Machine Learning Engineer Nanodegree

Omar Hegazy

Capstone Project – Starbucks - Report

Domain Background

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

Machine Learning algorithms use statistics to find patterns in massive amount of data, and data here encompasses a lot of things from numbers, words, images, clicks. If it can be digitally stored it can be fed into machine learning algorithms.

Machine learning has become increasingly important over the years. The need to get insights about the behavior of a customer based on the data collected over the years is one of the most trending uses of machine learning nowadays. My project is about Starbucks, Starbucks Corporation is an American multinational chain of coffeehouses and roastery reserves headquartered in Seattle, Washington. As the largest coffeehouse in the world, Starbucks is seen to be the main representation of the United States' second wave of coffee culture. Since the 2000s, third wave coffee makers have targeted quality-minded coffee drinkers with hand-made coffee based on lighter roasts, while Starbucks nowadays uses automatic espresso machines for efficiency. The company operates 30,000 locations worldwide in over 77 countries.

Starbucks means to attract and retain customers, Starbucks leverages a rewards program that honors regular customers with special offers not available to the standard customer. For this project, we’ll be combing through some fabricated customer and offer data provided by Starbucks and Udacity to understand how Starbucks may choose to alter its rewards program to better suit specific customer segments.

Problem Statement

Starbucks wants to find a way to give to each customer the right in app special offer. There are three types of offers

1- Buy One Get One (BOGO)
2- Classic Discount
3- Informational (no real offer)

Our goal is to analyze historical data about app usage and offers orders to develop an algorithm that associates each customer to the right offer type We can measure the performance of the algorithm by measuring its accuracy.

Dataset and Cleaning

There are three datasets provided by udacity and starbucks for this project .

profile.json (Rewards program users (17000 users x 5 fields)

1- gender: (categorical) M, F, O, or null
2- age: (numeric) missing value encoded as 118
3- id: (string/hash)
4- became_member_on: (date) format YYYYMMDD
5- income: (numeric)

Cleaning steps:

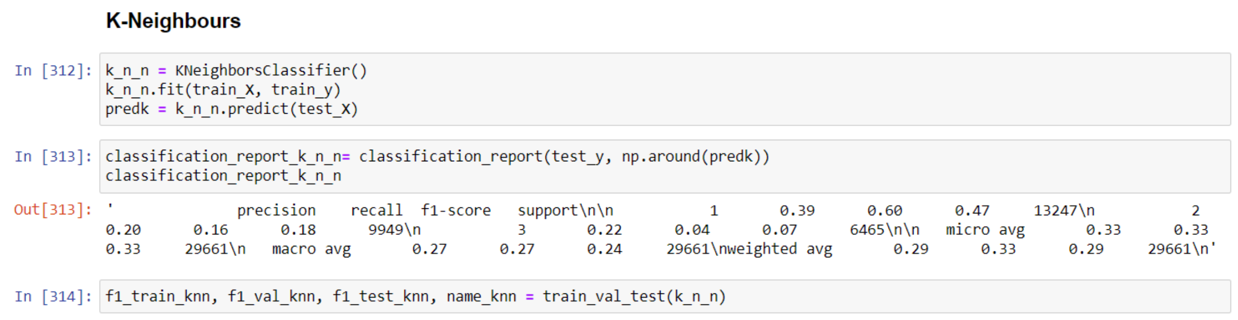
- rename id to customer_id
- change became_member_on datatype from string to datetime
- fill missing value in income with the mean
- fill missing value in income with the mode(most frequent)
- categorize people with there age

portfolio.json (Offers sent during 30-day test period (10 offers x 6 fields))

1- reward: (numeric) money awarded for the amount spent
2- channels: (list) web, email, mobile, social
3- difficulty: (numeric) money required to be spent to receive reward
4- duration: (numeric) time for offer to be open, in days
5- offer_type: (string) bogo, discount, informational
6- id: (string/hash)

Cleaning steps:

- rename id to customer_id
- apply one hot encoding to channels column



K-Neighbours

```
In [312]: k_n_n = KNeighborsClassifier()
          k_n_n.fit(train_X, train_y)
          predk = k_n_n.predict(test_X)

In [313]: classification_report_k_n_n= classification_report(test_y, np.around(predk))
          classification_report_k_n_n

Out[313]: '              precision    recall  f1-score   support\n\n           1       0.39      0.60      0.47     13247\n           2
          0.20      0.16      0.18      9949\n           3      0.22      0.04      0.07      6465\n\n   micro avg       0.33      0.33
          0.33     29661\n   macro avg       0.27      0.27      0.24     29661\nweighted avg       0.29      0.33      0.29     29661\n'

In [314]: f1_train_knn, f1_val_knn, f1_test_knn, name_knn = train_val_test(k_n_n)
```

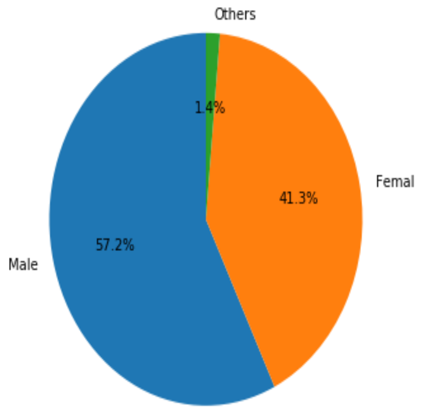
pt.json (Event log (306648 events x 4 fields))

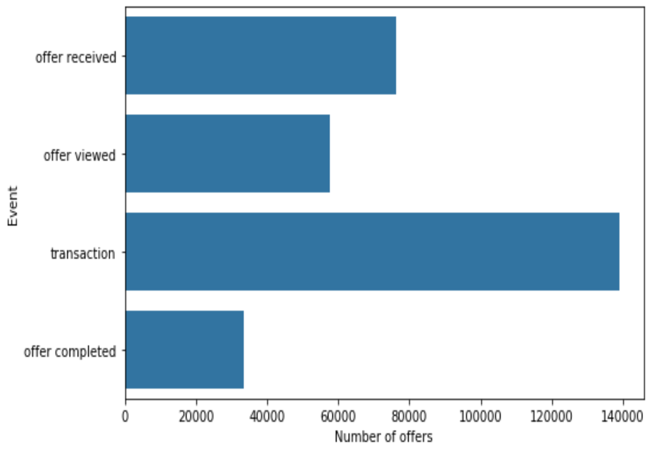
1- person: (string/hash)
2- event: (string) offer received, offer viewed, transaction, offer completed
3- value: (dictionary) different values depending on event type
4- offer id: (string/hash) not associated with any "transaction"
5- amount: (numeric) money spent in "transaction"
6- reward: (numeric) money gained from "offer completed"
7- time: (numeric) hours after start of test

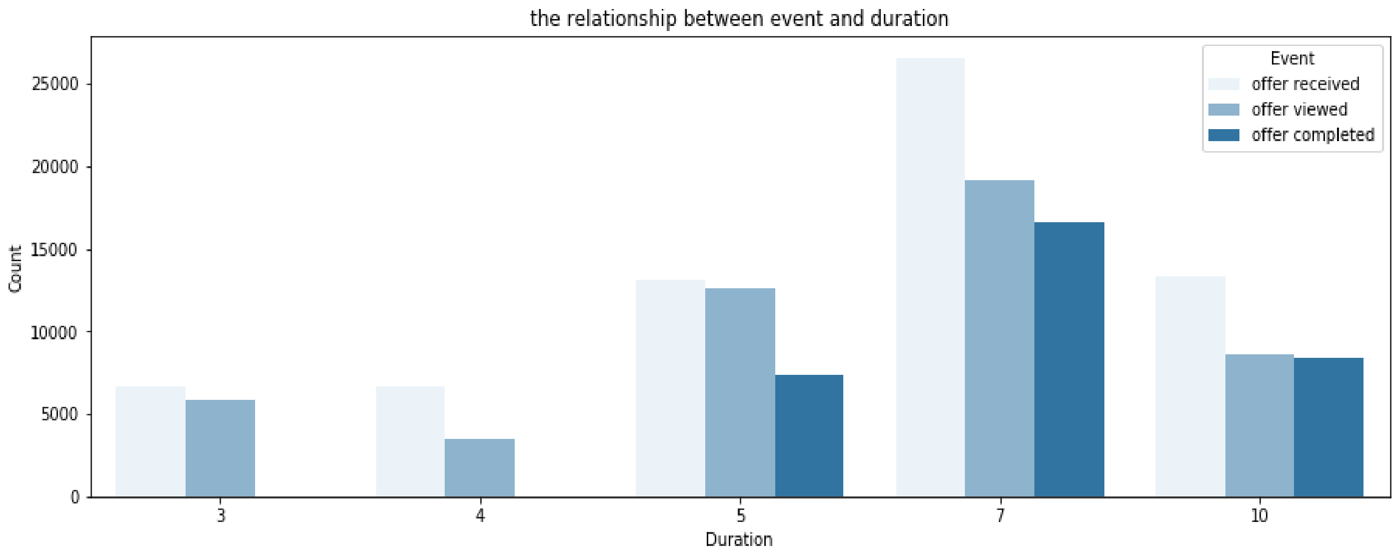
Cleaning steps:

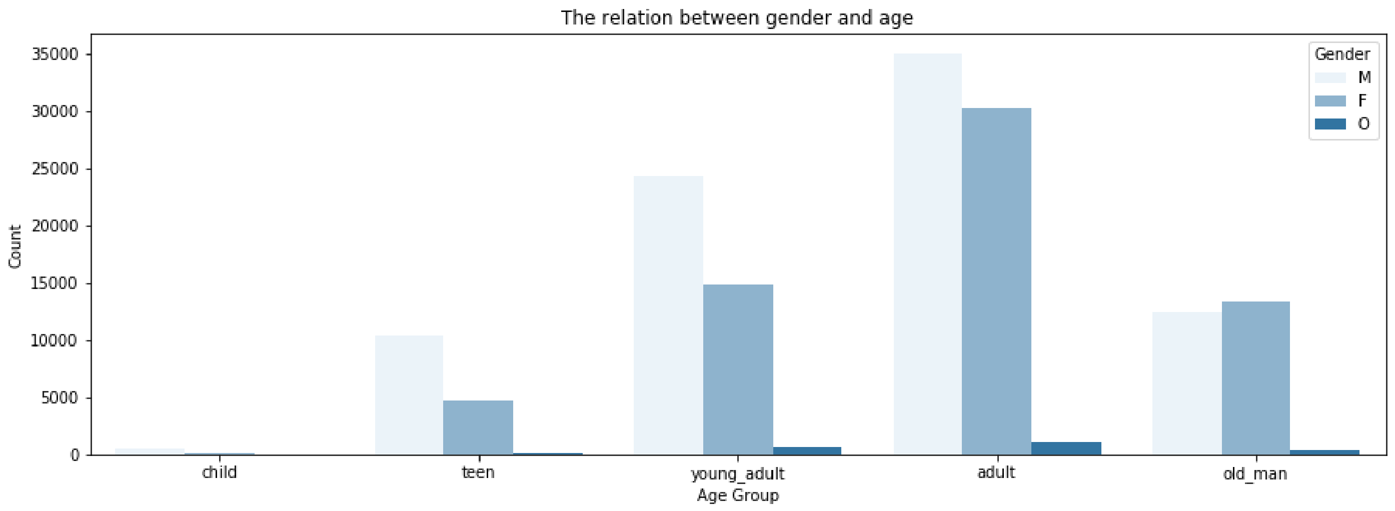
- Change the name of person column into customer_id
- split the column dictionary based on its keys for (offer_id, reward, amount)
- Drop the unnecessary columns

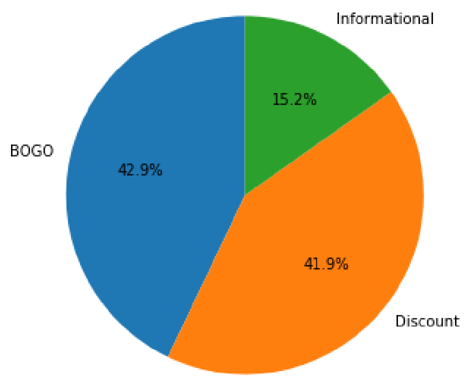
Data Exploration



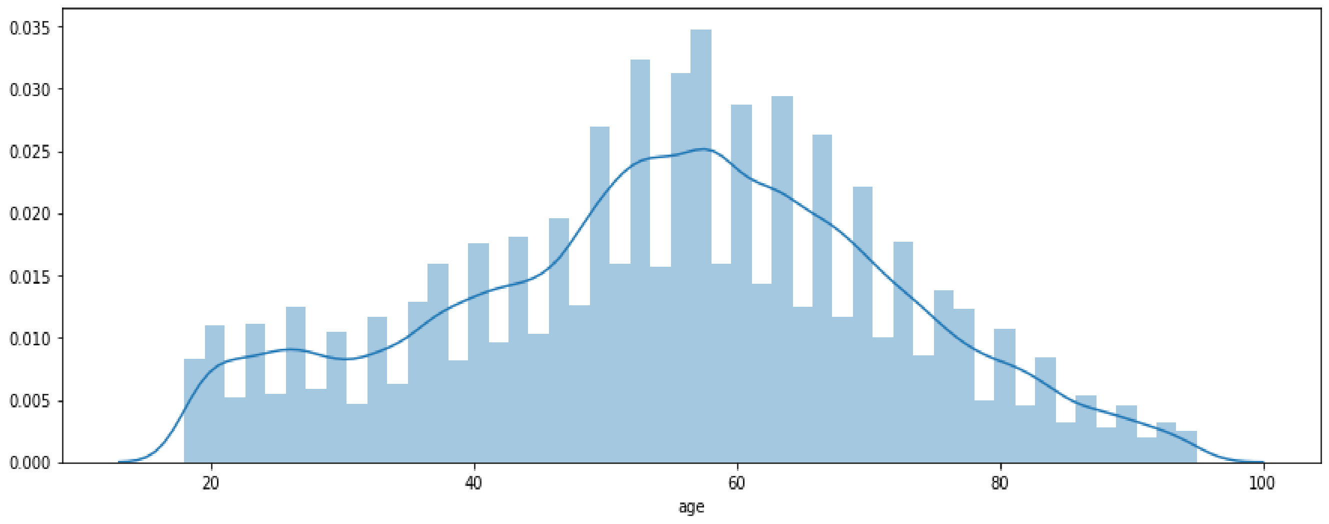


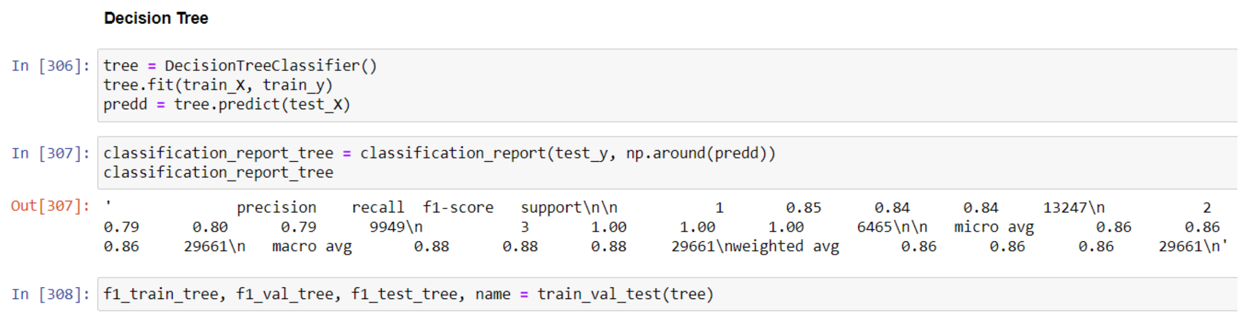






From Here we see that the most common gender is male and the most common age group is adults because of their work, I suppose and the most common Offer is BOGO and in second place Discount. The most common event is offer received.





Decision Tree

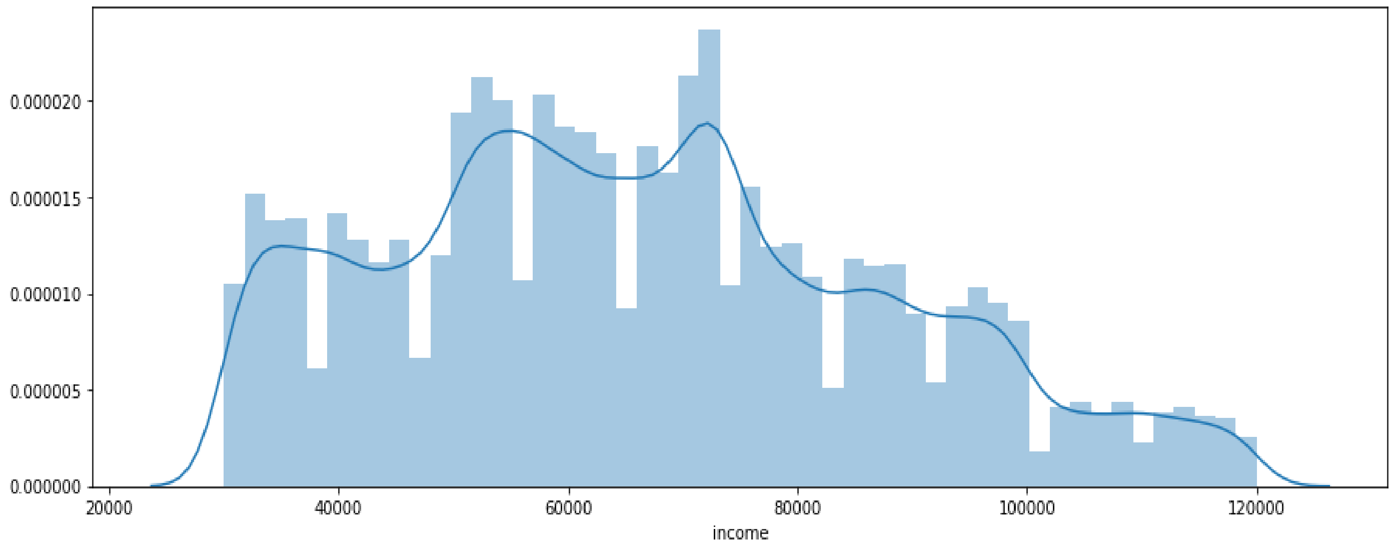
```
In [306]: tree = DecisionTreeClassifier()
          tree.fit(train_X, train_y)
          predd = tree.predict(test_X)

In [307]: classification_report_tree = classification_report(test_y, np.around(predd))
          classification_report_tree

Out[307]: '              precision    recall  f1-score   support\n\n           1       0.85      0.84      0.84     13247\n           2
          0.79      0.80      0.79      9949\n           3      1.00      1.00      1.00      6465\n\n   micro avg      0.86      0.86
          0.86     29661\n   macro avg       0.88      0.88      0.88     29661\nweighted avg       0.86      0.86      0.86     29661\n'

In [308]: f1_train_tree, f1_val_tree, f1_test_tree, name = train_val_test(tree)
```

Most of the customer are between 45 and 65, Most people use BOGO and discount offer type equally. The number of people review the offer are much more than those who complete it.

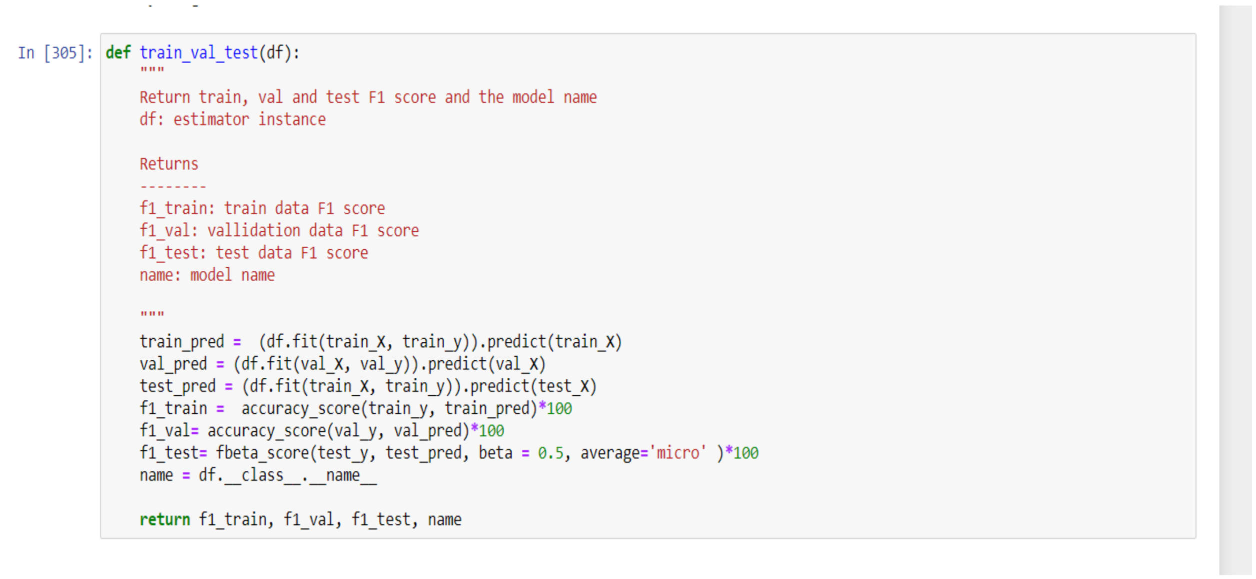


The income range Between 30000 and 120000 and reaches its highest at 70000.

Building the Model

Three different models with accuracy and f1-score being the primary metrics are build to predict whether a customer will respond to an offer. Before building models, our dataset should be split into train and test datasets so a model doesn’t overfit the data and also test dataset is used to evaluate how well our models are performing.

I used a function to apply it on every algorithm to get F1 score and accuracy of the algorithm used on the data



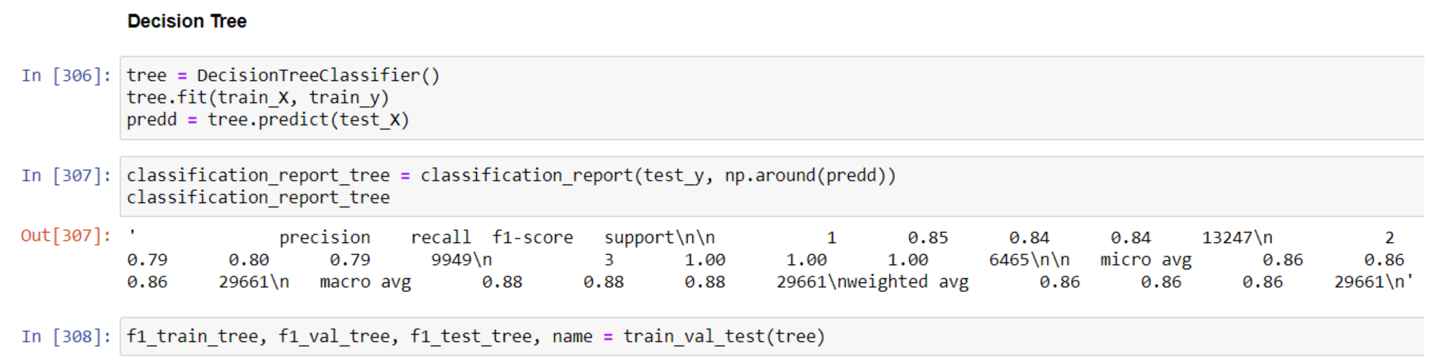
```
In [305]: def train_val_test(df):
              """
              Return train, val and test F1 score and the model name
              df: estimator instance

              Returns
              --------
              f1_train: train data F1 score
              f1_val: vallidation data F1 score
              f1_test: test data F1 score
              name: model name

              """
              train_pred =  (df.fit(train_X, train_y)).predict(train_X)
              val_pred = (df.fit(val_X, val_y)).predict(val_X)
              test_pred = (df.fit(train_X, train_y)).predict(test_X)
              f1_train =  accuracy_score(train_y, train_pred)*100
              f1_val= accuracy_score(val_y, val_pred)*100
              f1_test= fbeta_score(test_y, test_pred, beta = 0.5, average='micro' )*100
              name = df.__class__.__name__

              return f1_train, f1_val, f1_test, name
```

Decision Tree



Decision Tree

```
In [306]: tree = DecisionTreeClassifier()
          tree.fit(train_X, train_y)
          predd = tree.predict(test_X)

In [307]: classification_report_tree = classification_report(test_y, np.around(predd))
          classification_report_tree

Out[307]: '              precision    recall  f1-score   support\n\n           1       0.85      0.84      0.84     13247\n           2
          0.79      0.80      0.79      9949\n           3       1.00      1.00      1.00      6465\n\n   micro avg       0.86      0.86
          0.86     29661\n   macro avg       0.88      0.88      0.88     29661\nweighted avg       0.86      0.86      0.86     29661\n'

In [308]: f1_train_tree, f1_val_tree, f1_test_tree, name = train_val_test(tree)
```

K-Neighbours

K-Neighbours

```
In [312]: k_n_n = KNeighborsClassifier()
          k_n_n.fit(train_X, train_y)
          predk = k_n_n.predict(test_X)
```

```
In [313]: classification_report_k_n_n= classification_report(test_y, np.around(predk))
          classification_report_k_n_n
```

```
Out[313]: '              precision    recall  f1-score   support\n\n           1       0.39      0.60      0.47     13247\n           2
          0.20      0.16      0.18      9949\n           3       0.22      0.04      0.07      6465\n\n   micro avg       0.33      0.33
          0.33     29661\n   macro avg       0.27      0.27      0.24     29661\nweighted avg       0.29      0.33      0.29     29661\n'
```

```
In [314]: f1_train_knn, f1_val_knn, f1_test_knn, name_knn = train_val_test(k_n_n)
```

Logistic Regression

Logistic Regression

```
In [309]: from sklearn.linear_model import LogisticRegression

          LogReg = LogisticRegression()
          LogReg.fit(train_X, train_y)
          predl = tree.predict(test_X)
```

```
/home/ec2-user/anaconda3/envs/amazonei_mxnet_p36/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarnin
g: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/home/ec2-user/anaconda3/envs/amazonei_mxnet_p36/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:460: FutureWarnin
g: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
In [310]: classification_report_LogReg = classification_report(test_y, np.around(predl))
          classification_report_LogReg
```

```
Out[310]: '              precision    recall  f1-score   support\n\n           1       0.85      0.84      0.84     13247\n           2
          0.79      0.80      0.79      9949\n           3       1.00      1.00      1.00      6465\n\n   micro avg       0.86      0.86
          0.86     29661\n   macro avg       0.88      0.88      0.88     29661\nweighted avg       0.86      0.86      0.86     29661\n'
```

```
In [311]: f1_train_LogReg, f1_val_LogReg, f1_test_LogReg, name_LogReg = train_val_test(LogReg)
```

Random Forest

Random Forest

```
In [315]: import math
          Random = RandomForestClassifier(max_depth=10, random_state=0)
          Random.fit(train_X, train_y)
          predr = Random.predict(test_X)
```

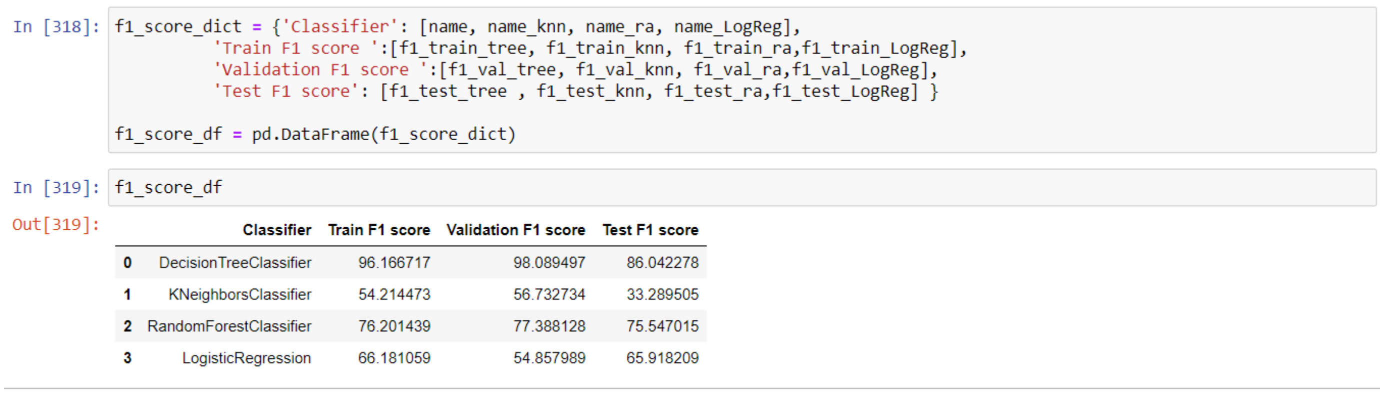
```
/home/ec2-user/anaconda3/envs/amazonei_mxnet_p36/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarning: The
default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
In [316]: classification_report_Random = classification_report(test_y, np.around(predr))
          classification_report_Random
```

```
Out[316]: '              precision    recall  f1-score   support\n\n           1       0.65      0.98      0.78     13247\n           2
          0.93      0.29      0.44      9949\n           3       1.00      1.00      1.00      6465\n\n   micro avg       0.76      0.76
          0.76     29661\n   macro avg       0.86      0.76      0.74     29661\nweighted avg       0.82      0.76      0.72     29661\n'
```

```
In [317]: f1_train_ra, f1_val_ra, f1_test_ra, name_ra = train_val_test(Random)
```

And now making a dataframe to show the results of every algorithm and comparing them.



```
In [318]: f1_score_dict = {'Classifier': [name, name_knn, name_ra, name_LogReg],
                    'Train F1 score ':[f1_train_tree, f1_train_knn, f1_train_ra,f1_train_LogReg],
                    'Validation F1 score ':[f1_val_tree, f1_val_knn, f1_val_ra,f1_val_LogReg],
                    'Test F1 score': [f1_test_tree , f1_test_knn, f1_test_ra,f1_test_LogReg] }

          f1_score_df = pd.DataFrame(f1_score_dict)
```

```
In [319]: f1_score_df
```

Out[319]:

	Classifier	Train F1 score	Validation F1 score	Test F1 score
0	DecisionTreeClassifier	96.166717	98.089497	86.042278
1	KNeighborsClassifier	54.214473	56.732734	33.289505
2	RandomForestClassifier	76.201439	77.388128	75.547015
3	LogisticRegression	66.181059	54.857989	65.918209

From these results we notice that Decision Tree Classifier is the best classification model for this problem then comes the Random Forest Classifier at second place then logistic regression and knn is the worst classification model. The decision tree classifier has the best f1 score on validation dataset with 98.2 but it decreases on the Test dataset to 86.1 but it still good classification. the random forest classifier got 77.45 on validation dataset but it increases on the Test dataset to 78. logistic regression got 66.8 on validation dataset but it decreases on the Test dataset to 65.93.KNN got 56.69.45 on validation dataset but it decreases on the Test dataset to 33.1 this is the worst classifier. so, the scores here a good enough to know if the customer will response to the offer or not.