Starbucks Capstone Project Report

Machine Learning Engineer Nanodegree | Shantanu Dave

Project Overview

Starbucks, one of the famous coffee sellers in today's time. However, it is necessary to see how these numbers are growing and how to keep track of it. In this case, ML plays a vital role.

In this project, three different datasets are provided for solving the questions such as which offers are attracting the customers and how the business is growing. Using the three json files we determine the target value and build model to further analyse it.

Problem Statement

The data from the course is taken to solve the task. The aim is to merge the portfolio, profile and transcript files and asking the questions like which age group is using the offers, what is the demographic and so on. Here I have tried KNN, Decision trees and Random Forest Classification where knn is used as bench model and finally using accuracy and f1 score as evaluation metrics.

Data Analysis

The data is contained in three files:

• portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7

```
array(['bogo', 'informational', 'discount'], dtype=object)
```

Portfolio offer_types

• profile.json - demographic data for each customer

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

Profile data

```
profile.isna().sum()
```

```
gender              2175
age                    0
id                     0
became_member_on       0
income              2175
dtype: int64
```

It can be seen that, there lies Nan values in gender and income which can be pre-processed further.

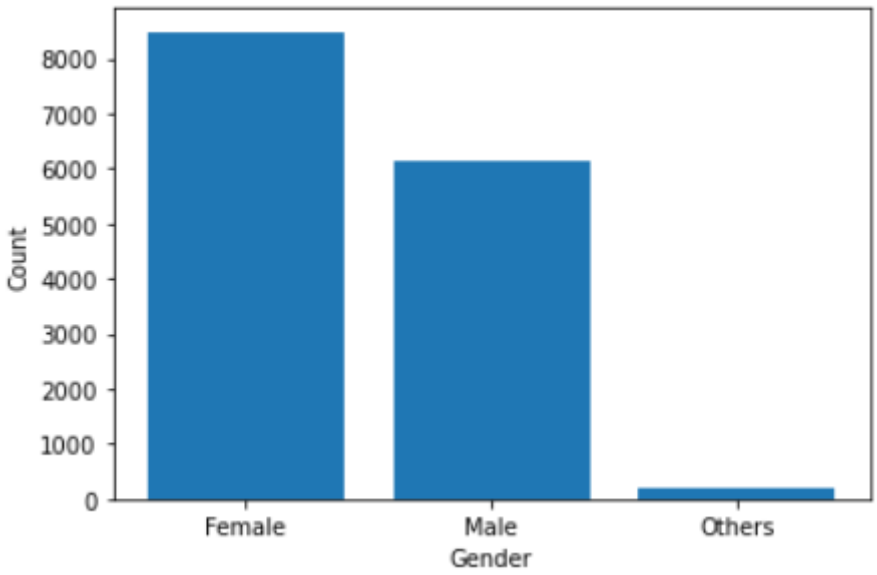
• transcript.json - records for transactions, offers received, offers viewed, and offers completed

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

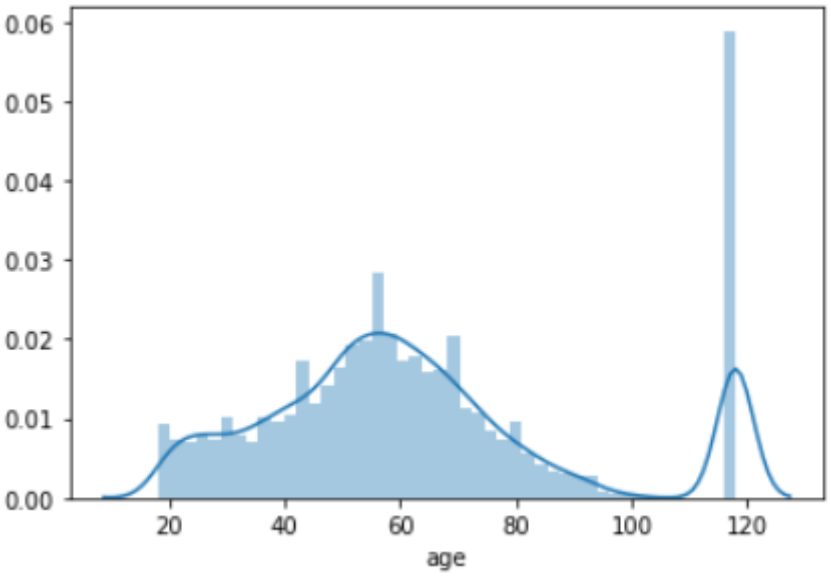
Transaction data

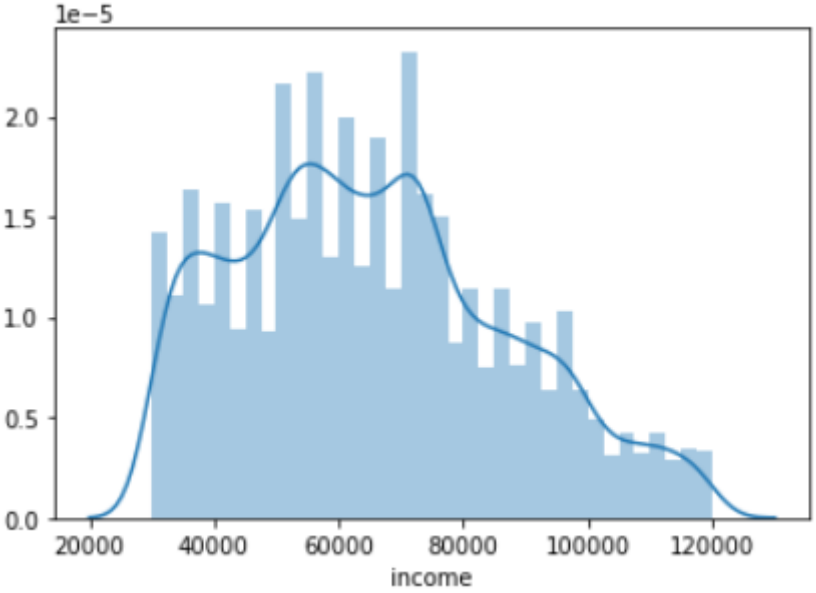
Exploratory Data Analysis

- From the profile file, the count for different genders are displayed using bar graph.

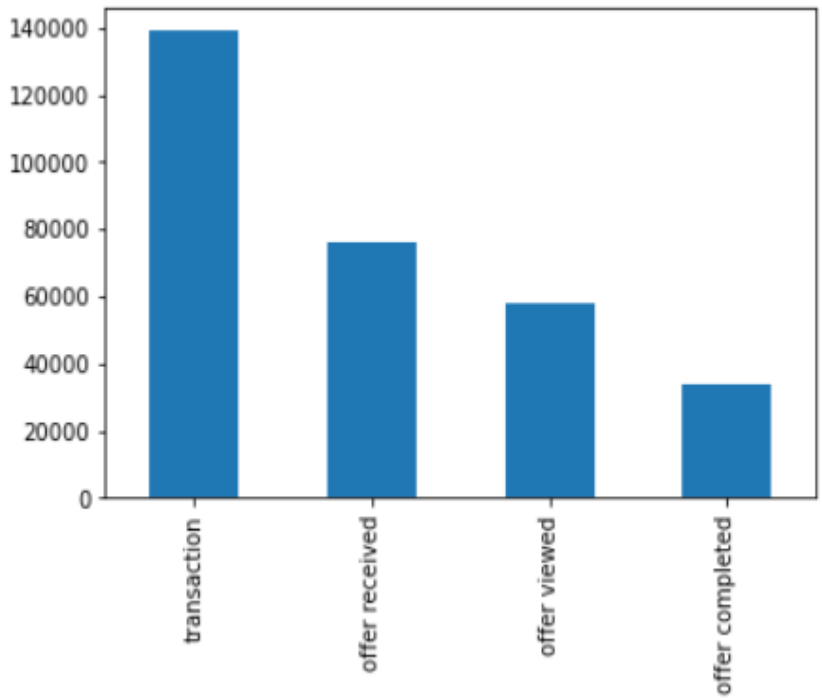


- The below graph shows the distplot of age where a rolling of age is clearly seen.





- Similar to age, income column is plotted as distplot to see the variation in the income data.



- The above figure shows the count of events in the transcript file

Pre-processing

Before implementing any model, it is necessary to clean the dataset for the best results. In this case, the following preprocessing steps are taken into consideration and finally the files are merged.

- **portfolio.json**

1. **offer_type** column is changed to columns similar to pd.dummies

```
portfolio_data['bogo'] = portfolio_data['offer_type'].apply(lambda x: 1 if 'bogo' in x else 0)
portfolio_data['discount'] = portfolio_data['offer_type'].apply(lambda x: 1 if 'discount' in x else 0)
portfolio_data['informational'] = portfolio_data['offer_type'].apply(lambda x: 1 if 'informational' in x else 0)
portfolio_data.drop(['offer_type'], axis=1, inplace=True)
```

2. **channel** column is changed to columns from array

```
portfolio_data['web']= portfolio_data['channels'].apply(lambda x:1 if 'web' in x else 0)
portfolio_data['email']= portfolio_data['channels'].apply(lambda x:1 if 'email' in x else 0)
portfolio_data['mobile']= portfolio_data['channels'].apply(lambda x:1 if 'mobile' in x else 0)
portfolio_data['social']= portfolio_data['channels'].apply(lambda x:1 if 'social' in x else 0)
portfolio_data.drop(['channels'], axis=1, inplace=True)
```

3. **id** column is changed to offer_id

```
portfolio_data['offer_id'] = portfolio_data['id']
portfolio_data.set_index('offer_id', inplace=True)
portfolio_data.drop(['id'], axis=1, inplace=True)
```

Final, portfolio dataset after cleaning is seen as below

offer_id	reward	difficulty	duration	bogo	discount	informational	web	email	mobile	social
ae264e3637204a6fb9bb56bc8210ddfd	10	10	7	1	0	0	0	1	1	1
4d5c57ea9a6940dd891ad53e9dbe8da0	10	10	5	1	0	0	1	1	1	1
3f207df678b143eea3cee63160fa8bed	0	0	4	0	0	1	1	1	1	0
9b98b8c7a33c4b65b9aebfe6a799e6d9	5	5	7	1	0	0	1	1	1	0
0b1e1539f2cc45b7b9fa7c272da2e1d7	5	20	10	0	1	0	1	1	0	0

- **profile.json**

1. **gender** column is changed to columns similar to pd.dummies

```
profile_data['male'] = profile_data['gender'].apply(lambda x: 1 if x == 'M' else 0)
profile_data['female'] = profile_data['gender'].apply(lambda x: 1 if x == 'F' else 0)
profile_data['others'] = profile_data['gender'].apply(lambda x: 1 if x != 'M' and x != 'F' else 0)
profile_data.drop(['gender'], axis=1, inplace=True)
```

2. Age column where age is 118 is converted to nan values and then removed

```
profile_data['age'] = profile_data['age'].apply(lambda x:np.nan if x == 118 else x)
profile_data.dropna(inplace=True)
```

3. **Age** is grouped into bins and labelled and finally converted to numerical values

```
profile_data['age_bin'] = pd.cut(x=profile_data["age"], bins=[18,39,59,79,99,110],
                                 labels=['group1', 'group2', 'group3', 'group4','group5'])
```

```
profile_data['age_bin'] = pd.cut(x=profile_data["age"], bins=[18,39,59,79,99,110],
                                 labels=['group1', 'group2', 'group3', 'group4','group5'])
```

```
age_bin = profile_data['age_bin'].astype('category').cat.categories.tolist()
age_bin = {'age_bin' : {k: v for k,v in zip(age_bin,list(range(1,len(age_bin)+1)))}}
```

4. **Income** column is also grouped into bins and finally converted to numerical values

```
profile_data['income_bin'] = pd.cut(x=profile_data['income'],
                                    bins=[30000, 50000, 70000, 90000, 120000],labels=['sal1', 'sal2', 'sal3','sal4'])
```

```
income_bin = profile_data['income_bin'].astype('category').cat.categories.tolist()
income_bin = {'income_bin' : {k: v for k,v in zip(income_bin,list(range(1,len(income_bin)+1)))}}
```

5. **Member_days** column is also grouped into bins and finally converted to numerical values

```
# creating a new column 'member_type' representing the type of the member: new, regular or loyal depending on the number of his
profile_data['member_days'] = pd.cut(profile_data['member_days'],
                                    bins=[1000, 1500, 2000, 2500, 3000],labels=['recent', 'new', 'regular', 'longtime'])
```

```
member_type = profile_data['member_days'].astype('category').cat.categories.tolist()
member_type = {'member_days' : {k: v for k,v in zip(member_type,list(range(1,len(member_type)+1)))}}
```

Final, profile dataset after cleaning is seen as below

profile_id	age	became_member_on	income	male	female	others	age_bin	income_bin	member_days
0610b486422d4921ae7d2bf64640c50b	55.0	20170715	112000.0	0	1	0	2.0	4.0	1
78afa995795e4d85b5d9ceeca43f5fef	75.0	20170509	100000.0	0	1	0	3.0	4.0	1
e2127556f4f64592b11af22de27a7932	68.0	20180426	70000.0	1	0	0	3.0	2.0	1
389bc3fa690240e798340f5a15918d5c	65.0	20180209	53000.0	1	0	0	3.0	2.0	1
2eeac8d8feae4a8cad5a6af0499a211d	58.0	20171111	51000.0	1	0	0	2.0	2.0	1
...	...	...	...	...	...	...	...	...	...
6d5f3a774f3d4714ab0c092238f3a1d7	45.0	20180604	54000.0	0	1	0	2.0	2.0	1
2cb4f97358b841b9a9773a7aa05a9d77	61.0	20180713	72000.0	1	0	0	3.0	3.0	1
01d26f638c274aa0b965d24cefe3183f	49.0	20170126	73000.0	1	0	0	2.0	3.0	2
9dc1421481194dcd9400aec7c9ae6366	83.0	20160307	50000.0	0	1	0	4.0	1.0	2
e4052622e5ba45a8b96b59aba68cf068	62.0	20170722	82000.0	0	1	0	3.0	3.0	1

- **transcript.json**

1. the array values in the **value** column is separated to various columns and then these values are iterated to fill the empty columns created.

```
transcript_data['amount'] = 0
transcript_data['reward'] = 0
transcript_data['offer_id'] = ''
```

```
for num, row in transcript_data.iterrows():
    for val in row['value']:
        if val == 'amount':
            transcript_data.at[num, 'amount'] = row['value'][val]
        if val == 'reward':
            transcript_data.at[num, 'reward'] = row['value'][val]
        if val == 'offer_id' or val == 'offer id':
            transcript_data.at[num, 'offer_id'] = row['value'][val]
```

2. I have excluded transaction column from event and used other three for the events such as offer received, viewed and completed.

```
transcript_data = transcript_data[transcript_data['event'] != 'transaction']
```

```
event_data = transcript_data['event'].astype('category').cat.categories.tolist()
event_label = {'event' : {k: v for k,v in zip(event_data,list(range(1,len(event_data)+1)))}}
```

3. Renamed person with profile_id

```
transcript_data['profile_id'] = transcript_data['person']
transcript_data.drop('person', axis=1, inplace=True)
```

Final, transcript dataset after cleaning is seen as below

	event	time	amount	reward	offer_id	profile_id
0	2	0	0	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	78afa995795e4d85b5d9ceeca43f5fef
1	2	0	0	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	a03223e636434f42ac4c3df47e8bac43
2	2	0	0	0	2906b810c7d4411798c6938adc9daaa5	e2127556f4f64592b11af22de27a7932
3	2	0	0	0	fafdcd668e3743c1bb461111dcafc2a4	8ec6ce2a7e7949b1bf142def7d0e0586
4	2	0	0	0	4d5c57ea9a6940dd891ad53e9dbe8da0	68617ca6246f4fbc85e91a2a49552598

Merging the files and Normalising the values

The three files after cleaning are merged together using the offer_id and profile_id. Further the data from the columns are normalised using the sklearns minmax scaler.

Model Implementation

- The data is first divided into X and y values where y being the event and X being the various features. Then the data is split into train and test sets where 80% of data is used as training set and 20% as test set.

```
def split_data(X, y, test_size):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state=1)
    return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = split_data(X, y, 0.2)
```

- **Benchmark Model – KNN model**

  In this case, a KNN model with default k values is implemented which gives us a prediction accuracy of 91%.

```
model = KNeighborsClassifier()

# Train the model using the training sets
model.fit(X_train,y_train)
```

```
KNeighborsClassifier()
```

```
k_predictions = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, k_predictions))
```

```
0.9171756243213898
```

```
f1_score(y_test, k_predictions)
```

```
1.0
```

- Decision Tree Classifier: Further a decision tree classifier model was implemented by fitting the X_train and y_train values.

```
model_dt = DecisionTreeClassifier()
model_dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

- Random Forest Classifier: Further a random forest classifier model was implemented by fitting the X_train and y_train values.

```
model_rf = RandomForestClassifier(n_estimators = 100, random_state = 32)
model_rf.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=32)
```

Evaluation

The bench mark model provided an accuracy of 100% and f1_score of 1.

Similar to the benchmark model, the Decision tree classifier and Random forest classifier provided an accuracy of 100% and f1_score 1.

The models can also be evaluated again various other metrics according to the requirement.

```
print(accuracy_score(y_test, dt_predictions))
```

```
1.0
```

```
from sklearn.metrics import f1_score
```

```
dt_score = f1_score(y_test, dt_predictions)
```

```
dt_score
```

```
1.0
```

```
print(accuracy_score(y_test, rf_predictions))
```

```
1.0
```

```
rf_score = f1_score(y_test, rf_predictions)
```

```
rf_score
```

```
1.0
```

Conclusion

The project started with uploading the data and exploring it. After Analysing the data the important Parameters were determined and then the three files were preprocessed seperately keeping in mind all the necessary changes. Further the data was merged into a single file which was used to build the model.

In this case I have used KNN as the bench Model which provided an accuracy of 100% and F1 Score of 1.0, keeping this in mind the ensemble trees classifier such as decision tree and random forest classifiers were built and fit for the training set. Finally, the models were evaluated using accuracy and F1 score as metrics which provided 100% result as of the bench model.

Further, this model can be improved and also the clean dataset can be further used for recommendations and also various other algorithms to check for the variations.