RewardDetail             0            2      int64
27      VerifyHousingAmount             0            2      int64
28     ProfileMaritalStatus             0            2      int64
29         ProfileChildren              0            1      int64
30         ProfileEducation             0            2      int64
31    ProfileEducationMajor             0            2      int64
32                  Rewards             0            2      int64
33              AccountView             0            2      int64
34       VerifyAnnualIncome             0            2      int64
35         VerifyIncomeType             0            2      int64
36          ProfileJobTitle             0            2      int64
37                    Login             0            2      int64
```

```
38  ProfileEmploymentLength          0          2      int64
39                  WebView          0          2      int64
40             SecurityModal         0          2      int64
41              ResendToken          0          2      int64
42           TransactionList         0          2      int64
43            NetworkFailure         0          2      int64
44               ListPicker          0          2      int64
45                    Other          0         32      int64
46              SavingCount          0         11      int64
47                  CMCount          0          6      int64
48                  CCCount          0          4      int64
49               LoansCount          0          4      int64
```

2.1 STEP # 3: Feature selection using Boruta

2.1.1 Why Feature selection?

1. Mitigates overfitting
2. Lowers cost
3. Reduces errors

Boruta is wrapper method that select relevant features. It eliminate arbitrary cutoff criterion. Features compared to stochastic realization of themselves; not to each other.

Let's address the warnings raised by the table 1 above and see if we have to reformat some of the features.

2.1.2 Addressing the warnings

Let's have a look at the disributions for those features that appeared in the warnings.

[30]:
```
# Generate table 1, group by last column (target)
TableOne(dataset, groupby=dataset.columns[-1],
         pval=True,
         dip_test=True,
         normal_test=True,
         tukey_test=True)
```

[30]:
```
                                   Grouped by LoansCount
                                                 Missing                Overall
0                     1                    2                    3 P-Value
n                                                                         50000
17229                 26776                 5341                 654
user, mean (SD)                                                0  186889.7 (107768.5)
187453.4 (107611.0)  186245.1 (107772.4)  187615.0 (108063.6)  192511.5
(109298.7)   0.327
dayofweek, n (%)                0                        0          7515 (15.0)
2568 (14.9)           3998 (14.9)           846 (15.8)           103 (15.7)
<0.001
```

	1			7139 (14.3)	
2453 (14.2)	3815 (14.2)	777 (14.5)		94 (14.4)	
	2			6315 (12.6)	
2241 (13.0)	3379 (12.6)	611 (11.4)		84 (12.8)	
	3			6659 (13.3)	
2485 (14.4)	3541 (13.2)	547 (10.2)		86 (13.1)	
	4			7531 (15.1)	
2523 (14.6)	4002 (14.9)	911 (17.1)		95 (14.5)	
	5			7423 (14.8)	
2413 (14.0)	4055 (15.1)	863 (16.2)		92 (14.1)	
	6			7418 (14.8)	
2546 (14.8)	3986 (14.9)	786 (14.7)		100 (15.3)	
hour, n (%)	0		0	2666 (5.3)	
796 (4.6)	1523 (5.7)	297 (5.6)		50 (7.6)	<0.001
	1			2438 (4.9)	
767 (4.5)	1395 (5.2)	237 (4.4)		39 (6.0)	
	10			849 (1.7)	
303 (1.8)	478 (1.8)	60 (1.1)		8 (1.2)	
	11			1111 (2.2)	
446 (2.6)	552 (2.1)	100 (1.9)		13 (2.0)	
	12			1511 (3.0)	
485 (2.8)	845 (3.2)	160 (3.0)		21 (3.2)	
	13			1946 (3.9)	
762 (4.4)	975 (3.6)	185 (3.5)		24 (3.7)	
	14			2309 (4.6)	
737 (4.3)	1276 (4.8)	264 (4.9)		32 (4.9)	
	15			2989 (6.0)	
1062 (6.2)	1473 (5.5)	409 (7.7)		45 (6.9)	
	16			2790 (5.6)	
843 (4.9)	1552 (5.8)	349 (6.5)		46 (7.0)	
	17			2811 (5.6)	
1071 (6.2)	1374 (5.1)	341 (6.4)		25 (3.8)	
	18			2729 (5.5)	
874 (5.1)	1485 (5.5)	331 (6.2)		39 (6.0)	
	19			2708 (5.4)	
1024 (5.9)	1354 (5.1)	289 (5.4)		41 (6.3)	
	2			2503 (5.0)	
852 (4.9)	1372 (5.1)	245 (4.6)		34 (5.2)	
	20			2818 (5.6)	
840 (4.9)	1582 (5.9)	348 (6.5)		48 (7.3)	
	21			2764 (5.5)	
1061 (6.2)	1360 (5.1)	315 (5.9)		28 (4.3)	
	22			2704 (5.4)	
785 (4.6)	1524 (5.7)	352 (6.6)		43 (6.6)	
	23			2635 (5.3)	
1016 (5.9)	1310 (4.9)	282 (5.3)		27 (4.1)	
	3			2158 (4.3)	

839 (4.9)	1134 (4.2)		160 (3.0)		25 (3.8)	
		4			1933 (3.9)	
626 (3.6)	1132 (4.2)		157 (2.9)		18 (2.8)	
		5			1570 (3.1)	
513 (3.0)	926 (3.5)		114 (2.1)		17 (2.6)	
		6			1283 (2.6)	
444 (2.6)	706 (2.6)		120 (2.2)		13 (2.0)	
		7			1107 (2.2)	
461 (2.7)	557 (2.1)		79 (1.5)		10 (1.5)	
		8			898 (1.8)	
321 (1.9)	495 (1.8)		76 (1.4)		6 (0.9)	
		9			770 (1.5)	
301 (1.7)	396 (1.5)		71 (1.3)		2 (0.3)	
age, n (%)		16		0	191 (0.4)	
77 (0.4)	114 (0.4)					<0.001
		17			696 (1.4)	
258 (1.5)	432 (1.6)		3 (0.1)		3 (0.5)	
		18			1199 (2.4)	
413 (2.4)	745 (2.8)		29 (0.5)		12 (1.8)	
		19			1646 (3.3)	
538 (3.1)	1039 (3.9)		51 (1.0)		18 (2.8)	
		20			1862 (3.7)	
583 (3.4)	1164 (4.3)		93 (1.7)		22 (3.4)	
		21			2130 (4.3)	
699 (4.1)	1276 (4.8)		128 (2.4)		27 (4.1)	
		22			2222 (4.4)	
663 (3.8)	1393 (5.2)		131 (2.5)		35 (5.4)	
		23			2348 (4.7)	
724 (4.2)	1416 (5.3)		176 (3.3)		32 (4.9)	
		24			2298 (4.6)	
705 (4.1)	1364 (5.1)		194 (3.6)		35 (5.4)	
		25			2339 (4.7)	
773 (4.5)	1328 (5.0)		197 (3.7)		41 (6.3)	
		26			2301 (4.6)	
722 (4.2)	1308 (4.9)		235 (4.4)		36 (5.5)	
		27			2221 (4.4)	
734 (4.3)	1230 (4.6)		215 (4.0)		42 (6.4)	
		28			2168 (4.3)	
710 (4.1)	1192 (4.5)		230 (4.3)		36 (5.5)	
		29			2021 (4.0)	
697 (4.0)	1074 (4.0)		230 (4.3)		20 (3.1)	
		30			1851 (3.7)	
595 (3.5)	1011 (3.8)		218 (4.1)		27 (4.1)	
		31			1746 (3.5)	
574 (3.3)	955 (3.6)		188 (3.5)		29 (4.4)	
		32			1578 (3.2)	
513 (3.0)	847 (3.2)		198 (3.7)		20 (3.1)	

33	541 (3.1)	804 (3.0)	200 (3.7)	18 (2.8)	1563 (3.1)
34	535 (3.1)	735 (2.7)	169 (3.2)	18 (2.8)	1457 (2.9)
35	471 (2.7)	608 (2.3)	188 (3.5)	18 (2.8)	1285 (2.6)
36	442 (2.6)	639 (2.4)	183 (3.4)	20 (3.1)	1284 (2.6)
37	385 (2.2)	565 (2.1)	177 (3.3)	15 (2.3)	1142 (2.3)
38	414 (2.4)	526 (2.0)	153 (2.9)	12 (1.8)	1105 (2.2)
39	332 (1.9)	456 (1.7)	169 (3.2)	13 (2.0)	970 (1.9)
40	308 (1.8)	435 (1.6)	144 (2.7)	8 (1.2)	895 (1.8)
41	324 (1.9)	379 (1.4)	134 (2.5)	12 (1.8)	849 (1.7)
42	260 (1.5)	324 (1.2)	127 (2.4)	8 (1.2)	719 (1.4)
43	265 (1.5)	332 (1.2)	121 (2.3)	8 (1.2)	726 (1.5)
44	249 (1.4)	274 (1.0)	92 (1.7)	9 (1.4)	624 (1.2)
45	201 (1.2)	307 (1.1)	107 (2.0)	5 (0.8)	620 (1.2)
46	197 (1.1)	246 (0.9)	110 (2.1)	7 (1.1)	560 (1.1)
47	218 (1.3)	240 (0.9)	83 (1.6)	8 (1.2)	549 (1.1)
48	205 (1.2)	229 (0.9)	83 (1.6)	4 (0.6)	521 (1.0)
49	192 (1.1)	191 (0.7)	65 (1.2)	5 (0.8)	453 (0.9)
50	165 (1.0)	195 (0.7)	62 (1.2)	2 (0.3)	424 (0.8)
51	147 (0.9)	142 (0.5)	68 (1.3)	3 (0.5)	360 (0.7)
52	141 (0.8)	128 (0.5)	36 (0.7)	3 (0.5)	308 (0.6)
53	137 (0.8)	138 (0.5)	54 (1.0)	2 (0.3)	331 (0.7)
54	107 (0.6)	104 (0.4)	38 (0.7)	2 (0.3)	251 (0.5)
55	140 (0.8)	109 (0.4)	44 (0.8)	2 (0.3)	295 (0.6)
56					253 (0.5)

106 (0.6)	108 (0.4)		36 (0.7)	3 (0.5)	
		57			212 (0.4)
91 (0.5)	89 (0.3)		30 (0.6)	2 (0.3)	
		58			199 (0.4)
101 (0.6)	73 (0.3)		21 (0.4)	4 (0.6)	
		59			165 (0.3)
84 (0.5)	64 (0.2)		16 (0.3)	1 (0.2)	
		60			148 (0.3)
66 (0.4)	61 (0.2)		18 (0.3)	3 (0.5)	
		61			127 (0.3)
63 (0.4)	48 (0.2)		15 (0.3)	1 (0.2)	
		62			118 (0.2)
56 (0.3)	41 (0.2)		21 (0.4)		
		63			96 (0.2)
47 (0.3)	36 (0.1)		12 (0.2)	1 (0.2)	
		64			82 (0.2)
33 (0.2)	42 (0.2)		7 (0.1)		
		65			89 (0.2)
40 (0.2)	36 (0.1)		13 (0.2)		
		66			66 (0.1)
31 (0.2)	32 (0.1)		3 (0.1)		
		67			52 (0.1)
30 (0.2)	19 (0.1)		2 (0.0)	1 (0.2)	
		68			48 (0.1)
22 (0.1)	21 (0.1)		5 (0.1)		
		69			41 (0.1)
25 (0.1)	12 (0.0)		4 (0.1)		
		70			29 (0.1)
13 (0.1)	14 (0.1)		2 (0.0)		
		71			29 (0.1)
7 (0.0)	21 (0.1)		1 (0.0)		
		72			24 (0.0)
12 (0.1)	9 (0.0)		3 (0.1)		
		73			14 (0.0)
6 (0.0)	6 (0.0)		2 (0.0)		
		74			20 (0.0)
8 (0.0)	10 (0.0)		1 (0.0)	1 (0.2)	
		75			16 (0.0)
7 (0.0)	8 (0.0)		1 (0.0)		
		76			9 (0.0)
6 (0.0)	2 (0.0)		1 (0.0)		
		77			5 (0.0)
1 (0.0)	4 (0.0)				
		78			8 (0.0)
2 (0.0)	6 (0.0)				
		79			1 (0.0)
1 (0.0)					

	80		4 (0.0)	2 (0.0)	1 (0.0)	1 (0.0)		
	81		7 (0.0)	3 (0.0)	4 (0.0)			
	82		3 (0.0)	1 (0.0)	2 (0.0)			
	84		3 (0.0)	3 (0.0)				
	85		3 (0.0)	1 (0.0)	2 (0.0)			
	86		3 (0.0)	1 (0.0)	2 (0.0)			
	87		5 (0.0)	3 (0.0)	2 (0.0)			
	88		1 (0.0)	1 (0.0)				
	90		3 (0.0)	1 (0.0)	2 (0.0)			
	98		1 (0.0)	1 (0.0)				
	101		1 (0.0)	1 (0.0)				
	83		3 (0.0)	3 (0.0)				
	89		2 (0.0)	1 (0.0)	1 (0.0)			
	100		2 (0.0)	2 (0.0)				
numscreens, n (%)	1	0	898 (1.8)	820 (4.8)	78 (0.3)			<0.001
	10		1680 (3.4)	747 (4.3)	728 (2.7)	201 (3.8)	4 (0.6)	
	103		3 (0.0)	1 (0.0)	1 (0.0)	1 (0.0)		
	106		4 (0.0)	1 (0.0)	2 (0.0)		1 (0.2)	
	107		5 (0.0)	1 (0.0)	3 (0.0)	1 (0.0)		
	11		1569 (3.1)	701 (4.1)	647 (2.4)	212 (4.0)	9 (1.4)	
	114		4 (0.0)	1 (0.0)	3 (0.0)			
	12		1648 (3.3)	738 (4.3)	712 (2.7)	195 (3.7)	3 (0.5)	
	126		2 (0.0)	1 (0.0)			1 (0.2)	
	13		1621 (3.2)					

684 (4.0)	759 (2.8)		170 (3.2)	8 (1.2)
		133		2 (0.0)
1 (0.0)				1 (0.2)
		14		1650 (3.3)
668 (3.9)	789 (2.9)		186 (3.5)	7 (1.1)
		15		1686 (3.4)
693 (4.0)	810 (3.0)		171 (3.2)	12 (1.8)
		16		1681 (3.4)
640 (3.7)	860 (3.2)		170 (3.2)	11 (1.7)
		17		1602 (3.2)
611 (3.5)	820 (3.1)		157 (2.9)	14 (2.1)
		18		1572 (3.1)
500 (2.9)	905 (3.4)		163 (3.1)	4 (0.6)
		19		1467 (2.9)
493 (2.9)	830 (3.1)		134 (2.5)	10 (1.5)
		2		855 (1.7)
669 (3.9)	163 (0.6)		23 (0.4)	
		20		1439 (2.9)
458 (2.7)	832 (3.1)		133 (2.5)	16 (2.4)
		21		1334 (2.7)
352 (2.0)	814 (3.0)		151 (2.8)	17 (2.6)
		22		1227 (2.5)
295 (1.7)	784 (2.9)		136 (2.5)	12 (1.8)
		23		1212 (2.4)
253 (1.5)	810 (3.0)		128 (2.4)	21 (3.2)
		24		1076 (2.2)
213 (1.2)	700 (2.6)		144 (2.7)	19 (2.9)
		25		1047 (2.1)
196 (1.1)	720 (2.7)		116 (2.2)	15 (2.3)
		26		947 (1.9)
167 (1.0)	662 (2.5)		105 (2.0)	13 (2.0)
		27		942 (1.9)
151 (0.9)	654 (2.4)		114 (2.1)	23 (3.5)
		28		910 (1.8)
138 (0.8)	667 (2.5)		88 (1.6)	17 (2.6)
		29		856 (1.7)
138 (0.8)	612 (2.3)		88 (1.6)	18 (2.8)
		3		1051 (2.1)
749 (4.3)	240 (0.9)		62 (1.2)	
		30		778 (1.6)
115 (0.7)	552 (2.1)		89 (1.7)	22 (3.4)
		31		692 (1.4)
92 (0.5)	514 (1.9)		65 (1.2)	21 (3.2)
		32		616 (1.2)
78 (0.5)	452 (1.7)		72 (1.3)	14 (2.1)
		33		569 (1.1)
72 (0.4)	416 (1.6)		62 (1.2)	19 (2.9)

		34		578 (1.2)
72 (0.4)	420 (1.6)		69 (1.3)	17 (2.6)
		35		557 (1.1)
58 (0.3)	419 (1.6)		66 (1.2)	14 (2.1)
		36		490 (1.0)
68 (0.4)	353 (1.3)		54 (1.0)	15 (2.3)
		37		481 (1.0)
51 (0.3)	345 (1.3)		61 (1.1)	24 (3.7)
		38		439 (0.9)
60 (0.3)	310 (1.2)		47 (0.9)	22 (3.4)
		39		371 (0.7)
37 (0.2)	275 (1.0)		44 (0.8)	15 (2.3)
		4		1307 (2.6)
889 (5.2)	322 (1.2)		96 (1.8)	
		40		383 (0.8)
48 (0.3)	280 (1.0)		40 (0.7)	15 (2.3)
		41		368 (0.7)
54 (0.3)	271 (1.0)		34 (0.6)	9 (1.4)
		42		314 (0.6)
33 (0.2)	230 (0.9)		43 (0.8)	8 (1.2)
		43		336 (0.7)
28 (0.2)	255 (1.0)		38 (0.7)	15 (2.3)
		44		297 (0.6)
34 (0.2)	209 (0.8)		38 (0.7)	16 (2.4)
		45		311 (0.6)
35 (0.2)	231 (0.9)		32 (0.6)	13 (2.0)
		46		256 (0.5)
19 (0.1)	202 (0.8)		27 (0.5)	8 (1.2)
		47		201 (0.4)
23 (0.1)	143 (0.5)		29 (0.5)	6 (0.9)
		48		199 (0.4)
18 (0.1)	145 (0.5)		30 (0.6)	6 (0.9)
		49		208 (0.4)
21 (0.1)	153 (0.6)		27 (0.5)	7 (1.1)
		5		1309 (2.6)
807 (4.7)	367 (1.4)		132 (2.5)	3 (0.5)
		50		186 (0.4)
18 (0.1)	138 (0.5)		26 (0.5)	4 (0.6)
		51		162 (0.3)
15 (0.1)	119 (0.4)		24 (0.4)	4 (0.6)
		52		183 (0.4)
16 (0.1)	127 (0.5)		35 (0.7)	5 (0.8)
		53		147 (0.3)
11 (0.1)	109 (0.4)		19 (0.4)	8 (1.2)
		54		126 (0.3)
6 (0.0)	96 (0.4)		20 (0.4)	4 (0.6)
		55		129 (0.3)

17 (0.1)	95 (0.4)		14 (0.3)	3 (0.5)	
		56			126 (0.3)
10 (0.1)	98 (0.4)		14 (0.3)	4 (0.6)	
		57			136 (0.3)
9 (0.1)	101 (0.4)		19 (0.4)	7 (1.1)	
		58			118 (0.2)
8 (0.0)	92 (0.3)		13 (0.2)	5 (0.8)	
		59			107 (0.2)
13 (0.1)	72 (0.3)		18 (0.3)	4 (0.6)	
		6			1478 (3.0)
878 (5.1)	480 (1.8)		119 (2.2)	1 (0.2)	
		60			89 (0.2)
8 (0.0)	62 (0.2)		17 (0.3)	2 (0.3)	
		61			93 (0.2)
5 (0.0)	70 (0.3)		12 (0.2)	6 (0.9)	
		62			85 (0.2)
8 (0.0)	60 (0.2)		15 (0.3)	2 (0.3)	
		63			80 (0.2)
7 (0.0)	61 (0.2)		9 (0.2)	3 (0.5)	
		64			64 (0.1)
9 (0.1)	43 (0.2)		9 (0.2)	3 (0.5)	
		65			71 (0.1)
2 (0.0)	53 (0.2)		15 (0.3)	1 (0.2)	
		66			64 (0.1)
5 (0.0)	45 (0.2)		10 (0.2)	4 (0.6)	
		67			44 (0.1)
3 (0.0)	35 (0.1)		4 (0.1)	2 (0.3)	
		68			40 (0.1)
5 (0.0)	30 (0.1)		5 (0.1)		
		69			56 (0.1)
4 (0.0)	43 (0.2)		6 (0.1)	3 (0.5)	
		7			1576 (3.2)
782 (4.5)	595 (2.2)		197 (3.7)	2 (0.3)	
		70			47 (0.1)
4 (0.0)	37 (0.1)		5 (0.1)	1 (0.2)	
		71			49 (0.1)
6 (0.0)	34 (0.1)		9 (0.2)		
		73			48 (0.1)
6 (0.0)	32 (0.1)		5 (0.1)	5 (0.8)	
		75			38 (0.1)
4 (0.0)	30 (0.1)		4 (0.1)		
		76			30 (0.1)
2 (0.0)	20 (0.1)		8 (0.1)		
		77			24 (0.0)
3 (0.0)	17 (0.1)		4 (0.1)		
		78			26 (0.1)
1 (0.0)	21 (0.1)		4 (0.1)		

		79			15 (0.0)
2 (0.0)	10 (0.0)		3 (0.1)		
		8			1570 (3.1)
775 (4.5)	620 (2.3)		168 (3.1)	7 (1.1)	
		80			25 (0.1)
1 (0.0)	18 (0.1)		6 (0.1)		
		81			20 (0.0)
1 (0.0)	14 (0.1)		5 (0.1)		
		82			12 (0.0)
1 (0.0)	6 (0.0)		3 (0.1)	2 (0.3)	
		87			11 (0.0)
1 (0.0)	5 (0.0)		3 (0.1)	2 (0.3)	
		88			12 (0.0)
2 (0.0)	5 (0.0)		5 (0.1)		
		89			8 (0.0)
1 (0.0)	6 (0.0)		1 (0.0)		
		9			1568 (3.1)
785 (4.6)	592 (2.2)		186 (3.5)	5 (0.8)	
		91			10 (0.0)
1 (0.0)	8 (0.0)		1 (0.0)		
		93			9 (0.0)
1 (0.0)	7 (0.0)		1 (0.0)		
		96			7 (0.0)
1 (0.0)	6 (0.0)				
		100			4 (0.0)
4 (0.0)					
		102			5 (0.0)
5 (0.0)					
		104			2 (0.0)
2 (0.0)					
		108			4 (0.0)
2 (0.0)	2 (0.0)				
		109			4 (0.0)
4 (0.0)					
		110			1 (0.0)
1 (0.0)					
		111			1 (0.0)
1 (0.0)					
		113			1 (0.0)
1 (0.0)					
		115			2 (0.0)
1 (0.0)	1 (0.0)				
		116			1 (0.0)
1 (0.0)					
		117			1 (0.0)
1 (0.0)					
		120			1 (0.0)

1 (0.0)				
		121		2 (0.0)
2 (0.0)				
		122		2 (0.0)
2 (0.0)				
		123		1 (0.0)
1 (0.0)				
		125		2 (0.0)
1 (0.0)			1 (0.2)	
		127		2 (0.0)
1 (0.0)	1 (0.0)			
		129		1 (0.0)
1 (0.0)				
		130		1 (0.0)
1 (0.0)				
		132		1 (0.0)
1 (0.0)				
		134		1 (0.0)
1 (0.0)				
		136		1 (0.0)
1 (0.0)				
		141		1 (0.0)
1 (0.0)				
		153		1 (0.0)
1 (0.0)				
		165		1 (0.0)
1 (0.0)				
		185		1 (0.0)
1 (0.0)				
		189		1 (0.0)
1 (0.0)				
		192		2 (0.0)
2 (0.0)				
		200		1 (0.0)
1 (0.0)				
		234		1 (0.0)
1 (0.0)				
		247		1 (0.0)
1 (0.0)				
		72		41 (0.1)
32 (0.1)	8 (0.1)		1 (0.2)	
		74		46 (0.1)
39 (0.1)	6 (0.1)		1 (0.2)	
		83		18 (0.0)
14 (0.1)	4 (0.1)			
		84		8 (0.0)
6 (0.0)	2 (0.0)			

		85		15 (0.0)
8 (0.0)	6 (0.1)		1 (0.2)	
		86		17 (0.0)
12 (0.0)	4 (0.1)		1 (0.2)	
		90		11 (0.0)
10 (0.0)			1 (0.2)	
		92		7 (0.0)
7 (0.0)				
		94		9 (0.0)
5 (0.0)	4 (0.1)			
		95		6 (0.0)
4 (0.0)	2 (0.0)			
		97		6 (0.0)
5 (0.0)	1 (0.0)			
		98		6 (0.0)
5 (0.0)	1 (0.0)			
		99		4 (0.0)
2 (0.0)	2 (0.0)			
		101		1 (0.0)
1 (0.0)				
		105		1 (0.0)
1 (0.0)				
		112		1 (0.0)
1 (0.0)				
		119		1 (0.0)
1 (0.0)				
		128		2 (0.0)
2 (0.0)				
		137		1 (0.0)
1 (0.0)				
		144		1 (0.0)
1 (0.0)				
		148		1 (0.0)
1 (0.0)				
		162		1 (0.0)
1 (0.0)				
		187		1 (0.0)
1 (0.0)				
		216		1 (0.0)
1 (0.0)				
		118		1 (0.0)
1 (0.2)				
		179		1 (0.0)
1 (0.2)				
		243		1 (0.0)
1 (0.2)				
		325		1 (0.0)

```
1 (0.2)
minigame, n (%)                0                       0         44609 (89.2)
15212 (88.3)         24025 (89.7)          4844 (90.7)           528 (80.7)
<0.001
                               1                                  5391 (10.8)
2017 (11.7)          2751 (10.3)            497 (9.3)           126 (19.3)
used_premium_feature, n (%)    0                       0         41399 (82.8)
15011 (87.1)         22753 (85.0)          3334 (62.4)           301 (46.0)
<0.001
                               1                                  8601 (17.2)
2218 (12.9)          4023 (15.0)          2007 (37.6)           353 (54.0)
enrolled, n (%)                0                       0         25150 (50.3)
8465 (49.1)         11690 (43.7)          4644 (87.0)           351 (53.7)
<0.001
                               1                                 24850 (49.7)
8764 (50.9)         15086 (56.3)           697 (13.0)           303 (46.3)
liked, n (%)                   0                       0         41750 (83.5)
14467 (84.0)         22274 (83.2)          4465 (83.6)           544 (83.2)
0.191
                               1                                  8250 (16.5)
2762 (16.0)          4502 (16.8)           876 (16.4)           110 (16.8)
location, n (%)                0                       0         24112 (48.2)
8963 (52.0)         10622 (39.7)          4311 (80.7)           216 (33.0)
<0.001
                               1                                 25888 (51.8)
8266 (48.0)         16154 (60.3)          1030 (19.3)           438 (67.0)
Institutions, n (%)            0                       0         35317 (70.6)
15308 (88.9)         16613 (62.0)          2970 (55.6)           426 (65.1)
<0.001
                               1                                 14683 (29.4)
1921 (11.1)         10163 (38.0)          2371 (44.4)           228 (34.9)
VerifyPhone, n (%)             0                       0         23770 (47.5)
8682 (50.4)         10287 (38.4)          4490 (84.1)           311 (47.6)
<0.001
                               1                                 26230 (52.5)
8547 (49.6)         16489 (61.6)           851 (15.9)           343 (52.4)
BankVerification, n (%)        0                       0         34023 (68.0)
12266 (71.2)         16891 (63.1)          4591 (86.0)           275 (42.0)
<0.001
                               1                                 15977 (32.0)
4963 (28.8)          9885 (36.9)           750 (14.0)           379 (58.0)
VerifyDateOfBirth, n (%)       0                       0         26326 (52.7)
9279 (53.9)         12146 (45.4)          4546 (85.1)           355 (54.3)
<0.001
                               1                                 23674 (47.3)
7950 (46.1)         14630 (54.6)           795 (14.9)           299 (45.7)
ProfilePage, n (%)             0                       0         42098 (84.2)
```

```
15199 (88.2)         22424 (83.7)          3953 (74.0)           522 (79.8)
<0.001
                               1                                  7902 (15.8)
2030 (11.8)          4352 (16.3)          1388 (26.0)           132 (20.2)
VerifyCountry, n (%)           0                       0         28842 (57.7)
10900 (63.3)         13062 (48.8)          4692 (87.8)           188 (28.7)
<0.001
                               1                                 21158 (42.3)
6329 (36.7)         13714 (51.2)           649 (12.2)           466 (71.3)
Cycle, n (%)                   0                       0         31757 (63.5)
11544 (67.0)         16421 (61.3)          3421 (64.1)           371 (56.7)
<0.001
                               1                                 18243 (36.5)
5685 (33.0)         10355 (38.7)          1920 (35.9)           283 (43.3)
idscreen, n (%)                0                       0         32965 (65.9)
11758 (68.2)         16418 (61.3)          4350 (81.4)           439 (67.1)
<0.001
                               1                                 17035 (34.1)
5471 (31.8)         10358 (38.7)           991 (18.6)           215 (32.9)
Splash, n (%)                  0                       0         39962 (79.9)
14726 (85.5)         21404 (79.9)          3410 (63.8)           422 (64.5)
<0.001
                               1                                 10038 (20.1)
2503 (14.5)          5372 (20.1)          1931 (36.2)           232 (35.5)
RewardsContainer, n (%)        0                       0         45800 (91.6)
16254 (94.3)         24310 (90.8)          4627 (86.6)           609 (93.1)
<0.001
                               1                                   4200 (8.4)
975 (5.7)           2466 (9.2)           714 (13.4)             45 (6.9)
EditProfile, n (%)             0                       0         47551 (95.1)
16514 (95.9)         25381 (94.8)          5026 (94.1)           630 (96.3)
<0.001
                               1                                   2449 (4.9)
715 (4.1)           1395 (5.2)            315 (5.9)             24 (3.7)
Finances, n (%)                0                       0         46173 (92.3)
16895 (98.1)         23938 (89.4)          4747 (88.9)           593 (90.7)
<0.001
                               1                                   3827 (7.7)
334 (1.9)          2838 (10.6)           594 (11.1)             61 (9.3)
Alerts, n (%)                  0                       0         46314 (92.6)
16696 (96.9)         25059 (93.6)          3977 (74.5)           582 (89.0)
<0.001
                               1                                   3686 (7.4)
533 (3.1)           1717 (6.4)          1364 (25.5)            72 (11.0)
Leaderboard, n (%)             0                       0         47424 (94.8)
16793 (97.5)         25094 (93.7)          4920 (92.1)           617 (94.3)
<0.001
```

```
                               1                        2576 (5.2)
436 (2.5)          1682 (6.3)           421 (7.9)             37 (5.7)
VerifyMobile, n (%)            0                       0       47307 (94.6)
16494 (95.7)        24956 (93.2)         5226 (97.8)           631 (96.5)
<0.001
                               1                                2693 (5.4)
735 (4.3)          1820 (6.8)           115 (2.2)             23 (3.5)
VerifyHousing, n (%)           0                       0       47746 (95.5)
16582 (96.2)        25441 (95.0)         5093 (95.4)           630 (96.3)
<0.001
                               1                                2254 (4.5)
647 (3.8)          1335 (5.0)           248 (4.6)             24 (3.7)
RewardDetail, n (%)            0                       0       48454 (96.9)
16827 (97.7)        25880 (96.7)         5111 (95.7)           636 (97.2)
<0.001
                               1                                1546 (3.1)
402 (2.3)           896 (3.3)           230 (4.3)             18 (2.8)
VerifyHousingAmount, n (%)     0                       0       47929 (95.9)
16645 (96.6)        25534 (95.4)         5115 (95.8)           635 (97.1)
<0.001
                               1                                2071 (4.1)
584 (3.4)          1242 (4.6)           226 (4.2)             19 (2.9)
ProfileMaritalStatus, n (%)    0                       0       47622 (95.2)
16597 (96.3)        25359 (94.7)         5036 (94.3)           630 (96.3)
<0.001
                               1                                2378 (4.8)
632 (3.7)          1417 (5.3)           305 (5.7)             24 (3.7)
ProfileChildren , n (%)        0                       0      50000 (100.0)
17229 (100.0)       26776 (100.0)        5341 (100.0)          654 (100.0)
1.000
ProfileEducation, n (%)        0                       0       47725 (95.5)
16589 (96.3)        25457 (95.1)         5045 (94.5)           634 (96.9)
<0.001
                               1                                2275 (4.5)
640 (3.7)          1319 (4.9)           296 (5.5)             20 (3.1)
ProfileEducationMajor, n (%)   0                       0       47947 (95.9)
16671 (96.8)        25566 (95.5)         5074 (95.0)           636 (97.2)
<0.001
                               1                                2053 (4.1)
558 (3.2)          1210 (4.5)           267 (5.0)             18 (2.8)
Rewards, n (%)                 0                       0       48779 (97.6)
16937 (98.3)        26065 (97.3)         5139 (96.2)           638 (97.6)
<0.001
                               1                                1221 (2.4)
292 (1.7)           711 (2.7)           202 (3.8)             16 (2.4)
AccountView, n (%)             0                       0       48689 (97.4)
17143 (99.5)        26194 (97.8)         4773 (89.4)           579 (88.5)
```

```
<0.001
                               1                                   1311 (2.6)
86 (0.5)            582 (2.2)           568 (10.6)            75 (11.5)
VerifyAnnualIncome, n (%)      0                        0        48821 (97.6)
16853 (97.8)         26123 (97.6)          5211 (97.6)           634 (96.9)
0.204
                               1                                   1179 (2.4)
376 (2.2)            653 (2.4)            130 (2.4)             20 (3.1)
VerifyIncomeType, n (%)        0                        0        48412 (96.8)
16732 (97.1)         25865 (96.6)          5187 (97.1)           628 (96.0)
0.007
                               1                                   1588 (3.2)
497 (2.9)            911 (3.4)            154 (2.9)             26 (4.0)
ProfileJobTitle, n (%)         0                        0        48877 (97.8)
16871 (97.9)         26109 (97.5)          5250 (98.3)           647 (98.9)
<0.001
                               1                                   1123 (2.2)
358 (2.1)            667 (2.5)             91 (1.7)              7 (1.1)
Login, n (%)                   0                        0        48510 (97.0)
16832 (97.7)         25929 (96.8)          5108 (95.6)           641 (98.0)
<0.001
                               1                                   1490 (3.0)
397 (2.3)            847 (3.2)            233 (4.4)             13 (2.0)
ProfileEmploymentLength, n (%) 0                        0        48942 (97.9)
16901 (98.1)         26144 (97.6)          5248 (98.3)           649 (99.2)
<0.001
                               1                                   1058 (2.1)
328 (1.9)            632 (2.4)             93 (1.7)              5 (0.8)
WebView, n (%)                 0                        0        45172 (90.3)
16341 (94.8)         25184 (94.1)          3186 (59.7)           461 (70.5)
<0.001
                               1                                   4828 (9.7)
888 (5.2)           1592 (5.9)          2155 (40.3)           193 (29.5)
SecurityModal, n (%)           0                        0        49289 (98.6)
16962 (98.5)         26366 (98.5)          5311 (99.4)           650 (99.4)
<0.001
                               1                                    711 (1.4)
267 (1.5)            410 (1.5)             30 (0.6)              4 (0.6)
ResendToken, n (%)             0                        0        49333 (98.7)
17073 (99.1)         26307 (98.2)          5302 (99.3)           651 (99.5)
<0.001
                               1                                    667 (1.3)
156 (0.9)            469 (1.8)             39 (0.7)              3 (0.5)
TransactionList, n (%)         0                        0        49330 (98.7)
17113 (99.3)         26534 (99.1)          5060 (94.7)           623 (95.3)
<0.001
                               1                                    670 (1.3)
```

```
116 (0.7)           242 (0.9)           281 (5.3)           31 (4.7)
NetworkFailure, n (%)         0                       0        49590 (99.2)
17077 (99.1)        26565 (99.2)         5297 (99.2)         651 (99.5)
0.530
                              1                                  410 (0.8)
152 (0.9)           211 (0.8)            44 (0.8)            3 (0.5)
ListPicker, n (%)             0                       0        49621 (99.2)
17162 (99.6)        26566 (99.2)         5250 (98.3)         643 (98.3)
<0.001
                              1                                  379 (0.8)
67 (0.4)           210 (0.8)            91 (1.7)           11 (1.7)
Other, n (%)                  0                       0          775 (1.6)
504 (2.9)           204 (0.8)            67 (1.3)                  <0.001
                              1                                 2961 (5.9)
1625 (9.4)          1012 (3.8)           318 (6.0)            6 (0.9)
                              10                                2313 (4.6)
391 (2.3)          1804 (6.7)           102 (1.9)           16 (2.4)
                              11                                2205 (4.4)
515 (3.0)          1580 (5.9)            86 (1.6)           24 (3.7)
                              12                                1600 (3.2)
320 (1.9)          1166 (4.4)            88 (1.6)           26 (4.0)
                              13                                1301 (2.6)
197 (1.1)           983 (3.7)            75 (1.4)           46 (7.0)
                              14                                 774 (1.5)
75 (0.4)           592 (2.2)            70 (1.3)           37 (5.7)
                              15                                 360 (0.7)
24 (0.1)           254 (0.9)            33 (0.6)           49 (7.5)
                              16                                 191 (0.4)
15 (0.1)           108 (0.4)            21 (0.4)           47 (7.2)
                              17                                 109 (0.2)
5 (0.0)            42 (0.2)            18 (0.3)           44 (6.7)
                              18                                  76 (0.2)
1 (0.0)            19 (0.1)            14 (0.3)           42 (6.4)
                              2                                 4132 (8.3)
1919 (11.1)          1402 (5.2)          796 (14.9)           15 (2.3)
                              3                                 4830 (9.7)
2106 (12.2)          1502 (5.6)         1195 (22.4)          27 (4.1)
                              4                                5696 (11.4)
2524 (14.6)          2245 (8.4)          902 (16.9)          25 (3.8)
                              5                                5737 (11.5)
1995 (11.6)         3115 (11.6)          600 (11.2)          27 (4.1)
                              6                                 4739 (9.5)
1285 (7.5)         3028 (11.3)          409 (7.7)          17 (2.6)
                              7                                 4455 (8.9)
1318 (7.6)         2872 (10.7)          250 (4.7)          15 (2.3)
                              8                                 4368 (8.7)
1755 (10.2)          2431 (9.1)           165 (3.1)           17 (2.6)
```

```
                               9                                   3175 (6.3)
655 (3.8)           2405 (9.0)             99 (1.9)            16 (2.4)
                               19                                    54 (0.1)
7 (0.0)             11 (0.2)             36 (5.5)
                               20                                    38 (0.1)
5 (0.0)              9 (0.2)             24 (3.7)
                               21                                    29 (0.1)
4 (0.1)             25 (3.8)
                               22                                    21 (0.0)
2 (0.0)             19 (2.9)
                               24                                    13 (0.0)
4 (0.1)              9 (1.4)
                               25                                    12 (0.0)
2 (0.0)             10 (1.5)
                               26                                     9 (0.0)
1 (0.0)              8 (1.2)
                               23                                    18 (0.0)
18 (2.8)
                               27                                     3 (0.0)
3 (0.5)
                               28                                     1 (0.0)
1 (0.2)
                               29                                     1 (0.0)
1 (0.2)
                               30                                     3 (0.0)
3 (0.5)
                               35                                     1 (0.0)
1 (0.2)
SavingCount, n (%)             0                       0         45737 (91.5)
15944 (92.5)         24308 (90.8)          4867 (91.1)           618 (94.5)
<0.001
                               1                                    860 (1.7)
289 (1.7)            479 (1.8)             87 (1.6)              5 (0.8)
                               10                                    89 (0.2)
14 (0.1)             57 (0.2)             15 (0.3)              3 (0.5)
                               2                                    323 (0.6)
125 (0.7)            166 (0.6)             29 (0.5)              3 (0.5)
                               3                                    640 (1.3)
183 (1.1)            344 (1.3)            100 (1.9)             13 (2.0)
                               4                                    671 (1.3)
226 (1.3)            385 (1.4)             54 (1.0)              6 (0.9)
                               5                                    376 (0.8)
112 (0.7)            216 (0.8)             46 (0.9)              2 (0.3)
                               6                                    450 (0.9)
137 (0.8)            273 (1.0)             39 (0.7)              1 (0.2)
                               7                                    183 (0.4)
48 (0.3)            109 (0.4)             26 (0.5)
```

```
                               8                                    649 (1.3)
146 (0.8)            425 (1.6)             75 (1.4)              3 (0.5)
                               9                                     22 (0.0)
5 (0.0)             14 (0.1)              3 (0.1)
CMCount, n (%)                 0                        0        26196 (52.4)
12587 (73.1)         11138 (41.6)          2108 (39.5)           363 (55.5)
<0.001
                               1                                 11306 (22.6)
1825 (10.6)          7918 (29.6)          1410 (26.4)           153 (23.4)
                               2                                  5122 (10.2)
1252 (7.3)          2776 (10.4)          1035 (19.4)             59 (9.0)
                               3                                   4668 (9.3)
1098 (6.4)          2941 (11.0)           573 (10.7)             56 (8.6)
                               4                                   2706 (5.4)
467 (2.7)           2002 (7.5)            214 (4.0)             23 (3.5)
                               5                                      2 (0.0)
1 (0.0)              1 (0.0)
CCCount, n (%)                 0                        0        45511 (91.0)
16433 (95.4)         23916 (89.3)          4562 (85.4)           600 (91.7)
<0.001
                               1                                   1529 (3.1)
271 (1.6)            902 (3.4)            331 (6.2)             25 (3.8)
                               2                                   1566 (3.1)
276 (1.6)           1003 (3.7)            266 (5.0)             21 (3.2)
                               3                                   1394 (2.8)
249 (1.4)            902 (3.6)            182 (3.4)              8 (1.2)
[1] Chi-squared tests for the following variables may be invalid due to the low
number of observations: age, numscreens, ListPicker, Other, SavingCount,
CMCount.
[2] Normality test reports non-normal
                                  distributions for: user.
```

[31]:

```
#80-20 partition
df_test = dataset.sample(frac=0.2)
df_train = dataset.drop(df_test.index)
display(df_train.shape)
display(df_test.shape)
```

```
(40000, 50)
```

```
(10000, 50)
```

[32]:

```
# get predictors and labels
X = np.array(df_train.drop('enrolled', axis=1))
y = np.array(df_train['enrolled'])

# define random forest classifier for boruta
forest = RandomForestClassifier(n_jobs=-1, class_weight='balanced', max_depth=3)
```

```python
# forest.fit(X, y) bc no need to fit

# define Boruta feature selection method
feat_selector = BorutaPy(forest, n_estimators='auto', verbose=0,␣
 ↪random_state=42)

# find all relevant features
feat_selector.fit(X, y)

# zip by names, ranks, and decisions in a single iterable
feature_ranks = list(zip(dataset.columns.drop('enrolled'),
                         feat_selector.ranking_,
                         feat_selector.support_))

# iterate through and print out the results
for feat in feature_ranks:
    display('Feature: {:<25} Rank: {},  Keep: {}'.format(feat[0], feat[1],␣
 ↪feat[2]))
```

```
'Feature: user                      Rank: 4,  Keep: False'

'Feature: dayofweek                 Rank: 7,  Keep: False'

'Feature: hour                      Rank: 1,  Keep: True'

'Feature: age                       Rank: 1,  Keep: True'

'Feature: numscreens                Rank: 1,  Keep: True'

'Feature: minigame                  Rank: 1,  Keep: True'

'Feature: used_premium_feature      Rank: 1,  Keep: True'

'Feature: liked                     Rank: 16,  Keep: False'

'Feature: location                  Rank: 1,  Keep: True'

'Feature: Institutions              Rank: 1,  Keep: True'

'Feature: VerifyPhone               Rank: 1,  Keep: True'

'Feature: BankVerification          Rank: 1,  Keep: True'

'Feature: VerifyDateOfBirth         Rank: 1,  Keep: True'

'Feature: ProfilePage               Rank: 1,  Keep: True'

'Feature: VerifyCountry             Rank: 1,  Keep: True'

'Feature: Cycle                     Rank: 1,  Keep: True'

'Feature: idscreen                  Rank: 1,  Keep: True'

'Feature: Splash                    Rank: 1,  Keep: True'

'Feature: RewardsContainer          Rank: 7,  Keep: False'
```

```
'Feature: EditProfile                Rank: 10,  Keep: False'
'Feature: Finances                   Rank: 1,  Keep: True'
'Feature: Alerts                     Rank: 1,  Keep: True'
'Feature: Leaderboard                Rank: 13,  Keep: False'
'Feature: VerifyMobile               Rank: 1,  Keep: True'
'Feature: VerifyHousing              Rank: 1,  Keep: True'
'Feature: RewardDetail               Rank: 3,  Keep: False'
'Feature: VerifyHousingAmount        Rank: 1,  Keep: True'
'Feature: ProfileMaritalStatus       Rank: 2,  Keep: False'
'Feature: ProfileChildren            Rank: 17,  Keep: False'
'Feature: ProfileEducation           Rank: 7,  Keep: False'
'Feature: ProfileEducationMajor      Rank: 7,  Keep: False'
'Feature: Rewards                    Rank: 1,  Keep: True'
'Feature: AccountView                Rank: 1,  Keep: True'
'Feature: VerifyAnnualIncome         Rank: 1,  Keep: True'
'Feature: VerifyIncomeType           Rank: 5,  Keep: False'
'Feature: ProfileJobTitle            Rank: 11,  Keep: False'
'Feature: Login                      Rank: 1,  Keep: True'
'Feature: ProfileEmploymentLength    Rank: 12,  Keep: False'
'Feature: WebView                    Rank: 1,  Keep: True'
'Feature: SecurityModal              Rank: 1,  Keep: True'
'Feature: ResendToken                Rank: 1,  Keep: True'
'Feature: TransactionList            Rank: 1,  Keep: True'
'Feature: NetworkFailure             Rank: 15,  Keep: False'
'Feature: ListPicker                 Rank: 14,  Keep: False'
'Feature: Other                      Rank: 1,  Keep: True'
'Feature: SavingCount                Rank: 1,  Keep: True'
'Feature: CMCount                    Rank: 1,  Keep: True'
'Feature: CCCount                    Rank: 1,  Keep: True'
'Feature: LoansCount                 Rank: 1,  Keep: True'
```

2.1.3 Note: Boruta recommend keeping all the informative features. But five features ‘mean fractal dimension’, ‘texture error’, ‘smoothness error’, ‘symmetry error’, ‘fractal dimension error’ are less less infomative and was removed.

[33]:
```
dataset.columns
```

[33]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'location', 'Institutions',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth', 'ProfilePage',
       'VerifyCountry', 'Cycle', 'idscreen', 'Splash', 'RewardsContainer',
       'EditProfile', 'Finances', 'Alerts', 'Leaderboard', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'ResendToken',
       'TransactionList', 'NetworkFailure', 'ListPicker', 'Other',
       'SavingCount', 'CMCount', 'CCCount', 'LoansCount'],
      dtype='object')
```

[34]:
```
dataset = dataset.drop(['dayofweek', 'liked', 'RewardsContainer',␣
 ↪'EditProfile', 'Leaderboard', 'RewardDetail', 'ProfileMaritalStatus',␣
 ↪'ProfileChildren ', 'ProfileEducation', 'ProfileEducationMajor',␣
 ↪'VerifyIncomeType', 'ProfileJobTitle', 'ProfileEmploymentLength',␣
 ↪'NetworkFailure', 'ListPicker'], axis =1)
dataset.columns
```

[34]:
```
Index(['user', 'hour', 'age', 'numscreens', 'minigame', 'used_premium_feature',
       'enrolled', 'location', 'Institutions', 'VerifyPhone',
       'BankVerification', 'VerifyDateOfBirth', 'ProfilePage', 'VerifyCountry',
       'Cycle', 'idscreen', 'Splash', 'Finances', 'Alerts', 'VerifyMobile',
       'VerifyHousing', 'VerifyHousingAmount', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'Login', 'WebView', 'SecurityModal',
       'ResendToken', 'TransactionList', 'Other', 'SavingCount', 'CMCount',
       'CCCount', 'LoansCount'],
      dtype='object')
```

[35]:
```
dataset.to_csv('/content/drive/MyDrive/Data_incubator/data_sets/
 ↪new_engineeredFinal_appdata10.csv', index = False)
```

[36]:
```
analyze_report = sw.analyze(dataset)
analyze_report.show_html('/content/drive/MyDrive/Data_incubator/cb_EDA_output.
 ↪htm', open_browser = True)
```

```
                                                |          | [  0%]   00:00 -> (?␣
 ↪left)
```

```
Report /content/drive/MyDrive/Data_incubator/cb_EDA_output.htm was generated!
```

```
NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report IS
saved in your notebook/colab files.
```

2.2 Step 4: Feature scaling and Train test split

[37]:
```
# Splitting Independent and Response Variables
response = dataset["enrolled"]
dataset = dataset.drop(columns="enrolled")

# Splitting the dataset into the Training set and Test set, ub stands for user␣
 ↪behavior
#from sklearn.model_selection import train_test_split
X_trainub, X_testub, y_trainub, y_testub = train_test_split(dataset, response,
                                                    test_size = 0.2,
                                                    random_state = 42)
```

[38]:
```
# Removing Identifiers form both train and test feature, but save them so when␣
 ↪the model is done we can relate it back
train_identity = X_trainub['user']
X_trainub = X_trainub.drop(columns = ['user'])
test_identity = X_testub['user']
X_testub = X_testub.drop(columns = ['user'])
```

[39]:
```
# Feature Scaling
#we do this to avoid that a given feature has great influence on the model just␣
 ↪because its absolute value is big
#from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
#when standardization done the data we loose cols and index, we use pd.
 ↪DataFrame to avoid that
X_trainub2 = pd.DataFrame(sc_X.fit_transform(X_trainub))
X_testub2 = pd.DataFrame(sc_X.transform(X_testub)) #Note no fit here as fit is␣
 ↪done when fitting the training set
X_trainub2.columns = X_trainub.columns.values #get original cols
X_testub2.columns = X_testub.columns.values
X_trainub2.index = X_trainub.index.values #to recuperate original index
X_testub2.index = X_testub.index.values
X_trainub = X_trainub2
X_testub = X_testub2
```

[40]:
```
dataset_final = pd.read_csv('/content/drive/MyDrive/Data_incubator/data_sets/
 ↪new_engineeredFinal_appdata10.csv')
dataset_final.head(10)
```

[40]:
```
     user  hour  age  numscreens  minigame  used_premium_feature  enrolled  \
0  235136     2   23          15         0                     0         0
1  333588     1   24          13         0                     0         0
```

```
2  254414    19   23           3         0                     1         0
3  234192    16   28          40         0                     0         1
4   51549    18   31          32         0                     0         1
5   56480     9   20          14         0                     0         1
6  144649     2   35           3         0                     0         0
7  249366     3   26          41         0                     1         0
8  372004    14   29          33         1                     1         0
9  338013    18   26          19         0                     0         1

   location  Institutions  VerifyPhone  …  Login  WebView  SecurityModal  \
0         0             0            1  …      1        0              0
1         1             1            1  …      0        0              0
2         0             0            0  …      0        0              0
3         1             0            1  …      0        0              0
4         0             1            1  …      0        0              0
5         0             1            1  …      0        0              0
6         0             0            0  …      0        0              0
7         0             0            1  …      0        0              0
8         1             1            1  …      0        0              0
9         1             0            1  …      0        0              0

   ResendToken  TransactionList  Other  SavingCount  CMCount  CCCount  \
0            0                0      7            0        0        0
1            0                0      5            0        0        0
2            0                0      0            0        0        0
3            0                0      6            0        3        0
4            0                0     10            0        2        0
5            0                0      6            0        2        0
6            0                0      3            0        0        0
7            0                1      8            0        2        0
8            0                0     19            0        0        0
9            0                0     11            0        0        0

   LoansCount
0           1
1           1
2           1
3           1
4           1
5           1
6           0
7           1
8           3
9           1

[10 rows x 35 columns]
```

```
[41]: df_summary(dataset_final)
```

```
[41]:                  index  Missing Data  Unique Data Data Types
      0                 user             0        49874      int64
      1                 hour             0           24      int64
      2                  age             0           78      int64
      3           numscreens             0          151      int64
      4             minigame             0            2      int64
      5   used_premium_feature             0            2      int64
      6             enrolled             0            2      int64
      7             location             0            2      int64
      8         Institutions             0            2      int64
      9          VerifyPhone             0            2      int64
      10      BankVerification             0            2      int64
      11     VerifyDateOfBirth             0            2      int64
      12           ProfilePage             0            2      int64
      13         VerifyCountry             0            2      int64
      14                 Cycle             0            2      int64
      15              idscreen             0            2      int64
      16                Splash             0            2      int64
      17              Finances             0            2      int64
      18                Alerts             0            2      int64
      19          VerifyMobile             0            2      int64
      20         VerifyHousing             0            2      int64
      21   VerifyHousingAmount             0            2      int64
      22               Rewards             0            2      int64
      23           AccountView             0            2      int64
      24    VerifyAnnualIncome             0            2      int64
      25                 Login             0            2      int64
      26               WebView             0            2      int64
      27         SecurityModal             0            2      int64
      28           ResendToken             0            2      int64
      29       TransactionList             0            2      int64
      30                 Other             0           32      int64
      31           SavingCount             0           11      int64
      32               CMCount             0            6      int64
      33               CCCount             0            4      int64
      34            LoansCount             0            4      int64
```

```
[42]: dataset_final.columns
```

```
[42]: Index(['user', 'hour', 'age', 'numscreens', 'minigame', 'used_premium_feature',
             'enrolled', 'location', 'Institutions', 'VerifyPhone',
             'BankVerification', 'VerifyDateOfBirth', 'ProfilePage', 'VerifyCountry',
             'Cycle', 'idscreen', 'Splash', 'Finances', 'Alerts', 'VerifyMobile',
             'VerifyHousing', 'VerifyHousingAmount', 'Rewards', 'AccountView',
             'VerifyAnnualIncome', 'Login', 'WebView', 'SecurityModal',
             'ResendToken', 'TransactionList', 'Other', 'SavingCount', 'CMCount',
             'CCCount', 'LoansCount'],
            dtype='object')
```

[43]:

```
# Splitting Independent and Response Variables
response = dataset_final["enrolled"]
dataset_final = dataset_final.drop(columns="enrolled")

# Splitting the dataset into the Training set and Test set, ub stands for user␣
 ↪behaivor
#from sklearn.model_selection import train_test_split
X_trainub, X_testub, y_trainub, y_testub = train_test_split(dataset_final,␣
 ↪response,
                                                    test_size = 0.2,
                                                    random_state = 42)
```

[44]:

```
# Removing Identifiers form both train and test feature, but save them so when␣
 ↪the model is done we can relate it back
train_identity = X_trainub['user']
X_trainub = X_trainub.drop(columns = ['user'])
test_identity = X_testub['user']
X_testub = X_testub.drop(columns = ['user'])
```

[45]:

```
# Feature Scaling
#we do this to avoid that a given feature has great influence on the model just␣
 ↪because its absolute value is big
#from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
#when standardization done the data we loose cols and index, we use pd.
 ↪DataFrame to avoid that
X_trainub2 = pd.DataFrame(sc_X.fit_transform(X_trainub))
X_testub2 = pd.DataFrame(sc_X.transform(X_testub)) #Note no fit here as fit is␣
 ↪done when fitting the training set
X_trainub2.columns = X_trainub.columns.values #get original cols
X_testub2.columns = X_testub.columns.values
X_trainub2.index = X_trainub.index.values #to recuperate original index
X_testub2.index = X_testub.index.values
X_trainub = X_trainub2
X_testub = X_testub2
```

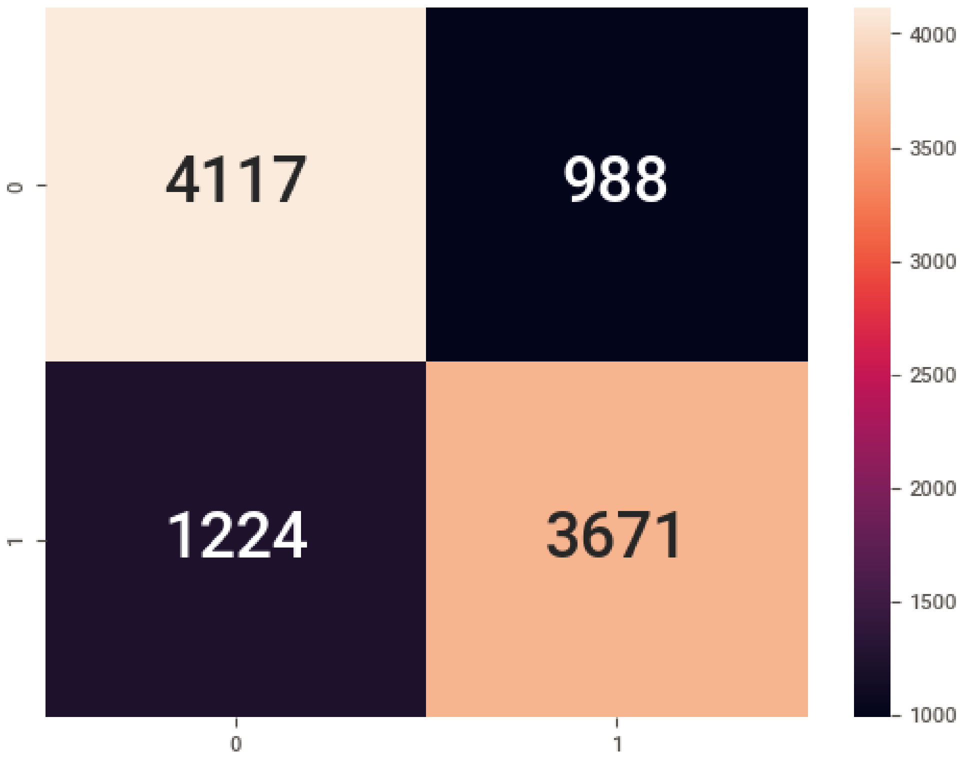
[46]:

```
xgb_classifier = XGBClassifier(colsample_bylevel=0.9, colsample_bytree=0.9,␣
 ↪importance_type='gain', learning_rate=0.01, max_depth=4, n_estimators=200,␣
 ↪reg_alpha=0.1, reg_lambda=0.5, subsample=1.0, random_state=0)
xgb_classifier.fit(X_trainub, y_trainub)
xgb_y_pred = xgb_classifier.predict(X_testub)
xgb_cm = confusion_matrix(y_testub, xgb_y_pred)
```

```
print('xgb Confusion Matrix:')
print(xgb_cm)
print("---------------------------------------------------------------")
print('xgb Accuracy: ', accuracy_score(y_testub, xgb_y_pred))
print("---------------------------------------------------------------")
print("classification report")
print(classification_report(y_testub, xgb_y_pred))
print("---------------------------------------------------------------")
print('XGBClassifier Confusion Matrix heatmap:')
sns.heatmap(xgb_cm, annot=True, fmt="d")
```

```
xgb Confusion Matrix:
[[4117  988]
 [1224 3671]]
---------------------------------------------------------------
xgb Accuracy:  0.7788
---------------------------------------------------------------
classification report
              precision    recall  f1-score   support

           0       0.77      0.81      0.79      5105
           1       0.79      0.75      0.77      4895

    accuracy                           0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.78      0.78      0.78     10000

---------------------------------------------------------------
XGBClassifier Confusion Matrix heatmap:
```

```
[46]: <Axes: >
```



[47]:
```
r_probs = [0 for _ in range(len(y_testub))]
rf_probs = xgb_classifier.predict_proba(X_testub)
```

[48]:
```
# Probabilities for the positive outcome is kept.
rf_probs = rf_probs[:, 1]
```

[50]:
```
r_auc = roc_auc_score(y_testub, r_probs)
rf_auc = roc_auc_score(y_testub, rf_probs)

#Print AUROC scores
print('Random (chance) Prediction: AUROC = %.3f' % (r_auc))
print('gb Customer Behavior: AUROC = %.3f' % (rf_auc))
```

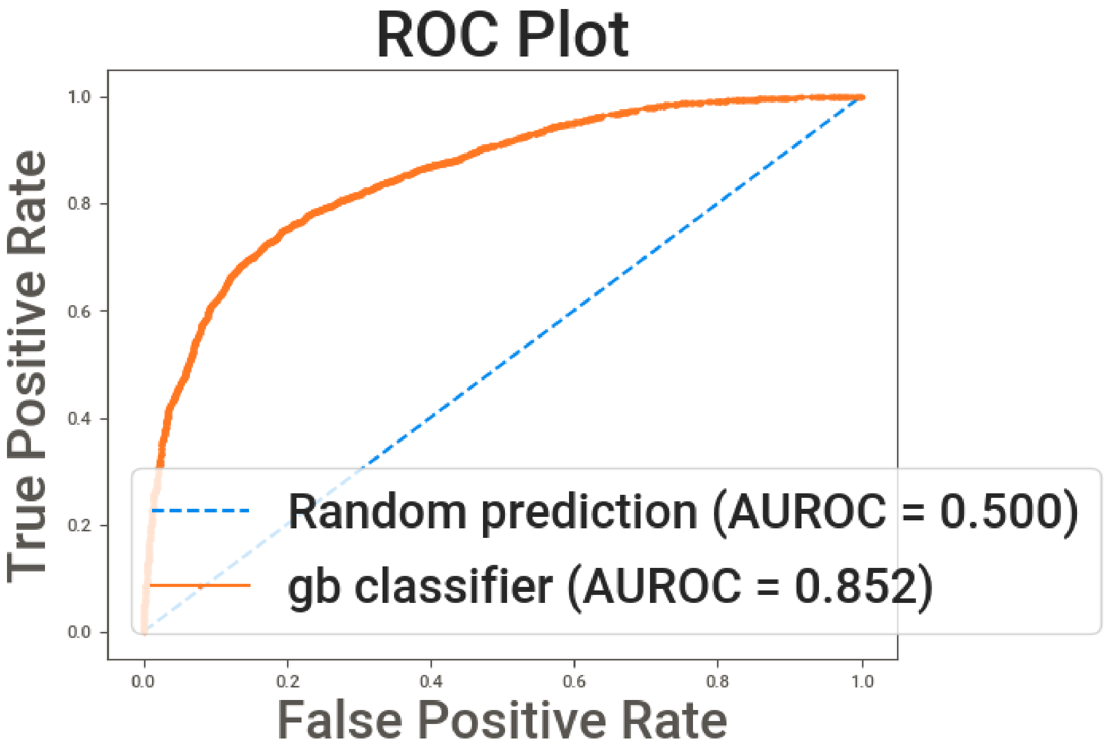
```
Random (chance) Prediction: AUROC = 0.500
gb Customer Behavior: AUROC = 0.852
```

[51]:
```
#Calculate ROC curve
r_fpr, r_tpr, _ = roc_curve(y_testub, r_probs)
rf_fpr, rf_tpr, _ = roc_curve(y_testub, rf_probs)
```

2.3 Plot the ROC curve

[52]:

```
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random prediction (AUROC = %0.
 ↪3f)' % r_auc)
plt.plot(rf_fpr, rf_tpr, marker='.', label='gb classifier (AUROC = %0.3f)' %␣
 ↪rf_auc)

# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```



3 The ML model developed here classifies customers into easily subscribers and hard to engage with 79 % Accuracy! This model can aid marketing team to target those custmers with hard to engage behaviors.

```
[54]: # Formatting Final Results
      final_results_XB = pd.concat([y_testub, test_identity], axis = 1).dropna()
      final_results_XB['predicted_reach'] = xgb_y_pred
      final_results_XB = final_results_XB[['user', 'enrolled', 'predicted_reach']].
       ↪reset_index(drop=True)
```

```
[ ]: final_results_XB
```

```
[ ]:         user  enrolled  predicted_reach
     0      48024         1                0
     1     266463         1                0
     2     262169         1                0
     3     213736         1                0
     4     169082         1                1
     …        …        …              …
     9995  367119         0                1
     9996    4433         0                0
     9997  146917         1                1
     9998   77281         1                1
     9999   40310         0                0

     [10000 rows x 3 columns]
```

```
[55]: # final prediction for marketing team. Model predcition and Customer ID␣
       ↪recombined
      final_results_XB.to_csv('/content/drive/MyDrive/Data_incubator/data_sets/
       ↪final_predictionXB_forMarketingTeam.csv', index = False) # this is clean␣
       ↪prediction to be used by marketing team to target the customers
```

```
[56]: # Save the model in pickle file
      pickle.dump(xgb_classifier, open("/content/drive/MyDrive/Data_incubator/
       ↪data_sets/customer_behaver_predictor.pkl","wb"))
```

```
[57]: !jupyter nbconvert --to pdf /content/drive/MyDrive/Data_incubator/
       ↪Adama_Capstone_project_final.ipynb
```

```
[NbConvertApp] Converting notebook
/content/drive/MyDrive/Data_incubator/Adama_Capstone_project_final.ipynb to pdf
[NbConvertApp] Support files will be in Adama_Capstone_project_final_files/
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
```

```
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Making directory ./Adama_Capstone_project_final_files
[NbConvertApp] Writing 187229 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 882656 bytes to
/content/drive/MyDrive/Data_incubator/Adama_Capstone_project_final.pdf
```