Coursework

Machine Learning

Lakshmanan Balasubramanian
MSc Advanced Computer Science
220237815

INDEX

Question 1

INTRODUCTION

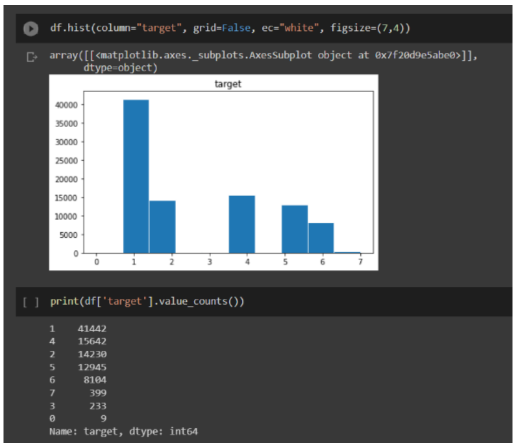
The dataset given here is obtained from the “Penn Machine Learning Benchmark (pmlb)” developed in the Computational Genetics Lab at University of Pennsylvania, USA. “PMLB” has lot of datasets and here we are using “Fatality Analysis Reporting System (fars)”.

EXPLORATION

The dataset explains about the US road traffic accidents. The dataset originally refers to the following data [https://sci2s.ugr.es/keel/dataset/data/classification/fars.zip] which has been processed and tuned to integers. It has 30 columns which describes about different attributes like *AGE*, *SEX*, *PERSON_TYPE*, *ALCOHOL_TEST_RESULT*, *DRUG_INVOLVEMENT*, *TAKEN_TO_HOSPITAL*, *RACE* and much more. From the given, the class label which is the “*target*” column describes about the severity of the accident. The “target” column has values ranging from 0 to 7. Given that the data is converted to the discrete values in the original dataset to numbers, it is quite easy to pre-process the data further. We don’t have to remove any columns as it was partially pre-processed where it might have lost lot of information during the pre-process. We will work with the data by doing the simplest/smallest pre-processing like excluding the duplicates and removing unused columns if applicable.

PREPROCESSING

The dataset obtained from the “PMLB” package is stored as a Data Frame to inspect the rows and columns further. The size of the rows and columns are noted, and all the available column names are also inspected. After the inspection, it is evident that there are some duplicate rows in the dataset. These duplicate data are removed from the dataset by picking out the duplicates using the “*duplicated* ()” function from ***pandas***. Now that we have a proper data frame with no duplicates and the dataset is ready to be pre-processed. Since the data was already converted to discrete values, we will perform a quick and simplest pre-processing. Before proceeding with the pre-processing, the target class label is explored. The number of individual values present in the “target” column is noted. The below image (*Figure 1*) refers the histogram of “target” class and the result of number of individual values present in the “target” class.



```
df.hist(column="target", grid=False, ec="white", figsize=(7,4))
```

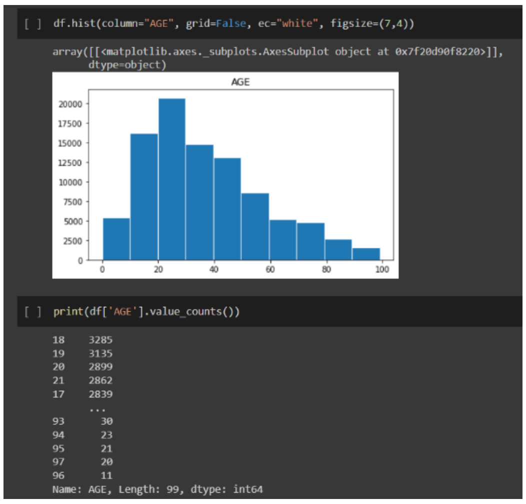
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f20d9e5abe0>]],
      dtype=object)
```

```
print(df['target'].value_counts())
```

```
1    41442
4    15642
2    14230
5    12945
6     8104
7      399
3      233
0        9
Name: target, dtype: int64
```

Figure 1

Similarly, the “AGE” column is also explored. This column is explored to see which age of people have involved with the accidents. The unique value in the column is also drawn. The below image (*Figure 2*) refers the histogram of “AGE” column and the result of number of individual values present in the column.



```
[ ]  df.hist(column="AGE", grid=False, ec="white", figsize=(7,4))

     array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f20d90f8220>]],
           dtype=object)
```

```
[ ]  print(df['AGE'].value_counts())

     18    3285
     19    3135
     20    2899
     21    2862
     17    2839
           ...
     93      30
     94      23
     95      21
     97      20
     96      11
     Name: AGE, Length: 99, dtype: int64
```

Figure 2

From the above histogram image, it is clearly seen that, the age group between 17 and 21 have involved in a greater number of accidents. The data flow of all the independent variable (input variable [X]) is checked. If the data flow is not a normal distribution, then the dataset is normalized. The Data normalization is done using “*MinMaxScaler*” from **sklearn** package. The below image (*Figure 3*) refers to the histogram of the data flow in the input variable (X).

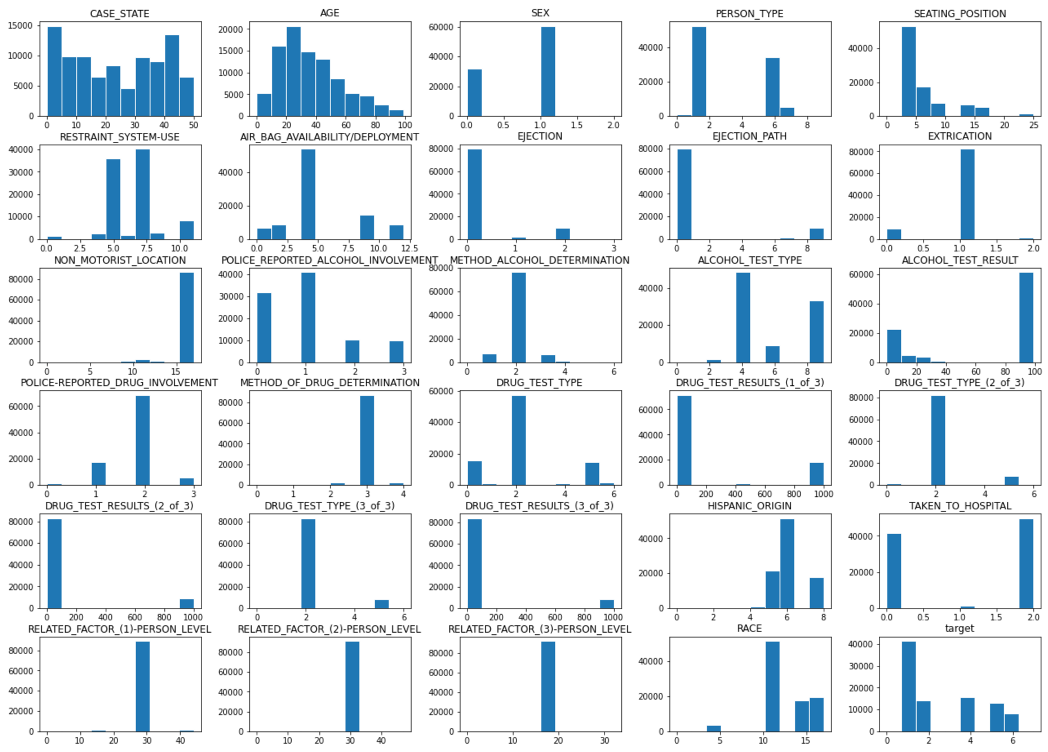
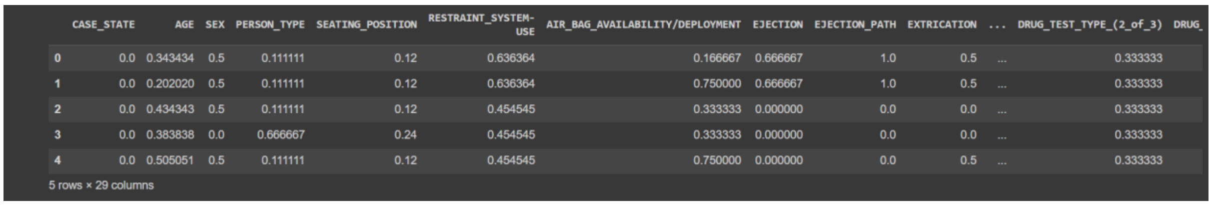


Figure 3

From the above visualization, apart from the "CASE_STATE" and "AGE" column the data distribution is not normal. Thus, the data are normalized using "*MinMaxScaler*" which is further converted to a numpy array. The below image (*Figure 4*) refers to the dataset being normalized and displayed in a Data frame.



	CASE_STATE	AGE	SEX	PERSON_TYPE	SEATING_POSITION	RESTRAINT_SYSTEM-USE	AIR_BAG_AVAILABILITY/DEPLOYMENT	EJECTION	EJECTION_PATH	EXTRICATION	...	DRUG_TEST_TYPE_(2_of_3)	DRUG_
0	0.0	0.343434	0.5	0.111111	0.12	0.636364	0.166667	0.666667	1.0	0.5	...	0.333333	
1	0.0	0.202020	0.5	0.111111	0.12	0.636364	0.750000	0.666667	1.0	0.5	...	0.333333	
2	0.0	0.434343	0.5	0.111111	0.12	0.454545	0.333333	0.000000	0.0	0.0	...	0.333333	
3	0.0	0.383838	0.0	0.666667	0.24	0.454545	0.333333	0.000000	0.0	0.0	...	0.333333	
4	0.0	0.505051	0.5	0.111111	0.12	0.454545	0.750000	0.000000	0.0	0.5	...	0.333333	

5 rows × 29 columns

Figure 4

Using the concept of **Feature Selection,** the normalized data frame is pre-processed by removing unwanted/unused columns. This is made into a new data frame since the unused columns might be added back if needed. The columns "RACE" and "HISPANIC_ORIGIN" are dropped. Since both the column deals with the colour and ethnicity of the person which has no impact in accident dataset. Thus, we have a clean and processed data ready to be trained for the model and results.

MODEL TRAINING AND INTERPRETATION

Using the "*Train_Test_Split*", the data is split into training and testing data. After splitting, we train with the training set and test with the test set. This will provide a more accurate evaluation because the testing dataset is not part of the dataset that has been used to train the model. Different classification models are used to get the best model.

The following models are used in this problem.

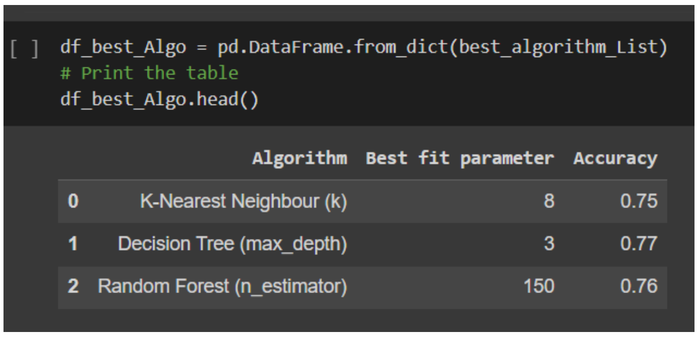
- K – Nearest Neighbour (KNN)
- Decision Tree
- Random Forest

First, we inspect the outcomes of the K – Nearest Neighbour (KNN) algorithm. With k = 3 as the default, the model is trained with the data that was split using "*Train_Test_Split*". The test set accuracy of 62% was obtained. The accuracy is below average and thus with the help of the cocept **hyper parameter tuning**, the best k is obtained. After parameter tuning, the best K was found out to be 9 with an accuracy of 64%. Yet, the accuracy was below average. As a result, the KNN didn't provide much of accuracy.

Secondly, we inspect the outcomes of Decision Tree algorithm. Using the training and testing data we fit the data into the model to notice the outcome of Decision tree. The accuracy obtained was 60%. Then, using the hyper parameter tuning to find the best "*max_depth*", the accuracy obtained was 61%. Thus, the Decision Tree also didn't provide much of accuracy.

Finally, we inspect the outcomes of Random Forest algorithm. Same as the previous model, we use the training and testing data that was fitted into the model to notice the outcome. The accuracy obtained after hyper parameter tuning was 63%.

Thus, all the three models didn't provide much of accuracy, and it was below average (*Figure 5*), we decided to add the two columns that was dropped earlier.



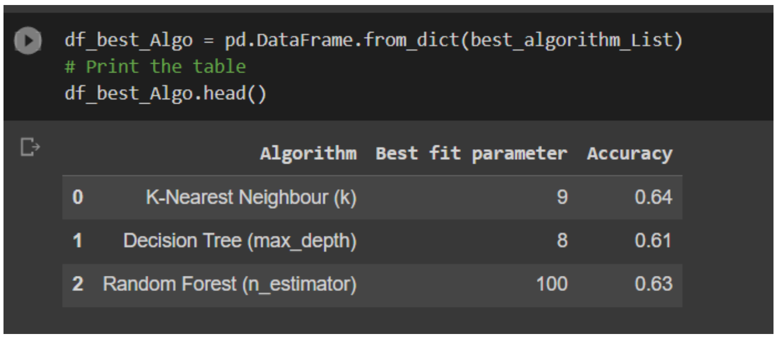
```
[ ] df_best_Algo = pd.DataFrame.from_dict(best_algorithm_List)
    # Print the table
    df_best_Algo.head()
```

	Algorithm	Best fit parameter	Accuracy
0	K-Nearest Neighbour (k)	8	0.75
1	Decision Tree (max_depth)	3	0.77
2	Random Forest (n_estimator)	150	0.76

Figure 5

After including the columns, same steps are repeated, all the three models are interpreted again with the training and testing data. The hyper parameter tuning is also performed to get the best output.

As a result, we get the following image (*Figure 6*) as the output.



```
df_best_Algo = pd.DataFrame.from_dict(best_algorithm_List)
# Print the table
df_best_Algo.head()
```

	Algorithm	Best fit parameter	Accuracy
0	K-Nearest Neighbour (k)	9	0.64
1	Decision Tree (max_depth)	8	0.61
2	Random Forest (n_estimator)	100	0.63

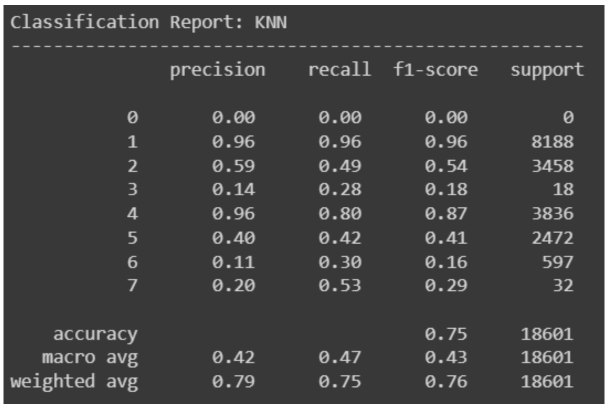
Figure 6

From the above image, it is evident that adding the two columns back gave us a decent and average accuracy. Also, out of all the three models, it was **Decision Tree** that had the best accuracy.

EVALUATION

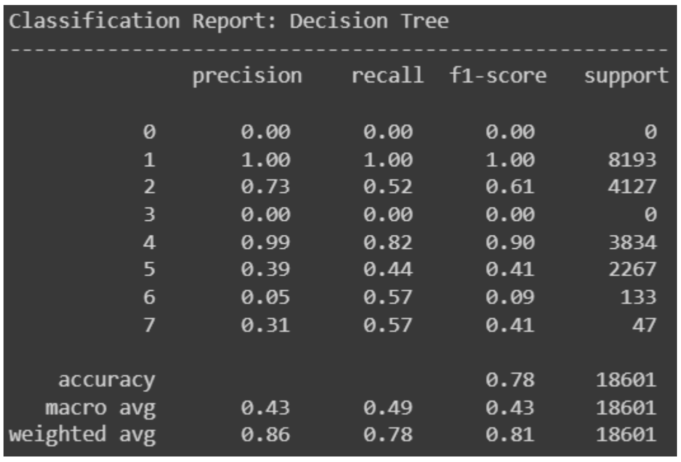
Using the Classification report, the model was evaluated. The classification report states the precision, recall, F1 score and support of the model.

Classification report of KNN:



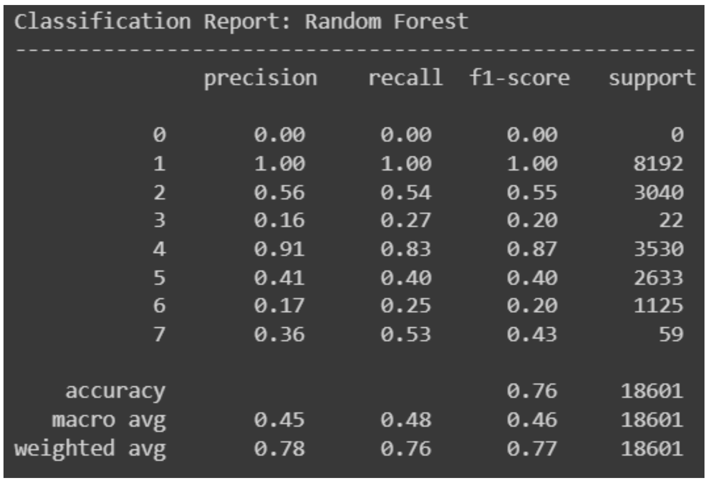
```
Classification Report: KNN
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.96      0.96      0.96      8188
           2       0.59      0.49      0.54      3458
           3       0.14      0.28      0.18        18
           4       0.96      0.80      0.87      3836
           5       0.40      0.42      0.41      2472
           6       0.11      0.30      0.16       597
           7       0.20      0.53      0.29        32

    accuracy                           0.75     18601
   macro avg       0.42      0.47      0.43     18601
weighted avg       0.79      0.75      0.76     18601
```

Classification report of Decision Tree:



```
Classification Report: Decision Tree
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      1.00      1.00      8193
           2       0.73      0.52      0.61      4127
           3       0.00      0.00      0.00         0
           4       0.99      0.82      0.90      3834
           5       0.39      0.44      0.41      2267
           6       0.05      0.57      0.09       133
           7       0.31      0.57      0.41        47

    accuracy                           0.78     18601
   macro avg       0.43      0.49      0.43     18601
weighted avg       0.86      0.78      0.81     18601
```

Classification report of Random Forest:



```
Classification Report: Random Forest
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      1.00      1.00      8192
           2       0.56      0.54      0.55      3040
           3       0.16      0.27      0.20        22
           4       0.91      0.83      0.87      3530
           5       0.41      0.40      0.40      2633
           6       0.17      0.25      0.20      1125
           7       0.36      0.53      0.43        59

    accuracy                           0.76     18601
   macro avg       0.45      0.48      0.46     18601
weighted avg       0.78      0.76      0.77     18601
```

DISCUSSION

With the evaluated reports we knew that the Decision tree had the best accuracy. Precision refers to the accuracy of positive predictions. Recall refers to the fraction of positives that were correctly identified. F1 score refers to the mean of precision and recall. Support refers to the number of actual occurrences. Although, the accuracy of other two models are almost near to the Decision Tree, the precision score was better in Decision tree than the other two models.

Question 3

INTRODUCTION

The dataset provided here are the twitter airline sentiment analysis. There are three datasets provided, they are: Training, Testing and Validation dataset. Each sample in the dataset represents a tweet. Each tweet has a sentiment label (Positive, Negative, Neutral).

EXPLORATION

The dataset refers to twitter data regarding to Airlines/Airlines customer conversation where it has the tweets and the connotation. The connotation refers to the Negative, Positive and Neutral connotation. Each tweet associates to either of the three connotation which are Positive, Negative and Neutral. The dataset has three columns "tweet_id" which refers to the unique id of the tweet. "text" which contains the tweet and "airline_sentiment" which refers to the connotation whether "Positive", "Negative" or "Neutral". Although, There is nothing much to drop the columns since it has only 3 columns, we can drop column "tweet_id" since it is not necessary. We will work with the data by doing the simplest text processing like excluding the expressions like "@, #, /, ",', \". Then, we will use various classification model to choose the best model.

PREPROCESSING

We have three datasets in total which are Training, Testing and Validation. With the help of exploring the data, all the data has same columns and different data in rows. We proceed with pre-processing the data all the three datasets. Initially, we make it to a Data frame and explores the data as a table. Then, the unwanted columns are dropped. In term of twitter data, the unwanted column is "tweet_id", thus we drop the column in all the three datasets. Now, we start with text-processing the text data in the column "Text". Some of the text processing includes removing expressions, username, emojis if any, punctuations and separating alphanumeric. Once the text is pre-processed, we investigate the number of each sentiment in the dataset. The below images (*Figure 7,8* & *9*) show the number of each sentiment in both training and testing dataset.

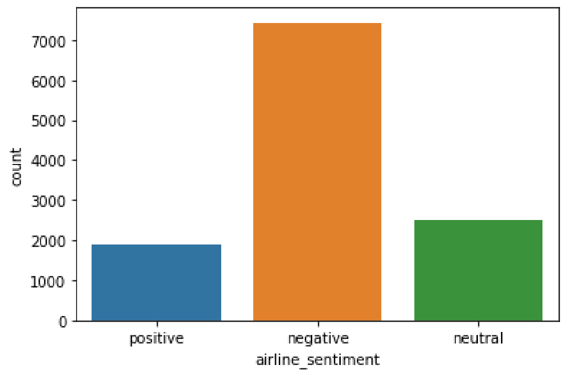


Figure 7 Training dataset

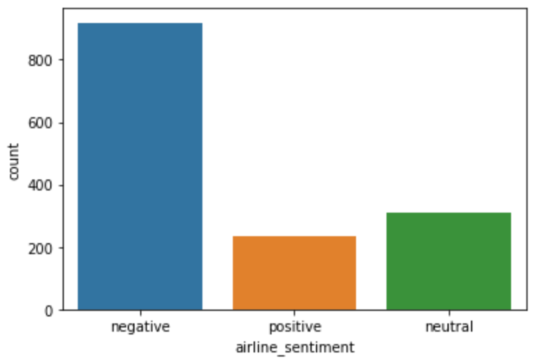


Figure 8 Testing dataset

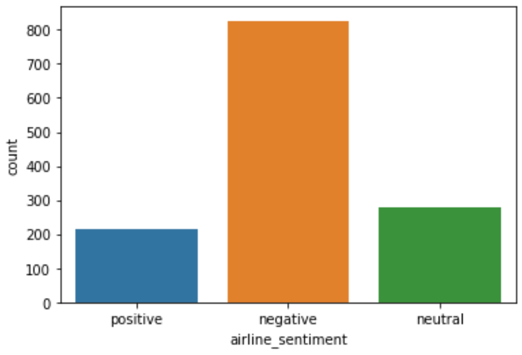


Figure 9 Validation dataset

To get the most frequent words used in each sentiment in the Training dataset, **word cloud** is used to get the most frequently used words in a sentiment. The following (*Figure 10.1, 10.2, 10.3*) refers the word cloud.

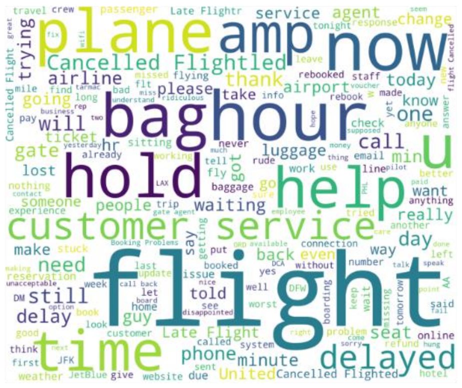


Figure 10.1 Negative

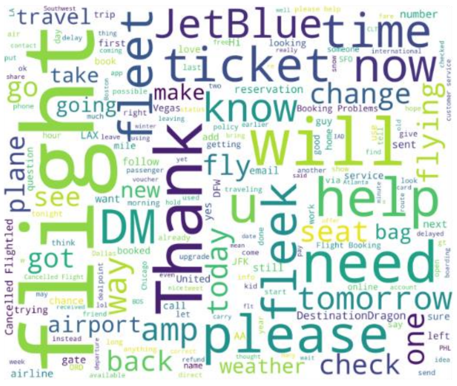


Figure 10.2 Neutral

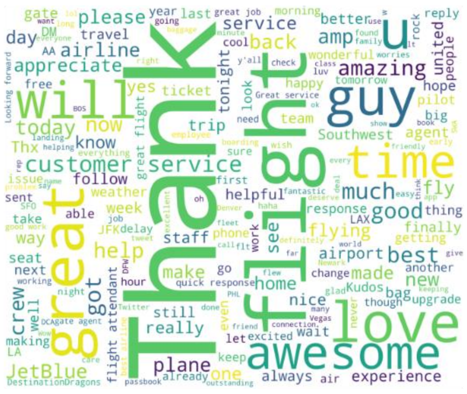


Figure 10.3 Positive

The target class that is "airline_Sentiment" class is converted to numerical values in both training and testing dataset. The following table refers the numerical values of the sentiment.

Sentiment	**Numerical value**
Negative	0
Neutral	1
Positive	2

Then, Synthetic Minority Oversampling Technique (SMOTE) is used to do the imbalance correction in all the three datasets. Now that we have pre-processed training, testing and validation data, the data can be loaded into the model to get the desired output.

MODEL TRAINING AND INTERPRETATION

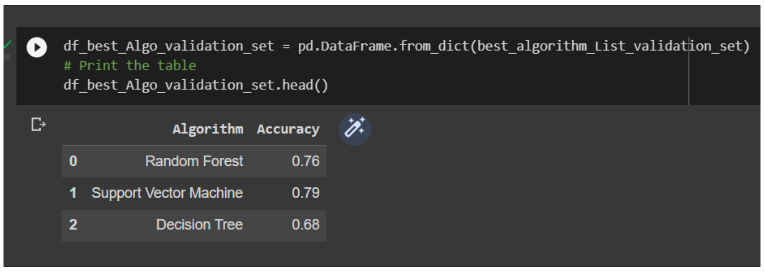
Since we have training, testing and validation data already, it is not necessary to split the data. Thus, we use the training dataset to train the model, validation dataset to validate the model and test dataset to test the model. Different classification models are used to get the best model out of it.

The following models are used in this problem.

- Random Forest Classifier
- Support Vector Machine Classifier
- Decision Tree Classifier

First, we inspect the outcomes of **Random Forest classifier**. The training data is fitted to train the model and the validation data is used to predict and calculate the F1 score for the model. Secondly, we inspect the outcomes of **Support Vector Machine (SVM) classifier** and finally the **Decision Tree classifier** in the same way as we did with Random Forest Classifier.

Finally, after the model interpretation, we could see the best classification model for the sentimental analysis. The below image (*Figure 11*) refers the best model for the sentimental analysis.



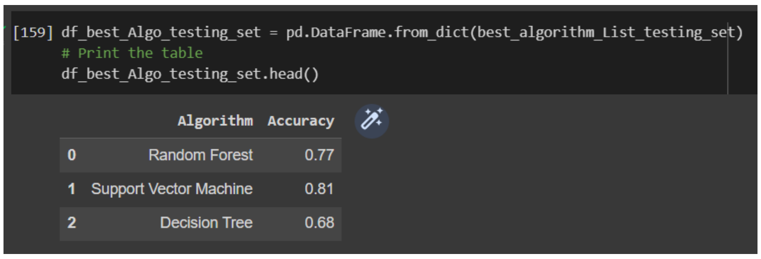
```
df_best_Algo_validation_set = pd.DataFrame.from_dict(best_algorithm_List_validation_set)
# Print the table
df_best_Algo_validation_set.head()
```

	Algorithm	Accuracy
0	Random Forest	0.76
1	Support Vector Machine	0.79
2	Decision Tree	0.68

Figure 11 Accuracy score with Validation data

From the above table, it is evident that the Support Vector Machine (SVM) produced the best accuracy than the other two models.

The same step is followed with the testing data to get the score of the model with the testing data. The below image (*Figure 12*) refers the accuracy score of the model with the testing data.



```
[159] df_best_Algo_testing_set = pd.DataFrame.from_dict(best_algorithm_List_testing_set)
      # Print the table
      df_best_Algo_testing_set.head()
```

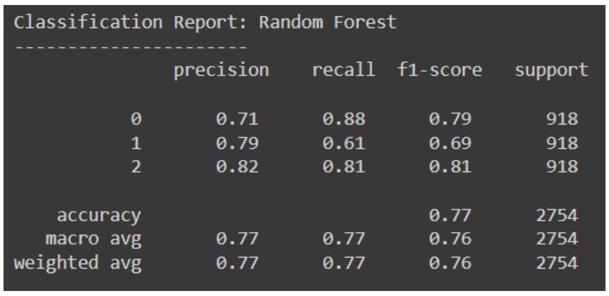
	Algorithm	Accuracy
0	Random Forest	0.77
1	Support Vector Machine	0.81
2	Decision Tree	0.68

Figure 12 Accuracy score with Testing data

EVALUATION

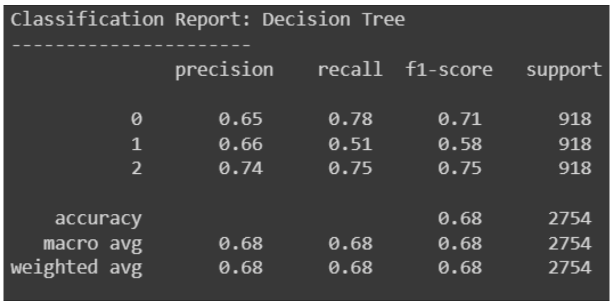
The model is evaluated using the Classification report. The classification report states the precision, recall, F1 score and support of the model.

Classification report of Random Forest classifier with Testing data:



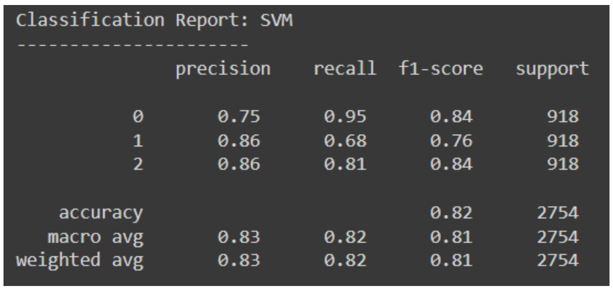
```
Classification Report: Random Forest
---------------------
              precision    recall  f1-score   support

           0       0.71      0.88      0.79       918
           1       0.79      0.61      0.69       918
           2       0.82      0.81      0.81       918

    accuracy                           0.77      2754
   macro avg       0.77      0.77      0.76      2754
weighted avg       0.77      0.77      0.76      2754
```

Classification report of Decision Tree classifier with Testing data:



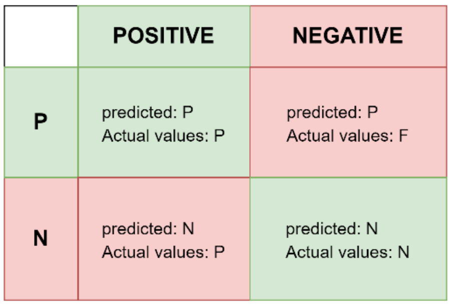
```
Classification Report: Decision Tree
---------------------
              precision    recall  f1-score   support

           0       0.65      0.78      0.71       918
           1       0.66      0.51      0.58       918
           2       0.74      0.75      0.75       918

    accuracy                           0.68      2754
   macro avg       0.68      0.68      0.68      2754
weighted avg       0.68      0.68      0.68      2754
```

Classification report of Support Vector Machine (SVM) classifier with Testing data:



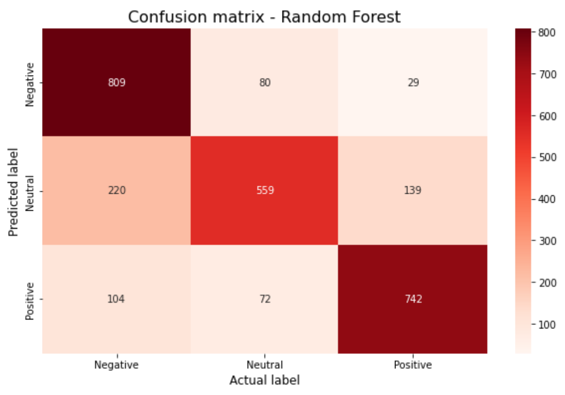
```
Classification Report: SVM
---------------------
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       918
           1       0.86      0.68      0.76       918
           2       0.86      0.81      0.84       918

    accuracy                           0.82      2754
   macro avg       0.83      0.82      0.81      2754
weighted avg       0.83      0.82      0.81      2754
```

Also, using the confusion matrix, we evaluate the outcomes of prediction. The confusion matrix is an *N x N matrix* used to evaluate the performance of a classification model. Where N is the number of target class. The below image explains the confusion matrix.

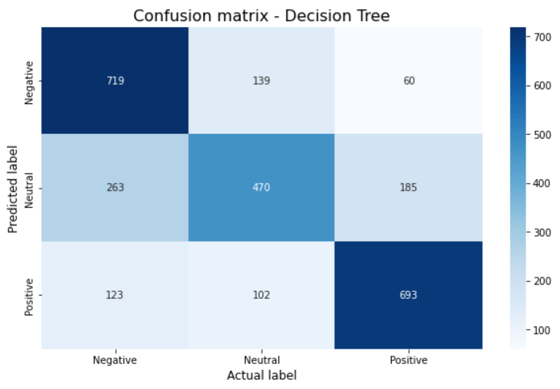


	POSITIVE	NEGATIVE
P	predicted: P Actual values: P	predicted: P Actual values: F
N	predicted: N Actual values: P	predicted: N Actual values: N

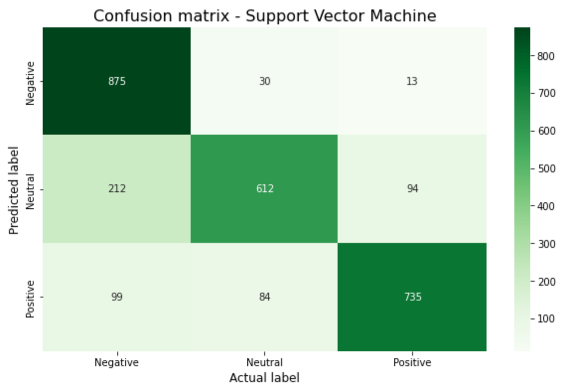
The confusion matrix of Random Forest model:



The confusion matrix of Decision Tree model:



The confusion matrix of the SVM model:



DISCUSSION

With the evaluated reports we knew that the Support Vector Machine Classifier had the best accuracy. From the classification report, the Precision refers to the accuracy of positive predictions. Recall refers to the fraction of positives that were correctly identified. F1 score refers to the mean of precision and recall. Support refers to the number of actual occurrences. Although, the accuracy of other two models are almost near to the Support Vector Machine Classifier, the precision score was better in SVM than the other two models. Looking at the confusion matrix, out of 2754 samples, 2247 samples was predicted successfully. Only about 507 samples where the prediction was gone wrong. It is evident that the accuracy of 82% was much better than the other models.

REFERENCES

1) https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397
2) https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
3) https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5