Customer Behaviour Prediction System

October 13, 2019

Contents

1 Abstract **1**

2 Introduction **2**

3 Scope **3**

4 Requirements **4**

4.1 Functional Requirements: 4

4.2 Non-functional Requirements: 4

5 Use Case Diagram **5**

6 Implementation **6**

6.1 Deployment Diagram 6

6.2 Dataset Description 7

6.3 Source Code 8

7 Testing **14**

8 Conclusion **15**

List of Figures

5.1 Use Case Diagram. 5

6.1 Deployment Diagram. 6

7.1 Confusion Matrix. 14

Chapter 1

Abstract

In today's market, many companies have a mobile presence. Often, these companies provide free products/services in their mobile apps in an attempt to transition their customers to a paid membership. Some examples of paid products, which originate from free ones, include YouTube Red, Pandora Premium, Audible Subscription and You Need a Budget. Since marketing efforts are never free, these companies need to know exactly who to target with their offers and promotions. The system will identify which users will most likely not enroll in a paid product, so that additional offer can be given to them. Due to the costs of these offers, the company does not want to offer them to everybody, especially customers who were going to enroll anyways.

Chapter 2

Introduction

The data comes from a fintech company that wants to provide its customers with a paid mobile app subscription that will allow them to track all their finances in one place. To attract customers, the company releases a free version of their app with some of the main features unlocked. The system have access to each customer's app behaviour data. App behaviour is charecterised as the list of app screens the user looked at and whether the user played the financial mini-games available. Due to this data, one can predict the behaviour of the customer who are not likely to enroll and who are.

Chapter 3

Scope

For a paid software, the system can narrow the marketing efforts only for those users who are "unlikely" to subscribe and thus can increase the subscriber rate by providing the target offers. The increase in overall subscriptions can measure the benefit of this model to the company. By this users who are likely to leave may convert to paid subscriber if we give them an offer they cannot refuse.

Chapter 4

Requirements

4.1 Functional Requirements:

- Ability to predict the customers who are not likely to enroll
- Identify correlation between different screen list

4.2 Non-functional Requirements:

- Each costumer's app behaviour data should be available
- Python IDE is required with sklearn library

Chapter 5

Use Case Diagram

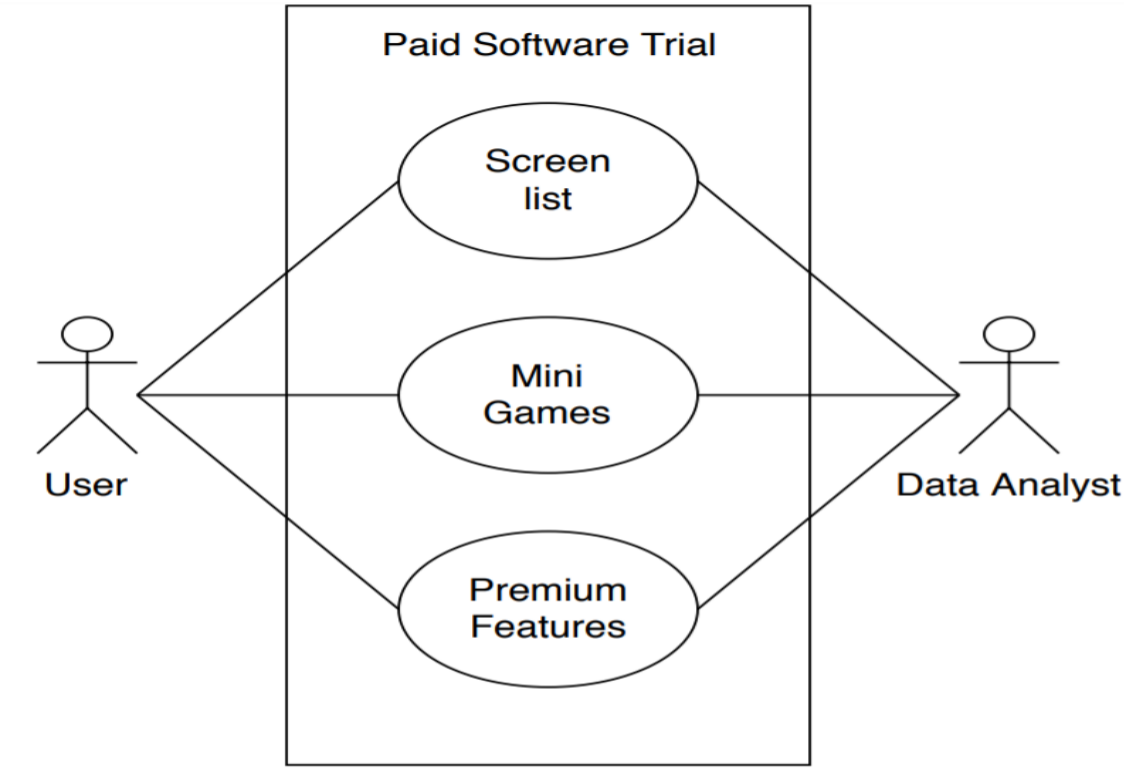

Figure 5.1: Use Case Diagram.

Chapter 6

Implementation

6.1 Deployment Diagram

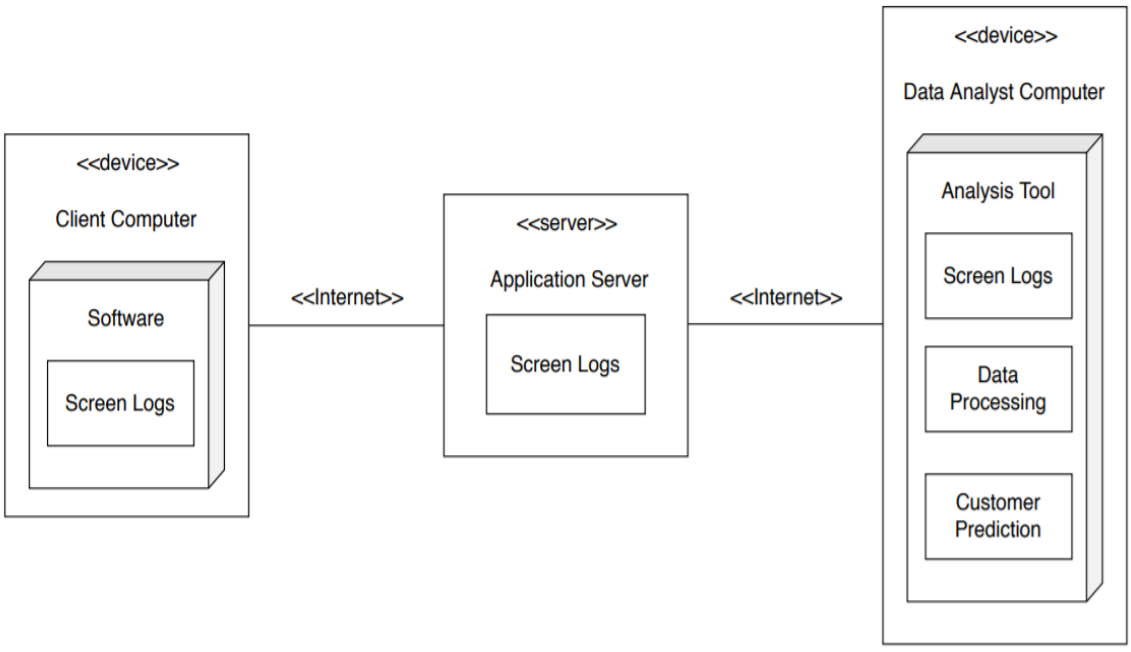

Figure 6.1: Deployment Diagram.

6.2 Dataset Description

The app usage data is only from the user's first day in the app. This limitation exists because users can enjoy a 24-hour free trial of the premium features and the company wants to target them with new offers shortly after the trial is over.
The data fields we'll be working with are as follows:

• **first-open:** The datetime when the user first opened the app

• **dayofweek:** the integer day of the week when the user first opened the app. Starts at 0, which is Sunday and runs to 6 which is Saturday.

• **hour:** Hour of the day when user first opened app. in 24 hours format as 18:00:00. Correlates with dayofweek.

• **age:** Age of the user.

• **screen-list:** Comma-seperated list of screens that the users accessed in their first 24 hours.

• **numscreens:** The number of screens accessed in their first 24 hours.

• **liked:** Each screen has an feature to 'like' that particualr screen. If any screens are liked, this value will be 1, otherwise 0.

• **minigame:** 1 if the user played the mini-game, 0 otherwise.

• **used-premium-feature:** 1 if the user accessed any of the premium features in the first 24 hours, 0 otherwise.

• **enrolled:** 1 if the user enrolled, 0 otherwise. This is the field that will be predicted later.

• **enrolled-date:** If the user enrolled at any time, this value is popopulated with the date of enrollment.

6.3 Source Code

Data Prep

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dateutil import parser
df = pd.read_csv('data/appdata10.csv')
df.head()
df.describe()
df.info()
df["hour"] = df.hour.str.slice(1, 3).astype(int)
df.describe()
df2 = df.copy().drop(['user', 'screen_list', 'enrolled_date', 'first_open', 'enrolled'], axis=1)
df2.head()
print(df2.shape)
print(df2.shape[1])
print(df2.shape[1]+1)
print(df2.columns)
print(df2.columns.values)
print(df2.columns.values[0])
print(np.size(df2))
print(np.size(df2.iloc[:,0]))
print(np.size(df2.iloc[:,0].unique()))
print(np.size(df2.iloc[:,1].unique()))
plt.figure(figsize=(17,10))
plt.suptitle('Histograms of Numerical Columns', fontsize=20) for i in range(1, df2.shape[1]+1):
plt.subplot(3, 3, i)
f = plt.gca()
f.set_title(df2.columns.values[i-1])
```

```
vals = np.size(df2.iloc[:, i-1].unique())
plt.hist(df2.iloc[:, i-1], bins=vals, color='#3F5D7D')
plt.subplots_adjust(hspace=0.5)
df2.corrwith(df.enrolled)
df2.corrwith(df.enrolled).plot.bar(figsize=(20, 10), title='Correlation with Enrolled', font-
size=15, rot=45, grid=True)
df2.corr()
sns.set(style='white', font_scale=2)
corr = df2.corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize = (18, 15))
f.suptitle('Correlation Matrix', fontsize=40)
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0, square=True, linewidths=.5,
cbar_kws='shrink': .5)
df.dtypes
df['first_open'] = pd.to_datetime(df.first_open)
df['enrolled_date'] = pd.to_datetime(df.enrolled_date)
df.dtypes
df.head()
df["difference"] = (df.enrolled_date - df.first_open).astype('timedelta64[h]')
df.head()
plt.figure(figsize=(13, 8))
plt.hist(df['difference'].dropna(), color='#3F5D7D')
plt.title('Most users who sign up do so within the first 500 hours of app usage')
plt.figure(figsize=(13, 8))
plt.hist(df['difference'].dropna(), color='#3F5D7D', range=[0, 100])
plt.title('Most signups occur within first 10 hours of usage')
plt.figure(figsize=(13, 8))
plt.hist(df['difference'].dropna(), color='#3F5D7D', range=[0, 10])
```

```
plt.title('Most signups occur within first hour of usage')
df.loc[df.difference ¿ 48, 'enrolled'] = 0
df = df.drop(['difference', 'enrolled_date', 'first_open'], axis=1)
top_screens = pd.read_csv('data/top_screens.csv')
top_screens.head()
top_screens = top_screens.top_screens.values top_screens
df['screen_list'] = df.screen_list.astype(str) + ','
for sc in top_screens:
df[sc] = df.screen_list.str.contains(sc).astype(int)
df['screen_list'] = df.screen_list.str.replace(sc+',', '')
df["other"] = df.screen_list.str.count(',')
df.head()
df = df.drop(['screen_list'], axis=1)
savings_screens = [
"Saving1",
"Saving2",
"Saving2Amount",
"Saving4",
"Saving5",
"Saving6",
"Saving7",
"Saving8",
"Saving9",
"Saving10",
]
df["savings_count"] = df[savings_screens].sum(axis=1)
df.head()
df = df.drop(savings_screens, axis=1)
cm_screens = [
"Credit1",
"Credit2",
```

```
"Credit3",
"Credit3Container",
"Credit3Dashboard",
]
df["cm_count"] = df[cm_screens].sum(axis=1)
df = df.drop(cm_screens, axis=1)
cc_screens = [
"CC1",
"CC1Category",
"CC3"
]
df["cc_count"] = df[cc_screens].sum(axis=1)
df = df.drop(cc_screens, axis=1)
loan_screens = [
"Loan",
"Loan2",
"Loan3",
"Loan4"
]
df["loan_count"] = df[loan_screens].sum(axis=1)
df = df.drop(loan_screens, axis=1)
df.head()
df.columns
df.to_csv('data/new_appdata10.csv', index=False)
df = pd.read_csv('data/new_appdata10.csv')
df.head()
df.shape
import time
response = df.enrolled
df = df.drop(['enrolled'], axis=1)
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(df, response, test_size=0.2, random_state=0)
train_identifier = X_train['user']
X_train = X_train.drop('user', axis=1)
test_identifier = X_test['user']
X_test = X_test.drop('user', axis=1)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns.values,
index=X_train.index.values)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns.values,
index=X_test.index.values)
X_train_scaled.head()
X_test_scaled.head()
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, penalty='L1', solver='saga')
clf.fit(X_train_scaled, y_train)
y_pred = clf.predict(X_test_scaled)
y_pred
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, f1_score,
precision_score, recall_score
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,7))
sns.set(font_scale=1.4)
sns.heatmap(pd.DataFrame(cm, index=(0,1), columns=(0,1)), annot=True, fmt='g')
print('Test data accuracy: :0.4f'.format(accuracy_score(y_test, y_pred)))
print(classification_report(y_test, y_pred))
from sklearn.model_selection import cross_val_score
acc = cross_val_score(clf, X_train_scaled, y_train, cv=10)
print("Logistic Accuracy: :0.3f (+/- :0.3f std. dev)".format(acc.mean(), acc.std()))
final_results = pd.concat([y_test, test_identifier], axis=1).dropna()
final_results['predicted'] = y_pred
```

final_results = final_results[['user', 'enrolled', 'predicted']].reset_index(drop=True)
final_results

Chapter 7

Testing

Test data accuracy: 76.81

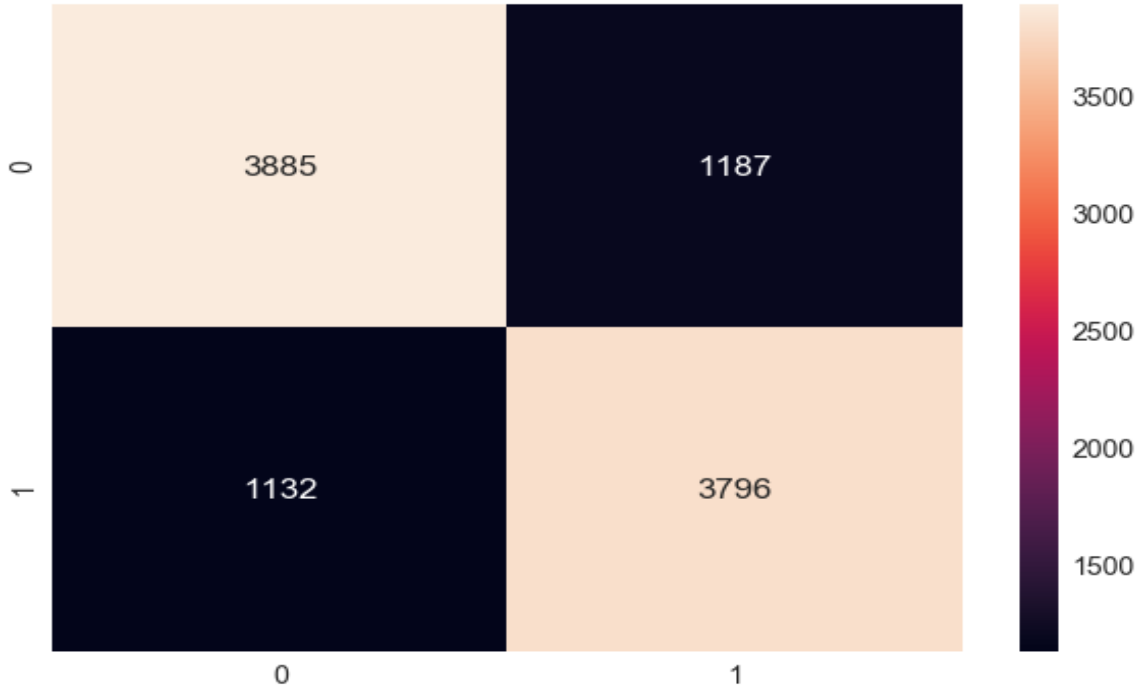

Figure 7.1: Confusion Matrix.

Chapter 8

Conclusion

This system will give a model that will label every new user as "highly likely" or "unlikely" to subscribe. It can further validate results by running the predictions on daily new installs and see whether the system's accuracy is consistent. From there, the app owner can narrow their marketing efforts only for those users who are "unlikely" to subscribe and thus increase their subscriber rate. The increase in overall subscriptions can measure the benefit of this model to the company.