Machine Learning Engineer Nanodegree

Capstone Project

I. Definition

- Starbucks Corporation is an American multinational chain of coffeehouses and roastery reserves headquartered in Seattle, Washington. As the world's largest coffeehouse chain, Starbucks is seen to be the main representation of the United States' second wave of coffee culture. Moreover Starbucks comes in fortune 500 companies and is ranked 227th in the year 2020.Starbucks have an mobile application which has various function and is used by the people to order online coffee via the app for pickup, pay for the purchase via the app and collect reward points.

- This app also provide a membership **"My Starbucks Rewards™ membership"**, after paying through the app the user receives free Stars/Bonus points that can be used to redeem a free drink of the user's choice.

- This app also offers various promotions to the users which includes Discount in a discount, a user gains a reward equal to a fraction of the amount spent on drinks ,BOGO (Buy One Get One Free) ,in a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount and Informational offer which basically includes any release of new product and there is no reward.

- But neither there is a requisite amount that the user is expected to spend. In this project the basic task is to use the data to identify which groups of people are most responsive to each type of offer, and how best to present each type of offer.

Problem Statement

We will be exploring the Starbuck's Dataset in which we will determine how people make purchasing decisions and how those decisions are influenced by promotional offers. The task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product. Some users might not receive any offer during certain weeks. Not all users receive the same offer, and that is the challenge to solve with this data set.

Metrics

I will use F1 score as an evaluation metrics in this case to determine which model will suite and performs better.The **F1 Score** is the 2*((precision*recall) / (precision + recall)). It is also called the F Score or the F Measure ,the F1 score conveys the balance between the precision and the recall.

lowest possible value of F1 score is 0 and Maximum value is 100, more the F1 score better the accuracy.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

TP= number of True Positives

FP= number of False positives

FN= number of false negatives

II. Analysis

Data Exploration

The data is contained in three files:

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

2) profile.json - demographic data for each customer

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

Here is the schema and explanation of each variable in the files:

portfolio.json
- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)

profile.json

- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income

transcript.json

- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

PORTFOLIO

1.Portfolio

In [2]:
```
portfolio.head()
```

Out[2]:

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

In [3]:
```
portfolio.shape
```

Out[3]: (10, 6)

In [4]:
```
portfolio.describe()
```

Out[4]:

	difficulty	duration	reward
count	10.000000	10.000000	10.000000
mean	7.700000	6.500000	4.200000
std	5.831905	2.321398	3.583915
min	0.000000	3.000000	0.000000
25%	5.000000	5.000000	2.000000
50%	8.500000	7.000000	4.000000
75%	10.000000	7.000000	5.000000
max	20.000000	10.000000	10.000000

In [5]:
```
portfolio.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
channels      10 non-null object
difficulty    10 non-null int64
duration      10 non-null int64
id            10 non-null object
offer_type    10 non-null object
reward        10 non-null int64
dtypes: int64(3), object(3)
memory usage: 560.0+ bytes
```

In [6]:
```
portfolio.columns
```

Out[6]: Index(['channels', 'difficulty', 'duration', 'id', 'offer_type', 'reward'], dtype='object')

PROFILE

2 Profile

In [11]:
```
profile.head()
```

Out[11]:

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

In [12]:
```
profile.shape
```

Out[12]: (17000, 5)

In [13]:
```
profile.columns
```

Out[13]: Index(['age', 'became_member_on', 'gender', 'id', 'income'], dtype='object')

In [14]:
```
profile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 5 columns):
age                 17000 non-null int64
became_member_on    17000 non-null int64
gender              14825 non-null object
id                  17000 non-null object
income              14825 non-null float64
dtypes: float64(1), int64(2), object(2)
memory usage: 664.1+ KB
```

In [15]:
```
profile.isnull().sum()
```

Out[15]:
```
age                    0
became_member_on       0
gender              2175
id                     0
income              2175
dtype: int64
```

In [16]:
```
profile.describe()
```

Out[16]:

	age	became_member_on	income
count	17000.000000	1.700000e+04	14825.000000
mean	62.531412	2.016703e+07	65404.991568
std	26.738580	1.167750e+04	21598.299410
min	18.000000	2.013073e+07	30000.000000
25%	45.000000	2.016053e+07	49000.000000
50%	58.000000	2.017080e+07	64000.000000
75%	73.000000	2.017123e+07	80000.000000
max	118.000000	2.018073e+07	120000.000000

In [18]:
```
#checking for gender
profile.gender.value_counts()
```

Out[18]:
```
M    8484
F    6129
O     212
Name: gender, dtype: int64
```

as we can see there are significantly more men that women we can now see that using the bargraph as well

TRANSCRIPT

3 transcript

In [27]:
```
transcript.head()
```

Out[27]:

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

In [28]:
```
transcript.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306534 entries, 0 to 306533
Data columns (total 4 columns):
event     306534 non-null object
person    306534 non-null object
time      306534 non-null int64
value     306534 non-null object
dtypes: int64(1), object(3)
memory usage: 9.4+ MB
```

In [29]:
```
transcript.shape
```

Out[29]: (306534, 4)

In [30]:
```
def create_offer_id_column(val):
    if list(val.keys())[0] in ['offer id', 'offer_id']:
        return list(val.values())[0]

def create_amount_column(val):
    if list(val.keys())[0] in ["amount"]:
        return list(val.values())[0]
```

In [31]:
```
transcript.shape
```

Out[31]: (306534, 4)

DATA CLEANING

Cleaning PORTFOLIO dataset

we will be passing the portfolio dataset in clean_portfolio which will return new dataset after one hot encoding

In [9]:
```
def clean_portfolio(df=portfolio):

    # One-hot encode channels column

    channels = portfolio["channels"].str.join(sep="*").str.get_dummies(sep="*")

    # One-hot encode offer_type column
    offer_type = pd.get_dummies(portfolio['offer_type'])

    # Concatinating one-hot and df
    new_df = pd.concat([df, channels, offer_type], axis=1, sort=False)

    # Removing channels and offer_type
    new_df = new_df.drop(['channels', 'offer_type'], axis=1)

    # Organizing columns
    columns = ["id", "difficulty", "duration", "reward", "email", "mobile", "social", "web", "bogo", "discount", "informational"]
    new_df = new_df[columns]

    return new_df
```

In [10]:
```
cleaned_portfolio= clean_portfolio()
cleaned_portfolio.head()
```

Out[10]:

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

Cleaning PROFILE dataset ¶

we will be passing the profile dataset in clean_profile which will return new dataset after one hot encoding

In [23]:
```
def clean_profile(profile = profile):

    # droping lines with income = NaN and age == 118(because null values are stored here)
    new_df = profile.drop(profile[(profile["income"].isnull()) & (profile["age"] == 118)].index)

    # One-hot encode Gender column
    gender_dummies = pd.get_dummies(new_df["gender"])

    # Specifying age range and one hot encoding
    range_ages = pd.cut(x=new_df["age"], bins=[18, 20, 29, 39, 49, 59, 69, 79, 89, 99, 102])
    # One-hot encode age column
    ages_dummies = pd.get_dummies(range_ages)

    # Specifying income range and one hot encoding

    range_income = pd.cut(x=new_df["income"], bins=[30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000,  120000])
    income_dummies = pd.get_dummies(range_income)


    # Concatinate
    new_df = pd.concat([new_df, ages_dummies, income_dummies, gender_dummies], axis=1, sort=False)

    # Dropping age,gender,income column from the dataset
    new_df = new_df.drop(["age", "gender", "income"], axis=1)

    return new_df
```

In [24]:
```
cleaned_profile = clean_profile()
cleaned_profile.head()
```

Out[24]:

	became_member_on	id	(18, 20]	(20, 29]	(29, 39]	(39, 49]	(49, 59]	(59, 69]	(69, 79]	(79, 89]	...	(50000, 60000]	(60000, 70000]	(70000, 80000]	(80000, 90000]	(90000, 100000]	(100000, 110000]
1	20170715	0610b486422d4921ae7d2bf64640c50b	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0
3	20170509	78afa995795e4d85b5d9ceeca43f5fef	0	0	0	0	0	0	1	0	...	0	0	0	0	1	0
5	20180426	e2127556f4f64592b11af22de27a7932	0	0	0	0	0	1	0	0	...	0	1	0	0	0	0
8	20180209	389bc3fa690240e798340f5a15918d5c	0	0	0	0	0	1	0	0	...	1	0	0	0	0	0
12	20171111	2eeac8d8feae4a8cad5a6af0499a211d	0	0	0	0	1	0	0	0	...	1	0	0	0	0	0

5 rows × 24 columns

cleaning TRASCRIPT dataset

we will be passing the profile dataset in clean_profile which will return new dataset after one hot encoding

In [32]:
```
def clean_transcript(transcript = transcript):

    #
    transcript['offer_id'] = transcript.value.apply(create_offer_id_column)
    transcript['amount'] = transcript.value.apply(create_amount_column)

    #  One hot encoding event column
    event = pd.get_dummies(transcript['event'])

    # Concatinating one hot and created dataframe
    new_df = pd.concat([transcript, event], axis=1, sort=False)

    # Create and Drop Transaction
    transaction = new_df[new_df["transaction"]==1]
    new_df = new_df.drop(transaction.index)

    # Drop
    new_df = new_df.drop(columns = ["event","value", "amount", "transaction"])

    return new_df
```

In [33]:
```
cleaned_transcript=clean_transcript()

cleaned_transcript.shape
```

Out[33]: (167581, 6)

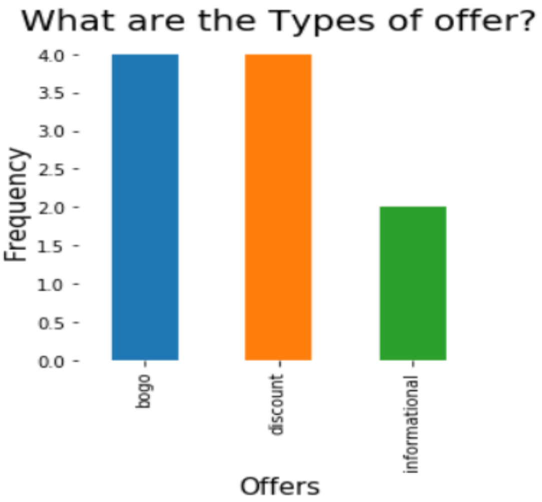
In [37]:
```
cleaned_transcript.head()
```

Out[37]:

	person	time	offer_id	offer completed	offer received	offer viewed
0	78afa995795e4d85b5d9ceeca43f5fef	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0	1	0
1	a03223e636434f42ac4c3df47e8bac43	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	1	0
2	e2127556f4f64592b11af22de27a7932	0	2906b810c7d4411798c6938adc9daaa5	0	1	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	0	fafdcd668e3743c1bb461111dcafc2a4	0	1	0
4	68617ca6246f4fbc85e91a2a49552598	0	4d5c57ea9a6940dd891ad53e9dbe8da0	0	1	0

DATA ANALYSIS AND VISUALIZATION

- Types of Offer Provided by the App

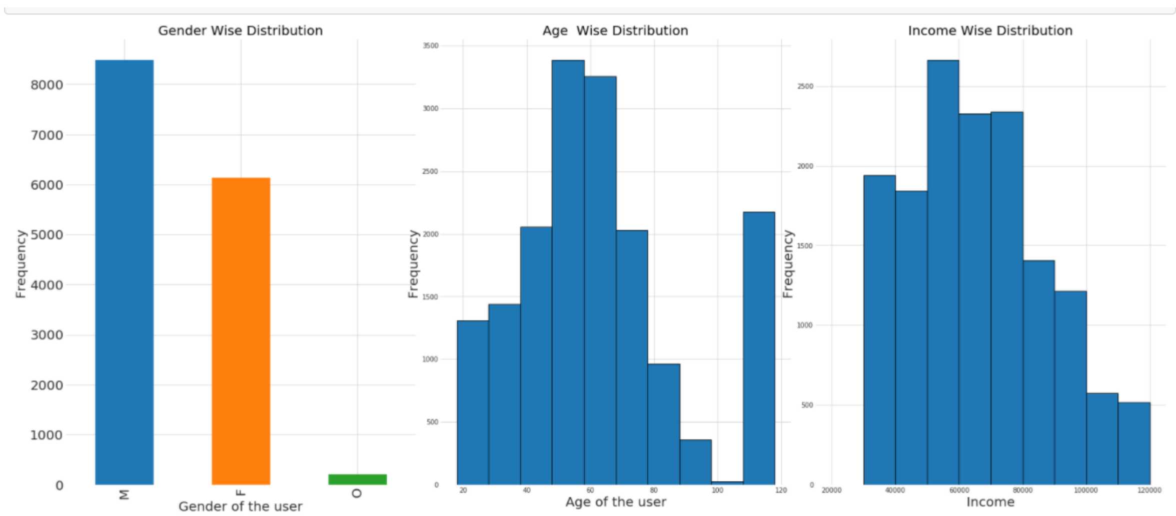


- Exploring Profile Dataset Gender / Age / Income Wise

Gender Wise: - We can see that there are more male than the females

Age Wise: - Majority of User are in range 20-100 and above 100 can be classified as outliners. Average of age users lies between 50-60

Income Wise: - Income range is from 30K to 120K and mean income comes in range 65K to 70k



- Exploring Cleaned Dataset on the basis of Gender / Age / Income Wise

In the cleaned dataset we have divided the dataset in groups so that it gives us the better understanding of the Dataset.

- **Exploring the Merged Dataset of Portfolio, Profile and Transcript**

In [48]:
```
merged_df.describe()
```

Out[48]:

	time	difficulty	duration	reward	email	mobile	social	web	bogo	discount
count	148805.000000	148805.000000	148805.000000	148805.000000	148805.0	148805.000000	148805.000000	148805.000000	148805.000000	148805.00000(
mean	354.570223	7.890561	6.625207	4.442445	1.0	0.917160	0.658311	0.806747	0.428978	0.418743
std	198.311301	5.041335	2.133035	3.372362	0.0	0.275641	0.474277	0.394851	0.494932	0.493355
min	0.000000	0.000000	3.000000	0.000000	1.0	0.000000	0.000000	0.000000	0.000000	0.000000
25%	168.000000	5.000000	5.000000	2.000000	1.0	1.000000	0.000000	1.000000	0.000000	0.000000
50%	408.000000	10.000000	7.000000	5.000000	1.0	1.000000	1.000000	1.000000	0.000000	0.000000
75%	510.000000	10.000000	7.000000	5.000000	1.0	1.000000	1.000000	1.000000	1.000000	1.000000
max	714.000000	20.000000	10.000000	10.000000	1.0	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 37 columns

In [51]:
```
#Total Records
t_records=len(merged_df.index)
print(t_records)
```

```
148805
```

In [52]:
```
#Number of offer completed
completed_offer = merged_df[merged_df["offer completed"] == 1].shape[0]
print(completed_offer)
```

```
32444
```

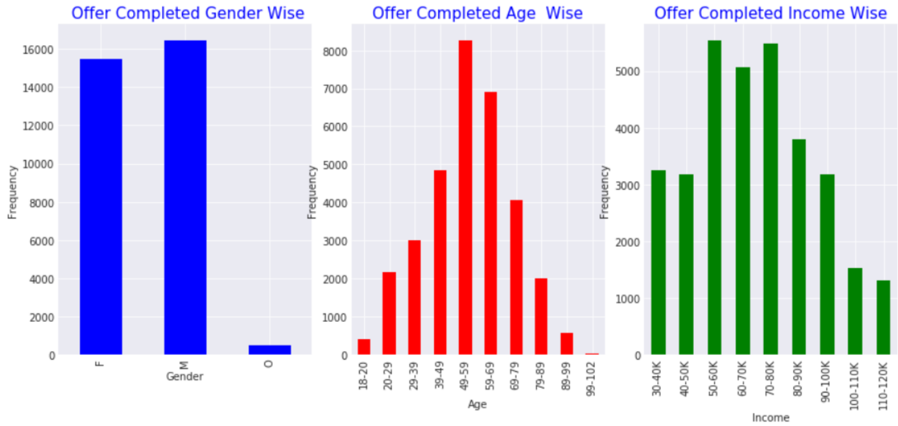
In [53]:
```
# Number of incomplete offer
incomplete_offer=merged_df[merged_df["offer completed"] == 0].shape[0]
print(incomplete_offer)
```

```
116361
```

In [54]:
```
# Percetage of offer completed
percent_completed = (completed_offer / t_records) * 100
print(percent_completed)
```

```
21.80303081213669
```

- Exploring Cleaned Dataset on the basis of Gender / Age / Income Wise

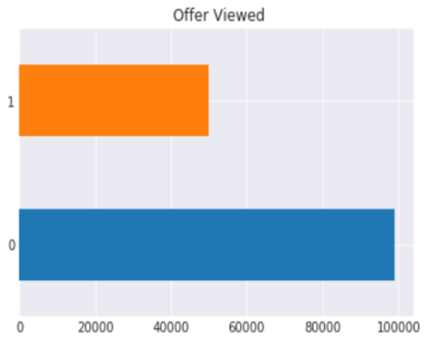


1.Based on Gender Female has completed more offers than male males have completed 16000+ offer whereas female has only completed 15000+ offers so we have to focus on M and F only and we can neglect Ither gender

2. Depicting from the subplot we can see that age range 49-59 have responded most to the offer and completed them

3. As we can see from the 3rd subplot people with income range 50-80k have completed the offer , we also saw a decrease in offer completion in 60-70k but that drop is comparable sso our target will be 50-80k.

- **Offer Received , Viewed and Claimed/completed**



1 on the Y axis represents the offer Received/Viewed/completed.

This implies most customers don't pay attention to the offer that they have recieved plus there are more number of customers who just view the offer and completely ignore it than the one's who actually completed the offer and redeem it.

- ## Which one is the most offer used by the use ?

In [63]:
```
#most used offer by customer

#calculating bogo
merged_df['bogo'].value_counts()
```

Out[63]:
```
0    84971
1    63834
Name: bogo, dtype: int64
```

In [64]:
```
merged_df['discount'].value_counts()
```

Out[64]:
```
0    86494
1    62311
Name: discount, dtype: int64
```

In [65]:
```
merged_df['informational'].value_counts()
```

Out[65]:
```
0    126145
1     22660
Name: informational, dtype: int64
```

from above we can clearly see that BOGO is the most popular of the all three and BOGO share a healthy competition with Discount offer.

Algorithms and Techniques

RandomForest Algorithm

Random forest is one of the best algorithm for classification and is widely used in day to day problems. It is an ensemble classifier that uses multiple Decision Trees to obtain better accuracy and performance.

It uses many classification trees and bootstrap sample technique is used to train each tree from the set of training data.This method only searches for a random subset of variables in order to obtain a split at each node.

The basic concept is that ‘weak learners’ come together to form a ‘strong learner’. It chooses the most voted prediction as the result. It also has the ability to handle larger input datasets when compared with other methods.

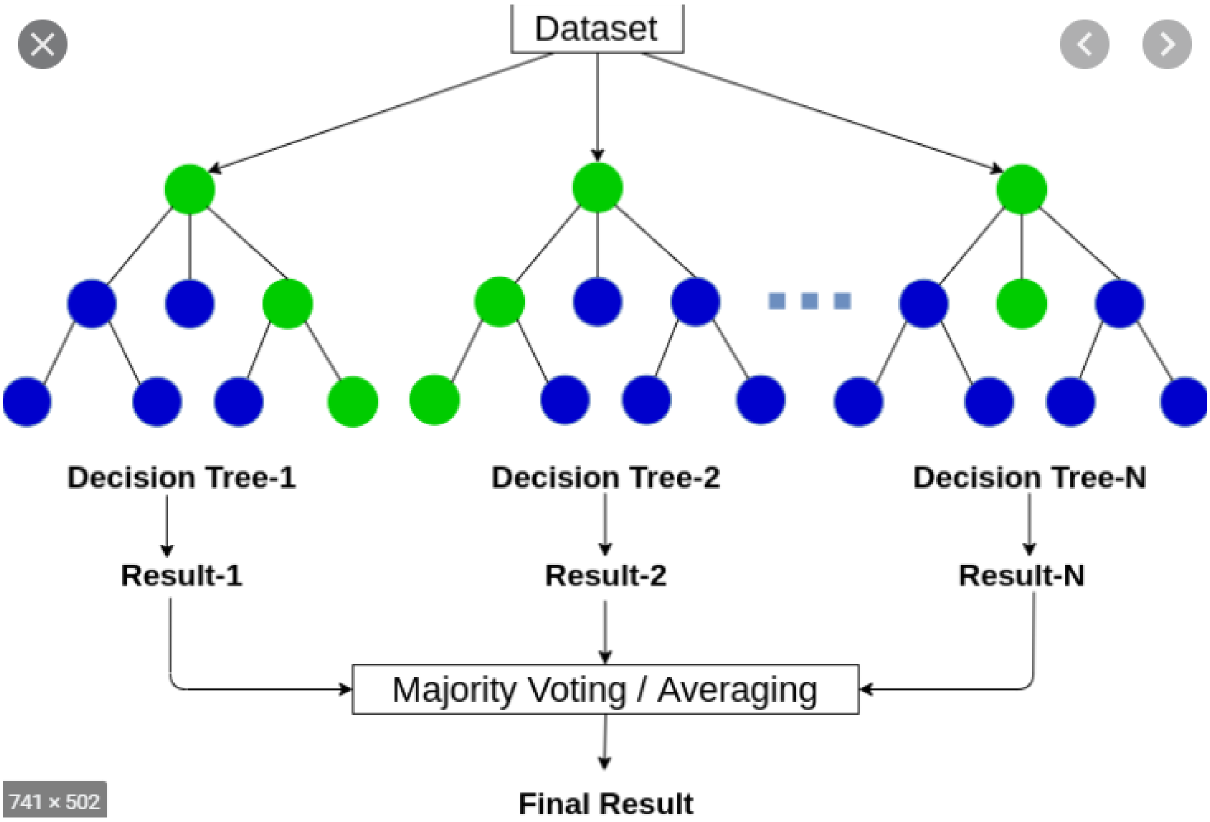


Image source: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Decision Tree Algorithm:

One of the most widely used and easily understandable algorithms which is easy to implement in machine learning. Decision tree works on the principle of splitting the dataset into small parts until the dataset is no longer splitable or the target variable is the same. The algorithm first assigns all training instances to the root of the tree and then splits the values based on the split feature with the criteria(In decision tree algorithm, gini index and information gain methods are used to calculate these nodes.). After which data is partitioned at nodes based on criteria and threshold, these partitioned values are called nodes and the above method is repeated until the tree is grown to full length or stopped forcefully.

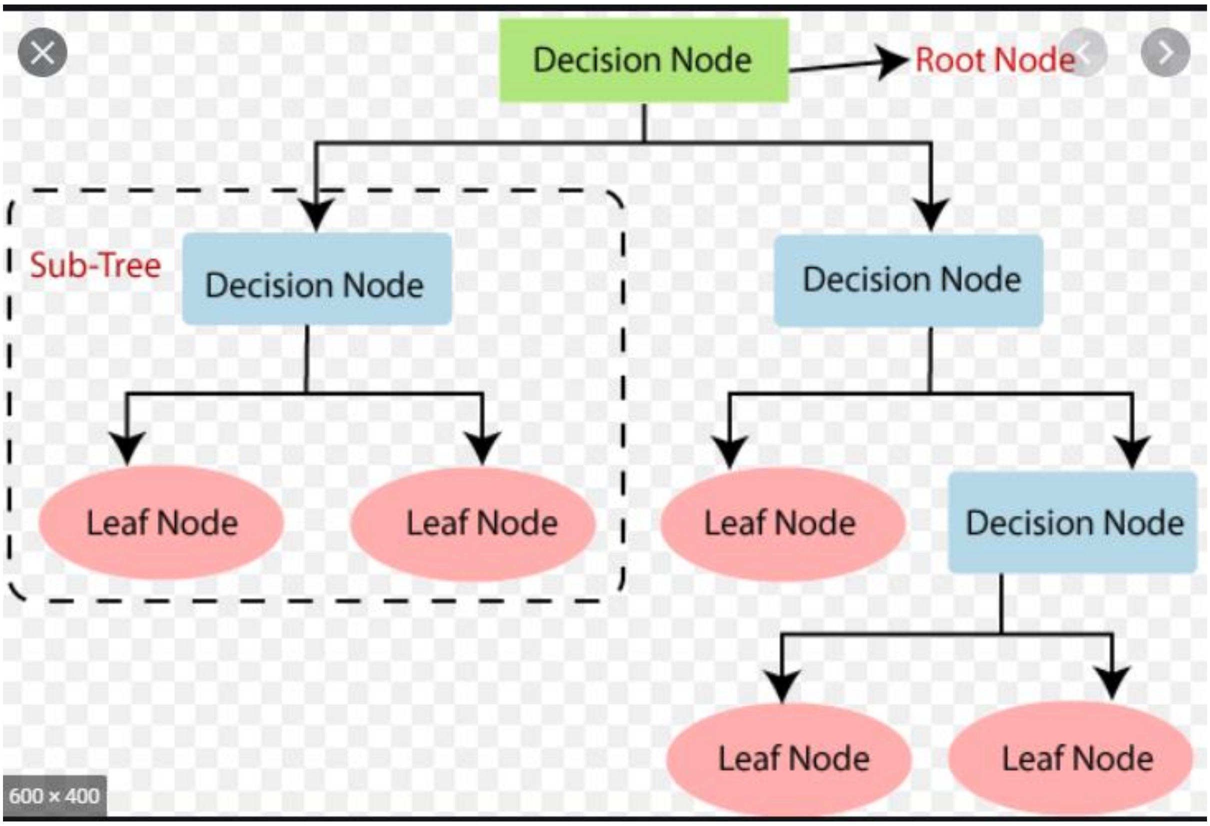


Image source: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Benchmark

As the data provided has both input and output this type of model comes in supervised learning, the model best suited for benchmark is KNeighbors Classifier as it is fast and accurate for this type of problem.

As we can see that Benchmark Model has shown a pretty good result as F1 on test score is around 80 out of 100. More the F1 score better the model.

	Benchmark Model	train F1 score	test F1 score
0	KNeighborsClassifier	84.58106	78.667814

III. Methodology

Data Preprocessing

- **One Hot Encoding :** Cleaning the portfolio , profile and transcript df and will be using One hot encoding method. One-Hot Encoding is another popular technique for treating categorical variables. It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. One-Hot Encoding is the process of creating dummy variables.

- **Train and Test Split:** The train test split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

  I decided to split the Dataset in the ratio of 25% Test and 75%Training Data Set. Training data will train the model and test data will be used for testing the model that is trained.

- **Normalization: Normalization** is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.
-

  Normalization is a rescaling of the data from the original range so that all values are within the new range of 0 and 1.

  I used the MinMaxScalar to Normalize the data. From the sklearn.preprocessing I imported the MinMaxScalar , after that I created a scalar object then scaled the Data.

In [70]:
```python
#Normalizing Data using MinMaxScalar
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # creating scalar object

numericals = features.columns[2:6]

features_scaled = pd.DataFrame(data = features)
features_scaled[numericals] = scaler.fit_transform(features[numericals])
```

In [71]:
```python
#checking scaled data
features_scaled.head()
```

Out[71]:

	person	offer_id	time	difficulty	duration	reward	email	mobile	social	web	...	O	30-40K	40-50K
0	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.000000	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
1	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.008403	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
2	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.184874	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
3	78afa995795e4d85b5d9ceeca43f5fef	f19421c1d4aa40978ebb69ca19b0e20d	0.705882	0.25	0.285714	0.5	1	1	1	1	...	0	0	0
4	78afa995795e4d85b5d9ceeca43f5fef	f19421c1d4aa40978ebb69ca19b0e20d	0.714286	0.25	0.285714	0.5	1	1	1	1	...	0	0	0

5 rows × 34 columns

In [72]:
```python
final_features=features_scaled[features_scaled.columns[2:]]
```

In [73]:
```python
final_features.head
```

Out[73]:
```
<bound method NDFrame.head of            time  difficulty  duration  reward  email  mobile  social  web  \
0      0.000000        0.25  0.571429     0.5      1       1       0    1
1      0.008403        0.25  0.571429     0.5      1       1       0    1
2      0.184874        0.25  0.571429     0.5      1       1       0    1
3      0.705882        0.25  0.285714     0.5      1       1       1    1
4      0.714286        0.25  0.285714     0.5      1       1       1    1
5      0.815126        0.25  0.285714     0.5      1       1       1    1
6      0.571429        0.50  0.571429     1.0      1       1       1    0
7      0.571429        0.50  0.571429     1.0      1       1       1    0
8      0.714286        0.50  0.571429     1.0      1       1       1    0
9      0.235294        0.00  0.000000     0.0      1       1       1    0
10     0.302521        0.00  0.000000     0.0      1       1       1    0
```

:
```python
#Normalizing Data using MinMaxScalar
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # creating scalar object

numericals = features.columns[2:6]

features_scaled = pd.DataFrame(data = features)
features_scaled[numericals] = scaler.fit_transform(features[numericals])
```


Implementation

1) First the Data was cleaned individually

2) Data was merged and analysed.

3) Conclusions were drawn on the basis of EDA

4)Data was prepared for the modelling-

Data was sent for Scaling in this case Normalization.

5) Splitting the dataset into test and train data set. The data set was divided into 25% Test and 75% Training dataset.

6) Creating the Benchmark model and checking it's F1 Score.

7) Passing the data to the model RandomForest and Decision Tree and calculate the result.

8) Compare the results and Select the best model.

Refinement

- When the data set was received it was not clean at all, the data was cleaned and then One hot encoding was performed which helped a lot in refinement of the model.
- Neglecting the Outliners has helped because it has helped because it did not tamper the results.
- Splitting the data set in the 25%-75% has helped and provided better result.
- F1 Score has helped a lot and has been a better metric and helped me determine a better model.

IV. Results

Model Evaluation and Validation

For model evaluation and validation Test data was used and test data has shown good results for model evaluation.

	Benchmark Model	train F1 score	test F1 score
0	KNeighborsClassifier	84.58106	78.667814

	Model	train F1 score	test F1 score
0	KNeighborsClassifier (Benchmark)	84.581060	78.667814
1	RandomForestClassifier	91.734989	78.786087
2	DecisionTreeClassifier	92.813813	79.710768

Justification

As we can clearly see *the final results found are stronger than the benchmark result.*

BenchMark Received the F1 Test score around 78.66 whereas both random forest as well as Decision tree have been proved to be better than the benchmark model.

- The reason for the model to perform better is that Splitting the data set in the 25%-75% Test Train has helped and provided better result because it had given the model the adequate amount of data for training and thus model has performed better.
- **Random forest** is an **ensemble** of **decision tree** algorithms. It is an extension of bootstrap aggregation (bagging) of **decision trees**, both of these are advanced algorithms and use better algorithms and thus overstand the benchmark model.

V. Conclusion

Reflection

- First of all I would like to add this project was quite interesting as well as challenging.
- Challenging as the data set that was given was imbalanced and cleaning the data set was not an easy task, moreover analysis of the data and selecting the important features was quite challenging as well.
- The main objective of this project was to build something practical so that it can be actually be used in practical world.
- The result obtained are quite promising and good as the model achieved an accuracy above 90%.

Improvement

The Accuracy could be improved if considered more factors and their interrelation.

Testing Additional Model as Light GBM, XG Boost etc.

Thank you