Starbucks_Project_Report

March 9, 2023

#

Capstone Project

#

José Estevan

##

Mar 08st, 2023

0.1 Data Science Nanodegree

#

Starbucks Capstone Project

##

Project Definition

1 Project Overview

This data set contains simulated data that mimics customer behavior on the Starbucksrewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks. Not all users receive the same offer, and that was the challenge to solve with this data set. Our task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products. Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. Informational offers also have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, we can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

1.1 Data Dictionary

profile.json Rewards program users (17000 users x 5 fields) * gender: (categorical) M, F, O, or null * age: (numeric) missing value encoded as 118 * id: (string/hash) portfolio.json Offers sent during

30-day test period (10 offers x 6 fields) * reward: (numeric) money awarded for the amount spent * channels: (list) web, email, mobile, social * difficulty: (numeric) money required to be spent to receive reward * duration: (numeric) time for offer to be open, in days * offer_type: (string) bogo, discount, informational * id: (string/hash) transcript.json Event log (306648 events x 4 fields) * person: (string/hash) * event: (string) offer received, offer viewed, transaction, offer completed * value: (dictionary) different values depending on event type * offer id: (string/hash) not associated with any "transaction" * amount: (numeric) money spent in "transaction" * reward: (numeric) money gained from "offer completed" * time: (numeric) hours after start of test

1.2 Offer Types

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. * In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. * In a discount, a user gains a reward equal to a fraction of the amount spent. * In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

2 Problem Statement

Forecasting the purchase offer to which a possible greater level of reaction or user activities such as "offer received," "offer seen," "transaction," and "offer completed" can be achieved based on the demographic parameters of the consumer and other qualities of the companies' buy offers.. ### Metrics: Accuracy is the quintessential classification metric. … And easily suited for binary as well as a multiclass classification problem. Accuracy = (TP+TN)/(TP+FP+FN+TN) Accuracy is the proportion of true results among the total number of cases examined. * became_member_on: (date) format YYYYMMDD * income: (numeric)

3 Data Exploration and Visualization

3.1 Data Cleaning

1. Portfolio Data

- create a copy of the original dataframe for further implementation .
- convert the column 'Channels' into 4 different channels on the basis of different types of channel .
- rename the column name from 'ID' to 'offer_id' .

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

2. Profile Data

- create a copy of the original dataframe for further implementation .
- convert the datatype of 'became_member_on' column and sort the date into proper format .
- change the column name from 'ID' to 'customer_id' .

	gender	age	customer_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

3. Transcript Data

- create a copy of the original dataframe for further implementation .
- change the column name from 'person' to 'customer_id' .
- convert the column 'Event' into 4 different columns on the basis of different types of event .
- convert the column 'Values' into 2 different columns .

	customer_id	time	rewards	amount	offer_id	offer_received	offer_viewed	offer_completed	transaction
0	78afa995795e4d85b5d9ceeca43f5fef	0	None	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	0	0	0
1	a03223e636434f42ac4c3df47e8bac43	0	None	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0	0
2	e2127556f4f64592b11af22de27a7932	0	None	NaN	2906b810c7d4411798c6938adc9daaa5	1	0	0	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	0	None	NaN	fafdcd668e3743c1bb461111dcafc2a4	1	0	0	0
4	68617ca6246f4fbc85e91a2a49552598	0	None	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	1	0	0	0

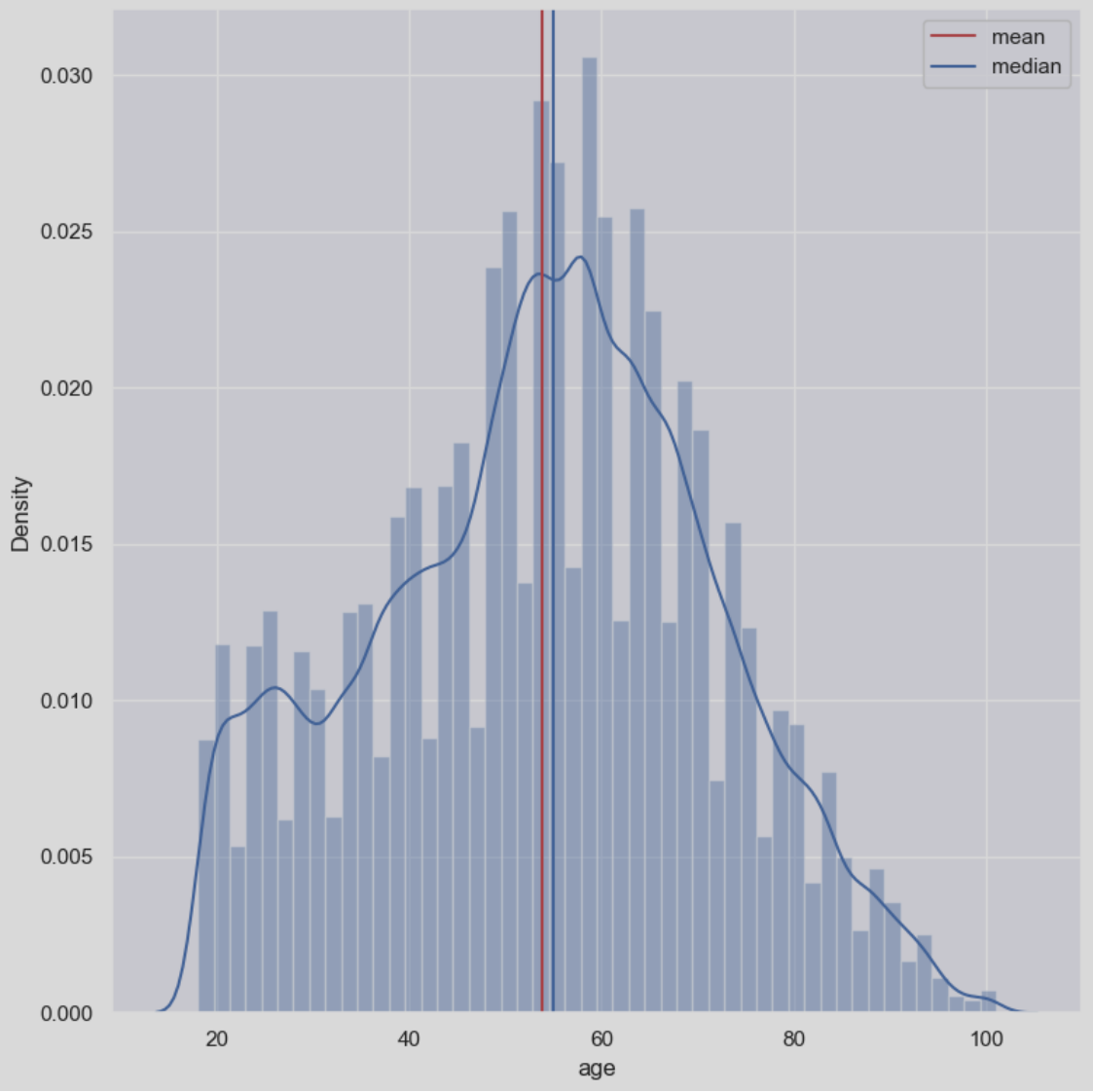
Final Dataset

- Concatenate all the three datasets together .
- Fixed the Offer_ids .
- fixed even_ids

	customer_id	event	time	offer-completed	offer-received	offer-viewed	transaction	offer_id	amount	gender	...	income	reward	difficulty	d
0	78afa995795e4d85b5d9ceeca43f5fef	offer-received	0	0	1	0	0	0.0	NaN	F	...	100000.0	5.0	5.0	
1	78afa995795e4d85b5d9ceeca43f5fef	offer-viewed	6	0	0	1	0	0.0	NaN	F	...	100000.0	5.0	5.0	
2	78afa995795e4d85b5d9ceeca43f5fef	transaction	132	0	0	0	1	NaN	19.89	F	...	100000.0	NaN	NaN	
3	78afa995795e4d85b5d9ceeca43f5fef	offer-completed	132	1	0	0	0	0.0	NaN	F	...	100000.0	5.0	5.0	
4	78afa995795e4d85b5d9ceeca43f5fef	transaction	144	0	0	0	1	NaN	17.78	F	...	100000.0	NaN	NaN	

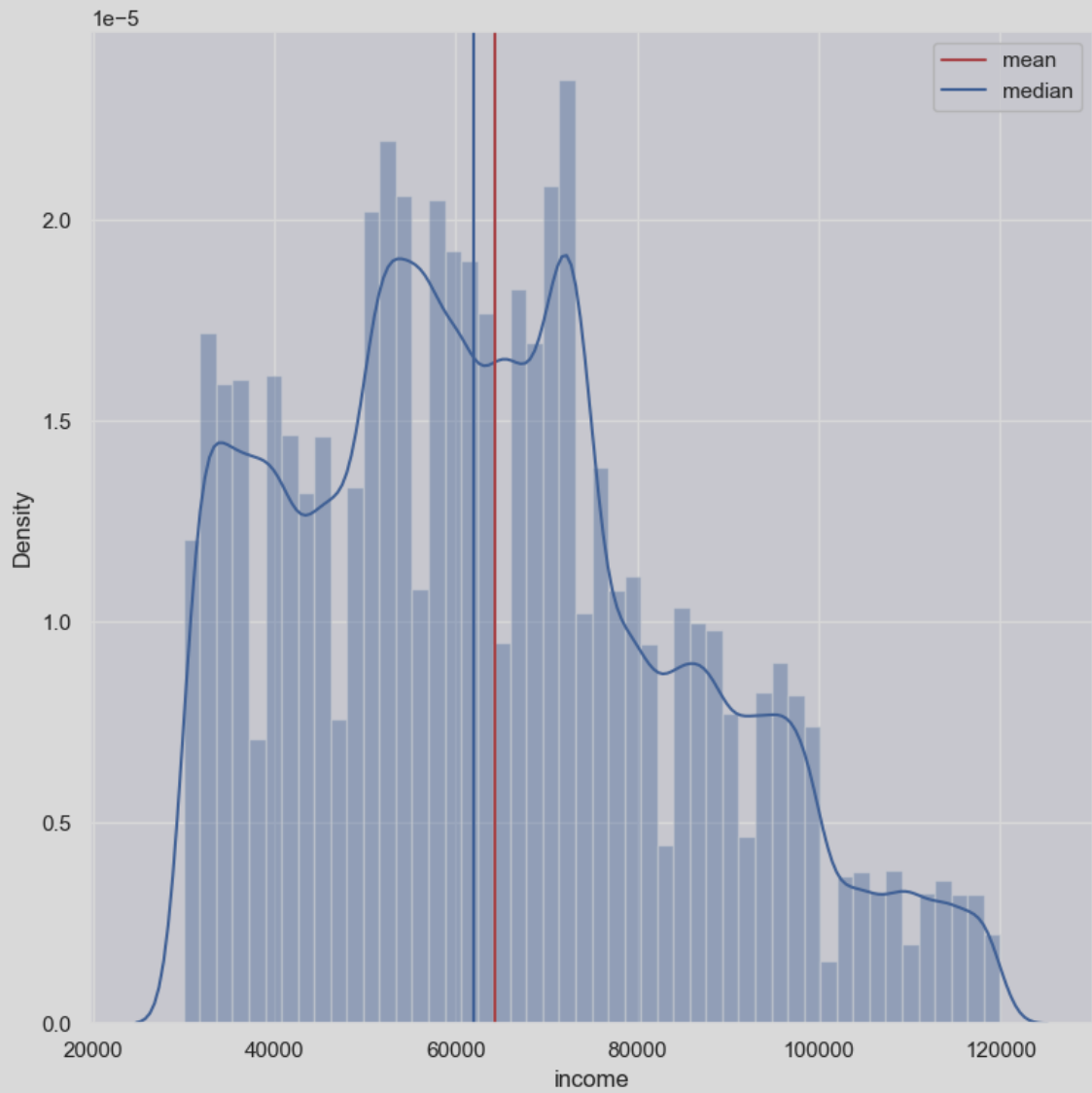
3.1.1 1. Age Group

- Average Aged user is middle age ie. around 50-62 years



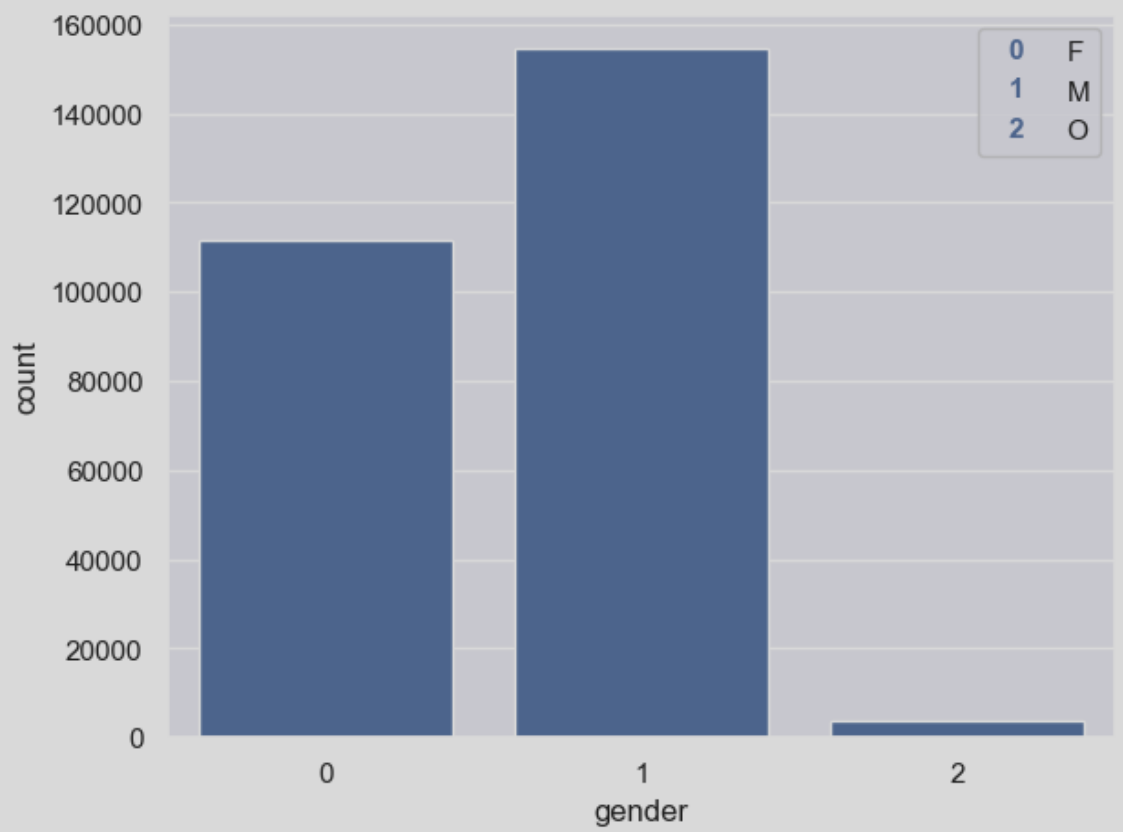
3.1.2 2 . Income Range

- Average income user is middle income group ie. 65000-70000



3.1.3 3. Gender :

- Males are more than 50 percent of users
- male_proportion = 50.79%,
- female_proportion = 36.89%,
- others_proportion = 1.29%



3.1.4 4. No. of Different Type of Offers Received by the Users

- BOGO offers are highly demanding , 30499 users received BOGO offers, 25449 viewed the offer and 15669 completed it.
- The percentage of BOGO Offer viewers is 83 percent .
- The percentage of DISCOUNT Offer viewers is 70 percent .

For BOGO Offer * Offer _Viewed_proportion = 83.44% , * Offer_Completed_proportion = 51.37%

4 Algorithms and Techniques

For this challenge we require a classification algorithm, it is a form of supervised learning that classifies new observations by making use of the training data it was given. A program will use the dataset or observations that are handed to it in order to learn how to classify fresh observations into a variety of classes or groupings. Take, for example, the choices 0 or 1, red or blue, yes or no, spam or not spam, etc. Classes can be described in a number of different ways, such by using targets, labels, or categories. Because it is a supervised learning technique that consists of both input and output information, the classification algorithm requires that the input data be labeled. In the process of classification, a discrete output function, denoted by y, is transformed into an input variable (x).

Classification, to put it in simpler terms, is a sort of pattern recognition in which classification algorithms are executed on training data in order to uncover the same pattern in fresh data sets.

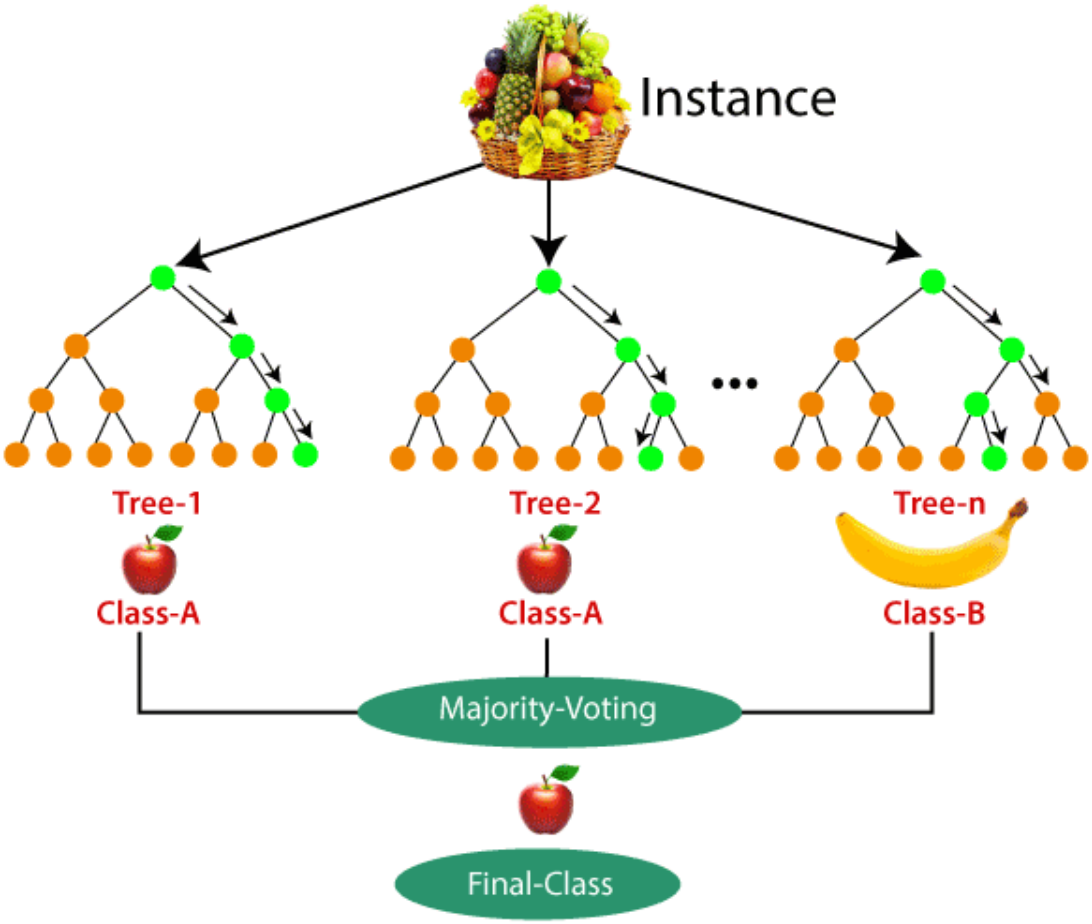
Inputs Source data fed into the classifier, with the goal of making a decision or prediction about the data.

4.0.1 Training Set

A set of inputs for which the correct outputs are known, used to train the classifier.

4.0.2 Random Forest Classifier

For this challenge we're going to use a Random Forest Classifier, the structure of a random forest is exactly what its name suggests: it is made up of a huge number of separate decision trees that work together as an ensemble. Every every tree in the random forest generates its own unique class prediction, and the class that receives the most votes ultimately serves as the basis for our model's prediction (see figure below).



4.0.3 Hyperparameters

A hyperparameter is a setting that affects the structure or operation of classifier. In real machine learning projects, tuning hyperparameters is the primary way to build an algorithm that provides accurate predictions for a certain problem. * n_estimators: the number of decision trees in the forest. Increasing this hyperparameter generally improves the performance of the model but also increases the computational cost of training and predicting. * max_depth: the maximum depth of each decision tree in the forest. Setting a higher value for max_depth can lead to overfitting while setting it too low can lead to underfitting.

5 Methodology

5.1 Data Preprocessing :

5.1.1 Splitting Dataset :

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.

```
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import cross_val_score
# from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
classifier = RandomForestClassifier()
```

```
y = df['respond_to_offer']
X = df.drop('respond_to_offer', axis=1)
```

```
scaler = MinMaxScaler()
normalize_col = list(X.columns)

X[normalize_col] = scaler.fit_transform(X[normalize_col])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((216026, 12), (54007, 12), (216026,), (54007,))
```

```
# training
classifier.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

5.2 Implementation :

5.2.1 Build a Model :

The implementation process can be split into two main stages: 1. The classifier training stage 2. The application development stage During the first stage, the classifier was trained on the preprocessed training data. 3. Define the algorithm and training parameters 4. Define the accuracy 5. Train the classifier, logging the validation accuracy 6. If the accuracy is not high enough, return to step 3

Initially, I implemented Random Forest Classifier with default parameters.

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
print("=== Classification Report ===")
print(classification_report(y_test, y_pred))
print('\n')
```

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     54007

    accuracy                           1.00     54007
   macro avg       1.00      1.00      1.00     54007
weighted avg       1.00      1.00      1.00     54007
```

Initial performance

6 Refinement :

6.1 Improving Prediction Model :

We have a overfitting problem, it happens whena model is unable to generalize and instead fits the training dataset in an excessively close manner, this is an example of overfitting. Overfitting can occur for a variety of causes, including the following: The training data size is insufficient and does not include sufficient data samples to accurately represent all of the possible input data values.

Now I will modify the hyperparameters in order to get a model that is more reliable and accurate.

```
rfc = RandomForestClassifier()
```

```
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [10,50,100,500,1000]
max_features = ['auto', 'sqrt']

# max depth
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)
# create random grid
random_grid = {
 'n_estimators': n_estimators,
 'max_features': max_features,
 'max_depth': max_depth
 }
```

```
# Random search of parameters
rfc_random = RandomizedSearchCV(estimator = rfc, param_distributions = random_grid,
                                n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
```

```
# Fit the model
rfc_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

```
print(rfc_random.best_params_)
```

Training Random Forest on best parameters

```
model = RandomForestClassifier(n_estimators=1000, max_depth=260, max_features='auto')
```

```
model.fit(X_train,y_train)
```

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

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     54007

    accuracy                           1.00     54007
   macro avg       1.00      1.00      1.00     54007
weighted avg       1.00      1.00      1.00     54007
```

Performance after tuning the hyperparameters

7 Results

7.1 Model Evaluation and Validation

During development , a validation set was used to evaluate the model. The final architecture and hyperparameters were chosen because they performed better.

8 Conclusion

- I found this project challenging, mainly due to the structure of the data in thetranscript dataset.
- Majority classes are performing well but the minorities are not.Problem of imbalanceddataset
- Most of the events are wrongly predicted as 'offer received'; offer received is the mostoccuring event or class.
- The main goal I chose was to build something practical the company could use tomake their choices more efficient.
- But the results of the model seem not so good . There is no change in rate of accuracy;it remains constant .

8.1 Improvement

- As we're using an imbalanced dataset we weren't expecting to get a greater accuracy .
- Analysing and building the models was Main challenging part and needed improvement .

Future Work: * There's a possibility of overfitting, which you can fix by taking into account extra data. * Build a Machine Learning Pipeline to seamlessly classify new data. * Make a Machine Learning model to predict transaction amount. * Deploy Machine Learning model to the web.

9 Resources

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.replace.html
- https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.merge.html
- https://stackoverflow.com/questions/37600711/pandas-split-column-into-multiple-columns-by-comma
- https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
- https://www.datacamp.com/tutorial/random-forests-classifier-python

- https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575

[ ]: