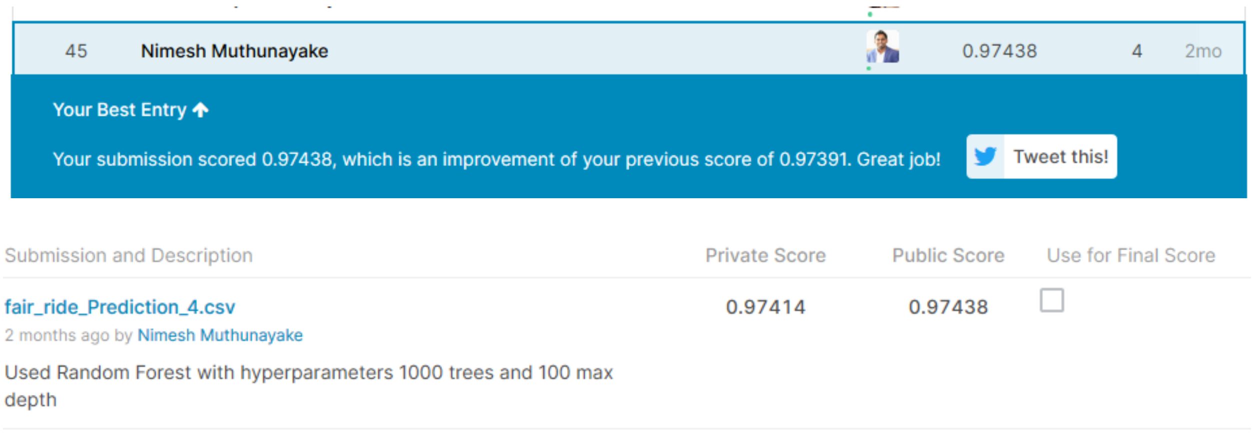
Name	Nimesha Asintha Muthunayake
Index No	209359G
Git Repository Link	https://github.com/nimeshacs/Fare-Classification-DS-MSc2020
Data Set	Ride Fare Classification

Details:

Kaggle Username	nimeshmuthunayake
Best result score	0.97438
public leaderboard rank	45
private leaderboard rank	45
Git Repository URL	https://github.com/nimeshacs/Fare-Classification-DS-MSc2020



45 Nimesh Muthunayake 0.97438 4 2mo

Your Best Entry

Your submission scored 0.97438, which is an improvement of your previous score of 0.97391. Great job!

Submission and Description	Private Score	Public Score	Use for Final Score
fair_ride_Prediction_4.csv 2 months ago by Nimesh Muthunayake Used Random Forest with hyperparameters 1000 trees and 100 max depth	0.97414	0.97438	☐

Fare Classification

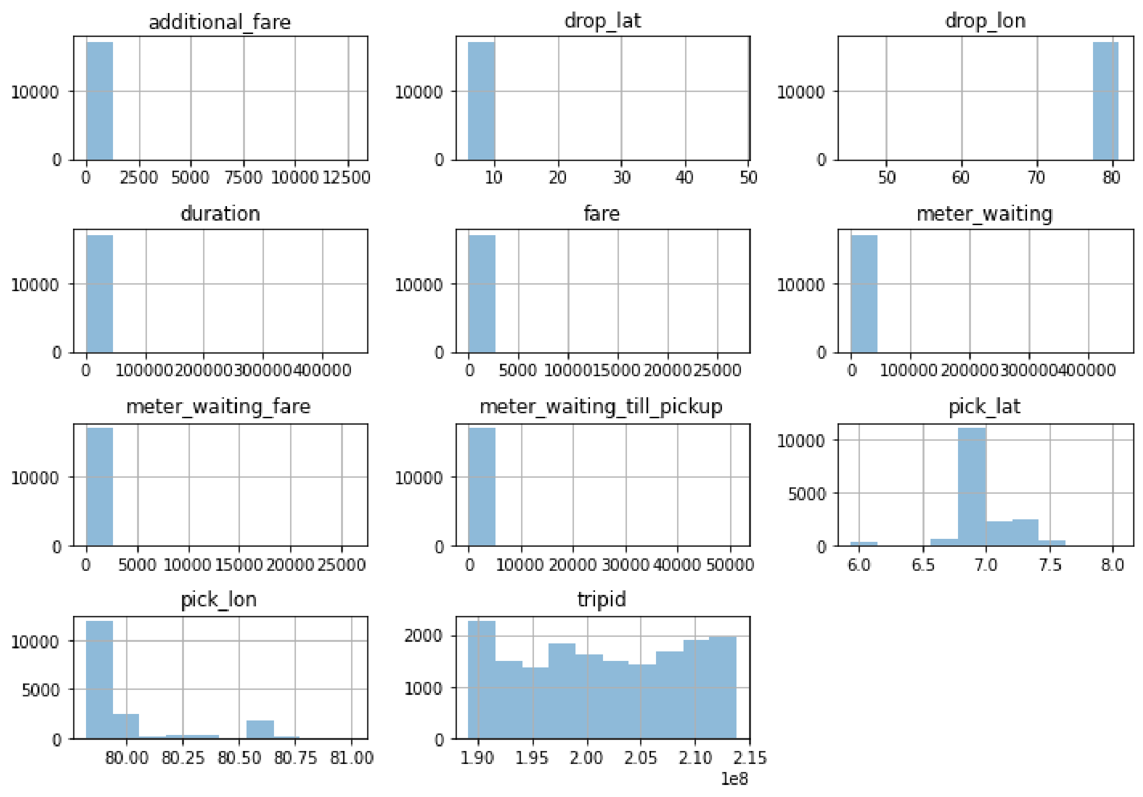
(1 - Provide a brief introduction to your solution)

It is difficult for individuals and organizations to estimate taxi trip fares using various dynamic conditions such as time , additional charges ,waiting time , pickup time and day, which affect the traffic conditions and starting location in a big city. Therefore, to address that concern we build Ride Faire Prediction model to identify the fare is valid or not (0 and 1)

First Look of Data (analysis of data)

(2 - Comment on feature engineering techniques you used)

If we look at the dimensions of the data there are 8,576 entries and 14 columns in the data set and I noticed several missing data, which is a great reminder that data collected in the real-world will never be perfect. hist() function is used to draw one histogram of the DataFrame's columns. A histogram is a representation of the distribution of data.



Data Preparation

Since we have empty (null) data we have to remove these NaN values and less important data columns form the learning model used df.dropna() to remove the NaN values and removed label,pickup_time,drop_time training data set

Classification Techniques

(3 - What classification techniques were tried?)

Tried with different type of algorithms to get maximum output for the classification problem , used **KNN** classifier first since KNN performs better with a lower number of features than a large number of features. But here it was not performed well and also **Decision Tree** algorithm is also used but it is also didn't perform well for this data set . the best accuracy is reported by **Random Forest** since It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

Random Forest for Classification

Random forests is a supervised learning algorithm. It can be used both for classification and regression and Random Forest Scikit-Learn API is used for this classification problem

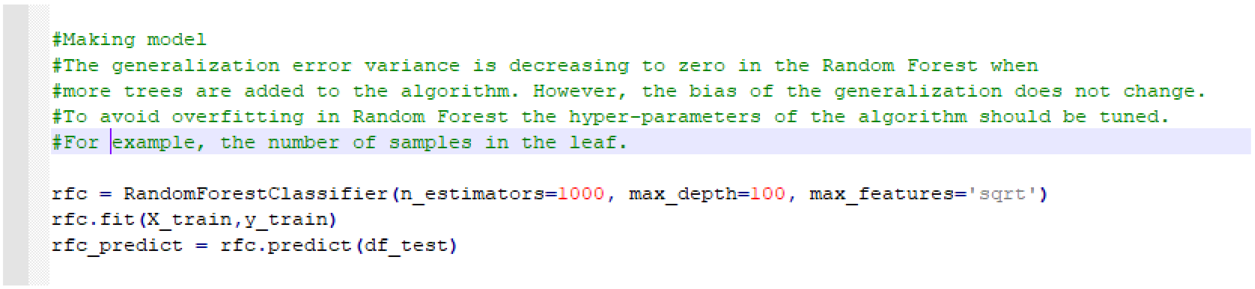
Overfitting and a Growing Number of Trees

(4 - What generalization techniques were used to avoid overfitting?)

An overfit model may look impressive on the training set but will be useless in a real-world application. Therefore, the standard procedure for hyperparameter optimization can be used for overfitting. Hyperparameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model. However, evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning: overfitting.

The generalization error variance is decreasing to zero in the Random Forest when more trees are added to the algorithm. However, the bias of the generalization does not change. To avoid overfitting in Random Forest the hyper-parameters of the algorithm should be tuned. For example, the number of samples in the leaf.

n_estimators	number of trees in the forest
max_features	max number of features considered for splitting a node
max_depth	max number of levels in each decision tree
min_samples_leaf	min number of data points allowed in a leaf node
min_samples_split	min number of data points placed in a node before the node is split
bootstrap	method for sampling data points (with or without replacement)



```
#Making model
#The generalization error variance is decreasing to zero in the Random Forest when
#more trees are added to the algorithm. However, the bias of the generalization does not change.
#To avoid overfitting in Random Forest the hyper-parameters of the algorithm should be tuned.
#For example, the number of samples in the leaf.

rfc = RandomForestClassifier(n_estimators=1000, max_depth=100, max_features='sqrt')
rfc.fit(X_train,y_train)
rfc_predict = rfc.predict(df_test)
```

Creating a Random Forest Model

Encoding Data

The first step for us is known as encoding of the data. This process takes categorical variables, mainly the label is converted into a numerical representation without an arbitrary ordering

```
df['label'].map({'correct': 1, 'incorrect': 0 })
```

Training and Testing Sets

(5 - Which sampling techniques were used?)

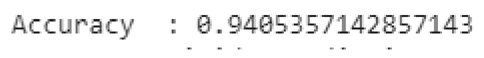
Generally, when training a model, we randomly split the data into training and testing sets to get a representation of all data points ,used 0.33% of data for the test set and used 0 as random state. random state values which performed well in the validation set do not correspond to those which would perform well in a new, unseen test set. Indeed, depending on the algorithm

After all the work of data preparation, creating and training the model is pretty simple using Scikit-learn and Instantiate model with 1000 decision trees with 100 Max depth

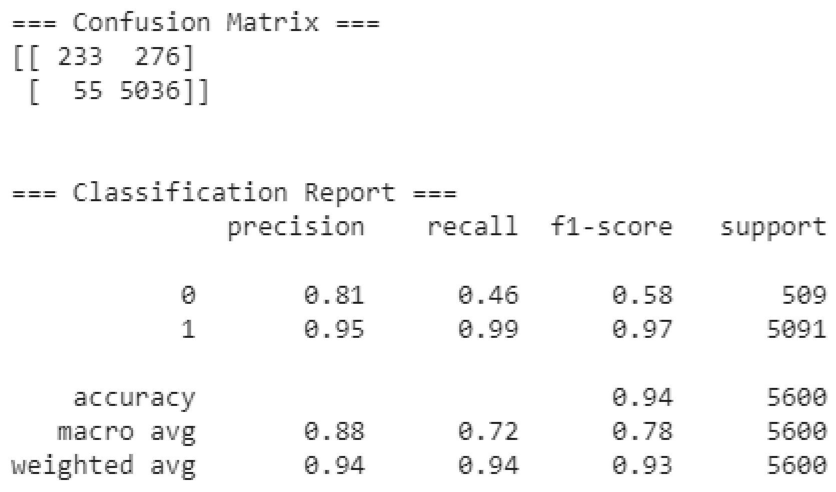
Evaluating Performance

In here in the data set we need to predict whether the fare is **correct (1)** or **incorrect(0).**
The confusion matrix is useful for giving the false positives and false negatives. The classification report tells the accuracy of the model.

The Accuracy of the model was **0.94**



```
Accuracy  : 0.9405357142857143
```



```
=== Confusion Matrix ===
[[ 233  276]
 [  55 5036]]


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

           0       0.81      0.46      0.58       509
           1       0.95      0.99      0.97      5091

    accuracy                           0.94      5600
   macro avg       0.88      0.72      0.78      5600
weighted avg       0.94      0.94      0.93      5600
```

Summary and Findings

(7 - Any noteworthy observations (or conclusion drawn) from the project.)

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase

- The Random Forest algorithm tend to be overfit .
- The generalization error variance is decreasing to zero in the Random Forest when more trees are added to the algorithm. the bias of the generalization does not change.
- Hyper-parameters of the algorithm should be tuned to avoid the overfit in the Random Forest algorithm . For example, the number of samples in the leaf.

-------------------- END --------------------