Airlines Customer Satisfaction

Analysis of Features that Contribute to the Positive and Negative Reviews of Airline Customer's Satisfaction

Bobbi McDermott
Higher Diploma in Science in Data Analytics
Co Dublin, Ireland
Sba22268@cct.ie

Damaris Nascimento de Souza Vargova
Higher Diploma in Science in Data Analytics
Co Dublin, Ireland
Sba22504@cct.ie

Leandro Marigo
Higher Diploma in Science in Data Analytics
Co Dublin, Ireland
Sba22145@cct.ie

Melissa Miskell
Higher Diploma in Science in Data Analytics
Co Dublin, Ireland
Sba22535@cct.ie

Dr. Muhammad Iqbal
Higher Diploma in Science in Data Analytics
Co Dublin, Ireland
miqbal@cct.ie

Table of Contents

I. INTRODUCTION

The Dataset is taken from Kaggle on a survey of Customer Satisfaction for an unknown airline. There were different features which were ranked from 1 to 5 based on overall satisfaction. The target variable is whether a customer was either satisfied or dissatisfied.

Words (43)

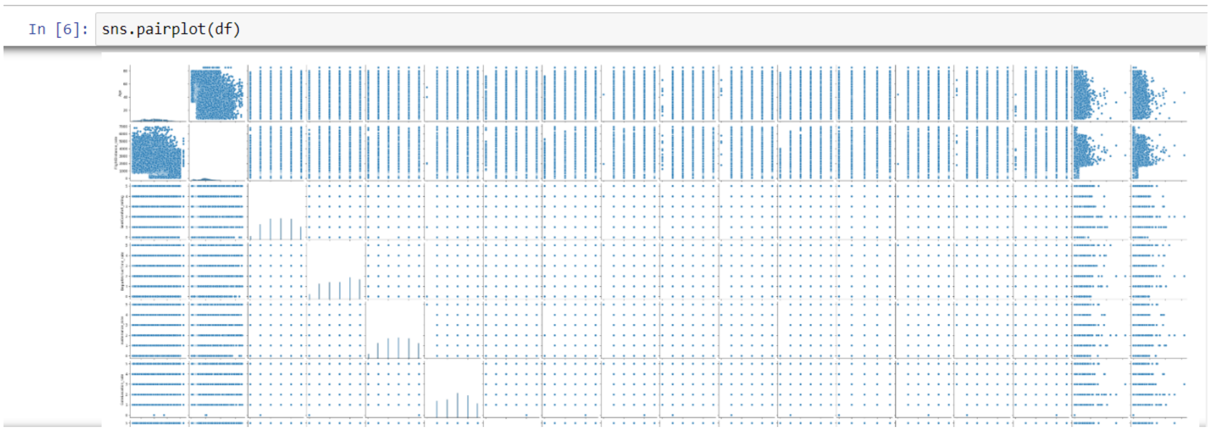
II. MOTIVATION

This dataset was chosen as it had a clear problem / solution objective. It could be run through analysis and machine learning algorithms to determine a) aspects that were important to the overall quality of a customer's experience and b) services that would require improvement for a better flying experience.

Words (50)

III. EXPLORATORY DATA ANALYSIS AND DESCRIPTIVE STATISTICS

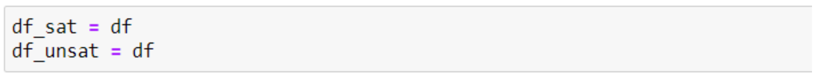
The first objective is understand the dataset as a combination of both numerical and categorical.. A pairplot was used to get a visualisation of the distribution of the set of all features where it was evident some correlations existed. (Figure 1)



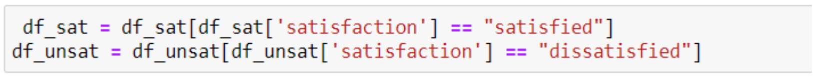
```
In [6]: sns.pairplot(df)
```

Figure 1

To analyse the statistical significance the describe function was used. Given the dataset holds information for two classes of customers (satisfied/dissatisfied) the describe function was used individually on each. (Figure 2).

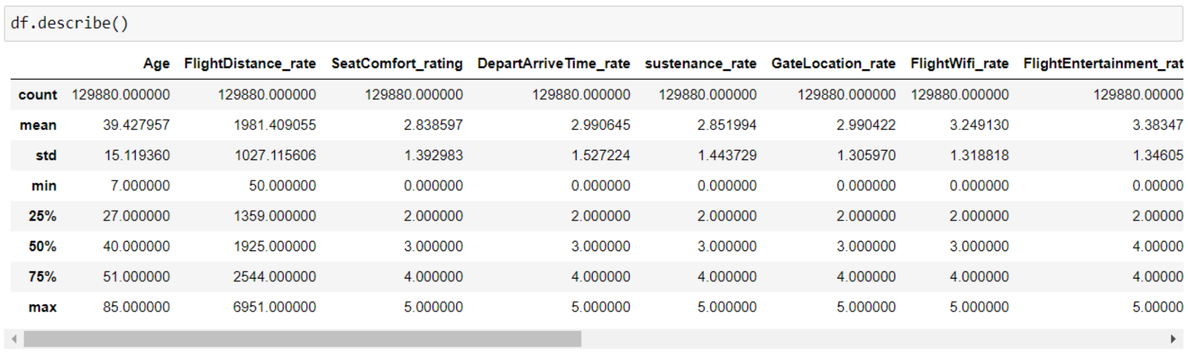


```
df_sat = df
df_unsat = df
```



```
 df_sat = df_sat[df_sat['satisfaction'] == "satisfied"]
df_unsat = df_unsat[df_unsat['satisfaction'] == "dissatisfied"]
```

The median score of all ratings is between 3 and 4 but when viewed individually these values changed. As hypothesised, the rating median was lower for unsatisfied versus higher for satisfied (Figure 2)



```
df.describe()
```

	Age	FlightDistance_rate	SeatComfort_rating	DepartArriveTime_rate	sustenance_rate	GateLocation_rate	FlightWifi_rate	FlightEntertainment_rat
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.00000
mean	39.427957	1981.409055	2.838597	2.990645	2.851994	2.990422	3.249130	3.38347
std	15.119360	1027.115606	1.392983	1.527224	1.443729	1.305970	1.318818	1.34605
min	7.000000	50.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	27.000000	1359.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000
50%	40.000000	1925.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.00000
75%	51.000000	2544.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.00000
max	85.000000	6951.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.00000

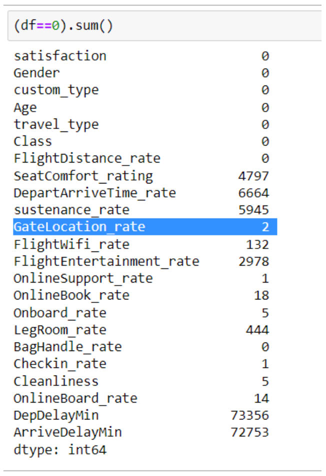
Figure 2

Words (109)

IV. Handling Missing Values

There are no duplicate observations and there are 393 missing values in the feature named "ArriveDelayMin" . These will be dropped as it represents only 0.3% of all data as considering dropping missing values, a good rule is to ensure data loss does not exceed 5% (Montelpare et al., 2020).

Separately, the sum of all 0's were checked across the df during EDA. Where 0 occurs in rating columns 0 represents not answered (Figure 3)

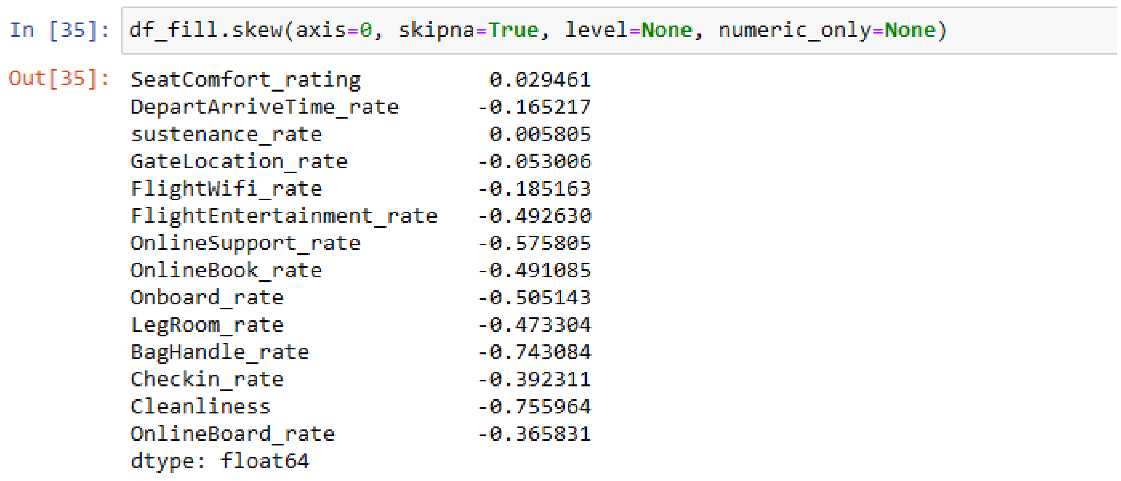


```
(df==0).sum()
```

```
satisfaction                 0
Gender                       0
custom_type                  0
Age                          0
travel_type                  0
Class                        0
FlightDistance_rate          0
SeatComfort_rating        4797
DepartArriveTime_rate     6664
sustenance_rate           5945
GateLocation_rate            2
FlightWifi_rate            132
FlightEntertainment_rate  2978
OnlineSupport_rate           1
OnlineBook_rate             18
Onboard_rate                 5
LegRoom_rate               444
BagHandle_rate               0
Checkin_rate                 1
Cleanliness                  5
OnlineBoard_rate            14
DepDelayMin              73356
ArriveDelayMin           72753
dtype: int64
```

Figure 3

Delay columns, 0 represents no delay. Therefore it was decided to fill unrated columns with a mean method. The skewness of each individual feature has been calculated in order to rationalise if mean can be used to fill missing values.(Figure 11)



```
In [35]: df_fill.skew(axis=0, skipna=True, level=None, numeric_only=None)

Out[35]: SeatComfort_rating           0.029461
         DepartArriveTime_rate       -0.165217
         sustenance_rate              0.005805
         GateLocation_rate           -0.053006
         FlightWifi_rate             -0.185163
         FlightEntertainment_rate    -0.492630
         OnlineSupport_rate          -0.575805
         OnlineBook_rate             -0.491085
         Onboard_rate                -0.505143
         LegRoom_rate                -0.473304
         BagHandle_rate              -0.743084
         Checkin_rate                -0.392311
         Cleanliness                 -0.755964
         OnlineBoard_rate            -0.365831
         dtype: float64
```

Figure 11

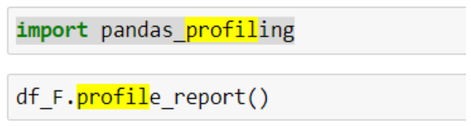
2010 Values between -2 and +2 are considered acceptable levels of distribution (George and Mallery, 2010). This is agreed with by Hair et al. (2010) and Bryne, 2010 who deem data to be within normal range, where skewness does not exceed in any direction -2 or +2. Considering the variation of the skew output, we could use the mean value to fill in 0 values where 0 represents NaN. The above code uses the columns while skipping null values to calculate a skew which is why we have converted 0 values to NaN prior to the calculation.

However, it must also be noted, as this is ordinal data, replacing NaN values with the mean is considered contentious as it renders the information as meaningless. This is a basis by basis argument and for the purpose of our analysis we will proceed to fill NaN values with the mean as our prediction model focuses on classifying unsatisfied customers as opposed to predicting individual ratings.

Words (272)

V. PANDAS'S PROFILING

Panda's profiling was used in order to gain insight into the distribution, descriptive statistics and ensure that the understanding of the data through EDA was correct (Figure 12)



```
import pandas_profiling
```

```
df_F.profile_report()
```

Figure 12

Words (27)

VI. FEATURE SELECTION

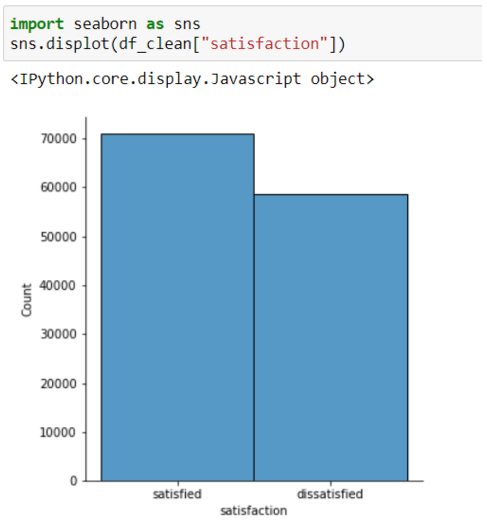
Feature selection as opposed to feature extraction has been determined to be the most optimal choice for reducing noise and maintaining important data for ML classification.

When considering a methodology, to cross examine the correlations determined in the heat map, feature selection using decision trees was most advantageous. F.S utilising decision trees allows for better readability and understanding especially when placed into a comparison with feature extraction.

When weighing up the pros and limitations, it was found that F.E has its limitations where it relates to dimensionality reduction. The focus of combining original features to create new features creates limitations on the interpretability of the data.

Another benefit to using decision trees presented itself in ways where data preparation for model training is minimal. The data being passed through is not required to be converted into a particular data type or scaled to suit any model requirements, thus its implementation can be quite simplistic.

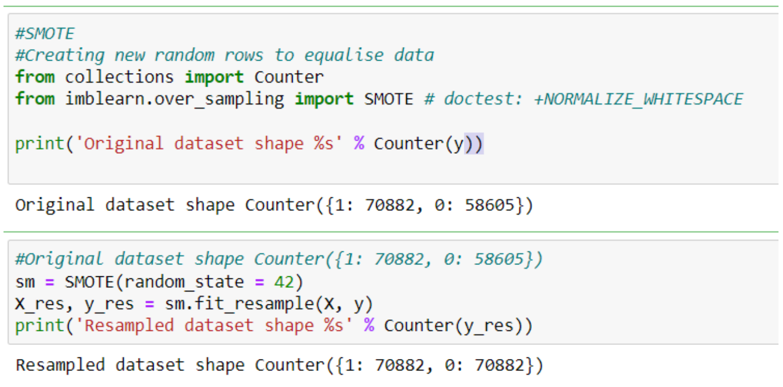
However, through visualisation imbalances were noted in the label category. To give the classification model an equal 50/50 chance of classifying for both classes, SMOTE was implemented as a method to balance this feature. (Figure 13), (Figure 14)



```
import seaborn as sns
sns.displot(df_clean["satisfaction"])
```

```
<IPython.core.display.Javascript object>
```

Figure 13



```
#SMOTE
#Creating new random rows to equalise data
from collections import Counter
from imblearn.over_sampling import SMOTE # doctest: +NORMALIZE_WHITESPACE

print('Original dataset shape %s' % Counter(y))
```

```
Original dataset shape Counter({1: 70882, 0: 58605})
```

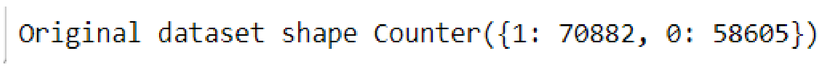
```
#Original dataset shape Counter({1: 70882, 0: 58605})
sm = SMOTE(random_state = 42)
X_res, y_res = sm.fit_resample(X, y)
print('Resampled dataset shape %s' % Counter(y_res))
```

```
Resampled dataset shape Counter({1: 70882, 0: 70882})
```

Figure 14

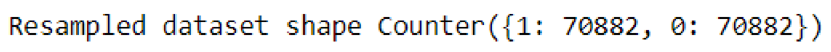
From a business perspective, the imbalance represents in favour of a higher count of satisfied customers as opposed to unsatisfied which is a favourable, although not an optimal, outcome for results.

In terms of selecting a balancing method, “a comparative review of SMOTE and ADASYN in imbalanced Data Classification, 2020”, was used as a reference guide. Understanding SMOTE’s utilisation of adding minority classes as opposed to removing data, duplicating data, or adding random rows was deemed to be the most appropriate. A closer look of the overall dataset appeared to confirm this where general features had what could be recognised, as unique responses, given the minority sampling results. In its comparative focus, SMOTE outperformed in multiple comparative reviews against ADASYN, which intended to address the problems of SMOTE. Below is the visual representation of data shape prior and after its implementation (Figure 15), (Figure 16)



```
Original dataset shape Counter({1: 70882, 0: 58605})
```

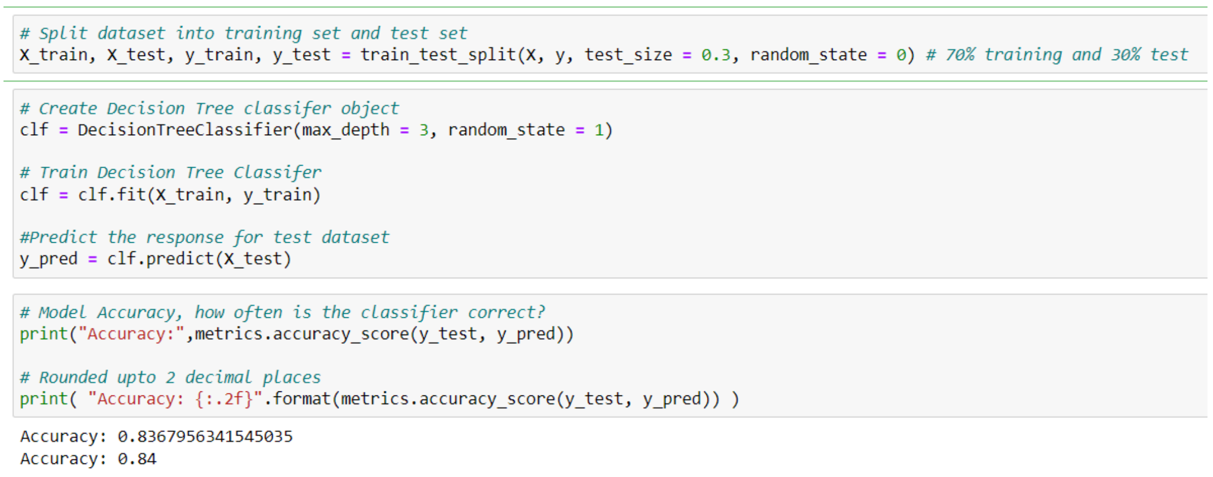
Figure 15



```
Resampled dataset shape Counter({1: 70882, 0: 70882})
```

Figure 16

When implementing the decision tree, a max depth of 3 was defined. All data was passed through to get a general mapping of features, represented with a low Gini index, where a Gini index of 0 represents perfect equality, while an index of 100 implies perfect inequality (Figure 17)



```python
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0) # 70% training and 30% test
```

```python
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(max_depth = 3, random_state = 1)

# Train Decision Tree Classifer
clf = clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```python
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Rounded upto 2 decimal places
print( "Accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)) )
```

```
Accuracy: 0.8367956341545035
Accuracy: 0.84
```

Figure 17

This method was re-run three times with the highest resulting accuracy at 70% training 30% testing ratio.

Results indicate where passengers satisfied or unsatisfied, a satisfied customer could be identified with a good rating of inflight entertainment and seat comfort whilst an unsatisfied customer could be correlated with a low rating of seat comfort (Figure 18), (Figure 19)

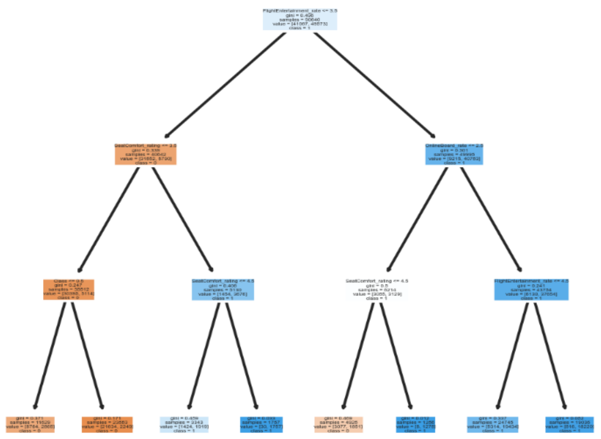


Figure 18

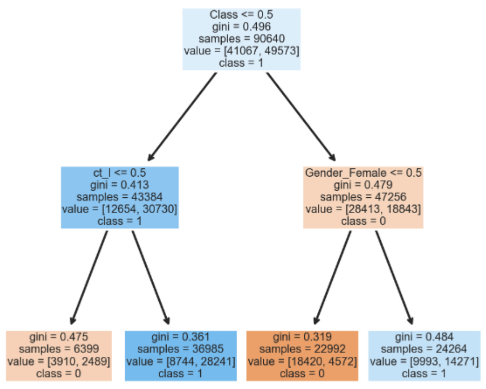


Figure 19

To get a better understanding of the type of person, only data relating to the classification of the individual was passed through the model. The Gini index demonstrated that females were most likely to be satisfied, with class of travel at eco leading to more unsatisfied customers and business class having higher levels of satisfaction. A random forest method was implemented as a cross validation technique which also demonstrated similar results.

Results from the algorithm could also be related to other approaches of correlation methods such as graphical representations and heatmaps (Figure 20)

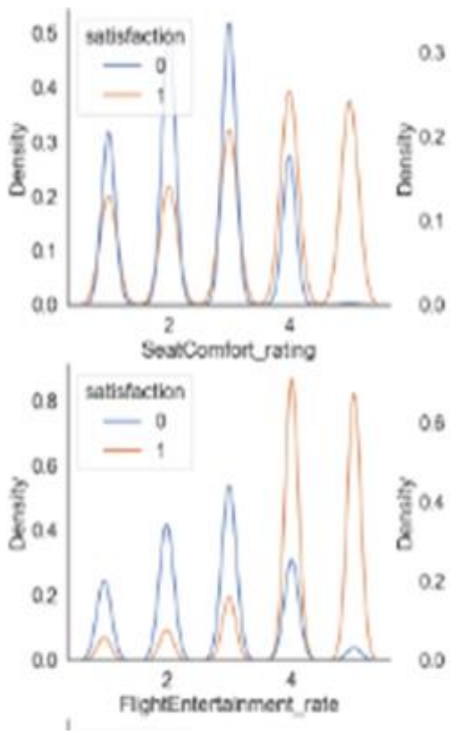


Figure 20

Random Forest was applied as a cross-check validation for the Decision Tree method and also to support the final decision around feature selection. The below image (Figure x) summarises the outcome of the Random Forest model which is similar to the Decision Tree results, putting Flight Entertainment, Seat Comfort and Online Board rating features as the most important from the rating features perspective and it shows that the most important features from the categorical (Customer Profile) perspective are Class, Age and Gender.

The ”Age” was passed through the ML models during the testing phase, however this feature was bringing accuracy down preventing the model to generalise predictions and so it was decided to remove this feature for the final models.

Random Forest is widely used for feature selection given it’s high accuracy, ability to generalize and for being easy to read and interpret. The model consists in multiple trees which will run through random extracted observations and features and “divide the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. Not every tree sees all the features or all the observations, and this guarantees that the trees are de-correlated and therefore less prone to overfitting.” (Dubey, A. (2018)) (Figure 21), (Figure 22)

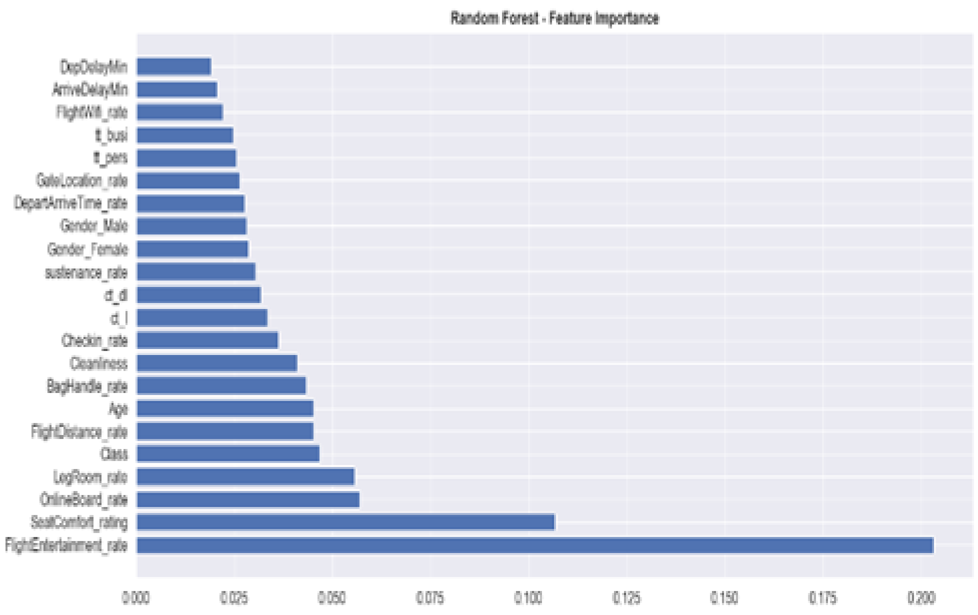


Figure 21

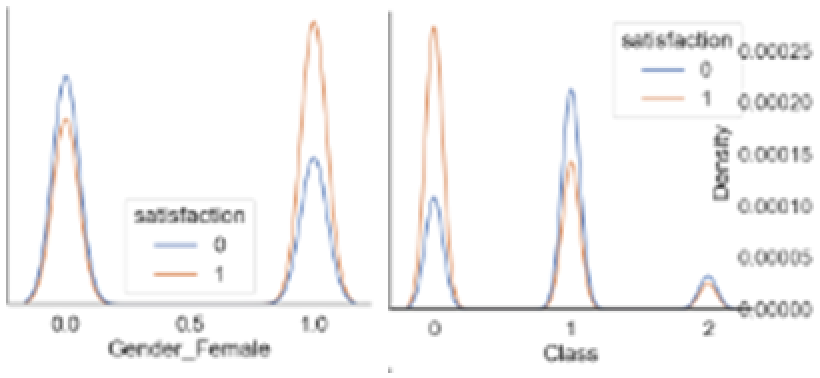
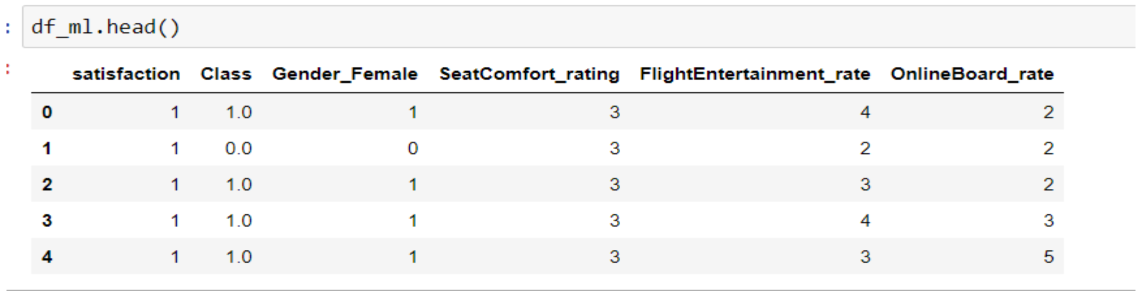


Figure 22

As demonstrated in the snapshot below, a new data frame was formed for classification prediction utilising lowest Gini calculations (Figure 23)



```
df_ml.head()
```

	satisfaction	Class	Gender_Female	SeatComfort_rating	FlightEntertainment_rate	OnlineBoard_rate
0	1	1.0	1	3	4	2
1	1	0.0	0	3	2	2
2	1	1.0	1	3	3	2
3	1	1.0	1	3	4	3
4	1	1.0	1	3	3	5

Figure 23

Words (631)

VII. Encoding

CATEGORICAL ENCODING

In order to create dummy variables from Pandas objects, the getdummies function was applied. The Pandas getdummies function, pd.get_dummies(), allows you to efficiently oneHot encode categorical data. A dummy variable is a numeric variable that encodes categorical information. Dummy variables have two possible values: 0 or 1.

In a dummy variable:

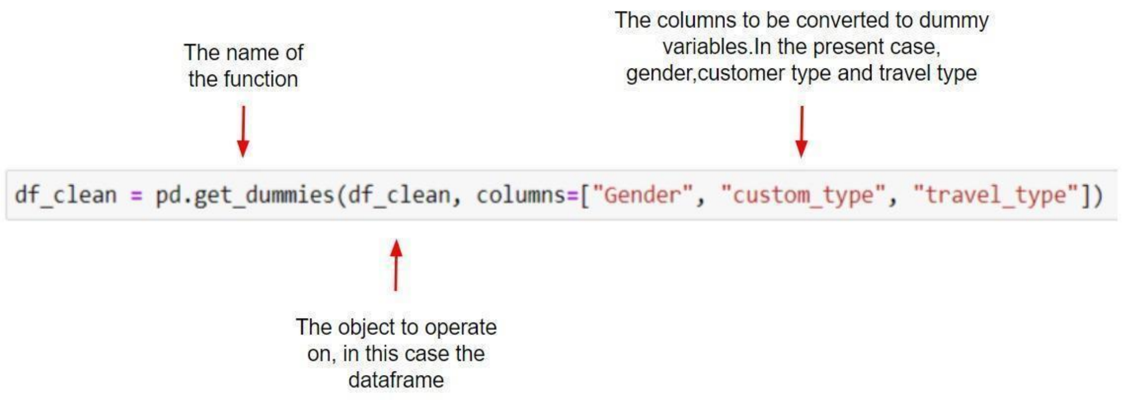
- A 1 encodes the presence of a category
- A 0 encodes the absence of a category

Importantly, pd.get_dummies can create dummy variables from a Pandas Series, or from a column or columns in a Pandas dataframe.

This applies in particular to machine learning. Many machine learning algorithms – like linear regression and logistic regression applied later in this model – strictly require numeric input data. Any attempt to use them with string-based categorical data will throw an error.

To use such tools, it is first necessary to encode categorical data as numeric dummy

The syntax of getdummies is simply explained in (Figure 24)



```
df_clean = pd.get_dummies(df_clean, columns=["Gender", "custom_type", "travel_type"])
```

The name of the function
The columns to be converted to dummy variables.In the present case, gender,customer type and travel type
The object to operate on, in this case the dataframe

Figure 24

ORDINAL ENCODING

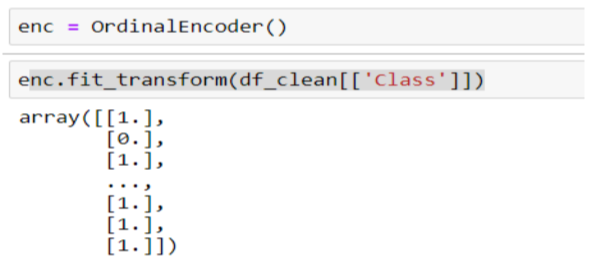
Once getdummies was applied, Ordinal Encoding was then used.

This type of encoding is used when the variables in the data are ordinal.

Ordinal encoding converts each label into integer values and the encoded data represents the sequence of labels in ordinal encoding, each unique category value is assigned an integer value.

The input to this transformer should be an array-like of integers or strings, denoting the values taken by categorical (discrete) features. The features are converted to ordinal integers.

The first step in the dataframe in question, an Ordinal Encoder was used to convert the feature class (Figure 25)



```
enc = OrdinalEncoder()
```

```
enc.fit_transform(df_clean[['Class']])
```

```
array([[1.],
       [0.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]])
```

Figure 25

As usual after each encoding was applied, the pandas function head() was also used to determine if it works properly.

Words (270)

VIII. Scaling

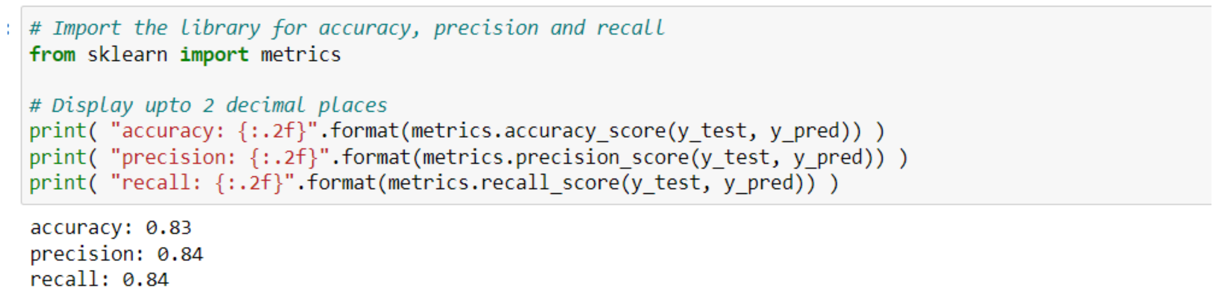
The MinMax scaler was applied before passing the dataset through the ML models and we saw accuracy improving by around 5% after scaling. MinMax scaler was used as we noted data is not normally distributed.

MinMax scaling will transform the data and scale it down to the range of 0 to 1 or -1 to 1 in case there are negative values in the dataset. "MinMax scaler preserves the shape of the original distribution, subtracting the minimum value in the feature and then dividing by the range. The range is the difference between the original maximum and original minimum." (Gogia, N. (2019). *Why Scaling is Important in Machine Learning?)*

Words (99)

IX. KNN

KNN is a valuable technique used when approaching a classification problem and it classifies the data point on how it's neighbour is classified (Kumar, 2020). It is needed to establish the view through KNN the accuracy, precision and recall (Figure 26)



```
# Import the library for accuracy, precision and recall
from sklearn import metrics

# Display upto 2 decimal places
print( "accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)) )
print( "precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)) )
print( "recall: {:.2f}".format(metrics.recall_score(y_test, y_pred)) )
```

```
accuracy: 0.83
precision: 0.84
recall: 0.84
```

Figure 26

In order to get the most accurate outcome from this method, it is important to look at hyperparameters to establish 7a k value. From the heatmap created, 5 was the optimal number to do this (Figure 27)

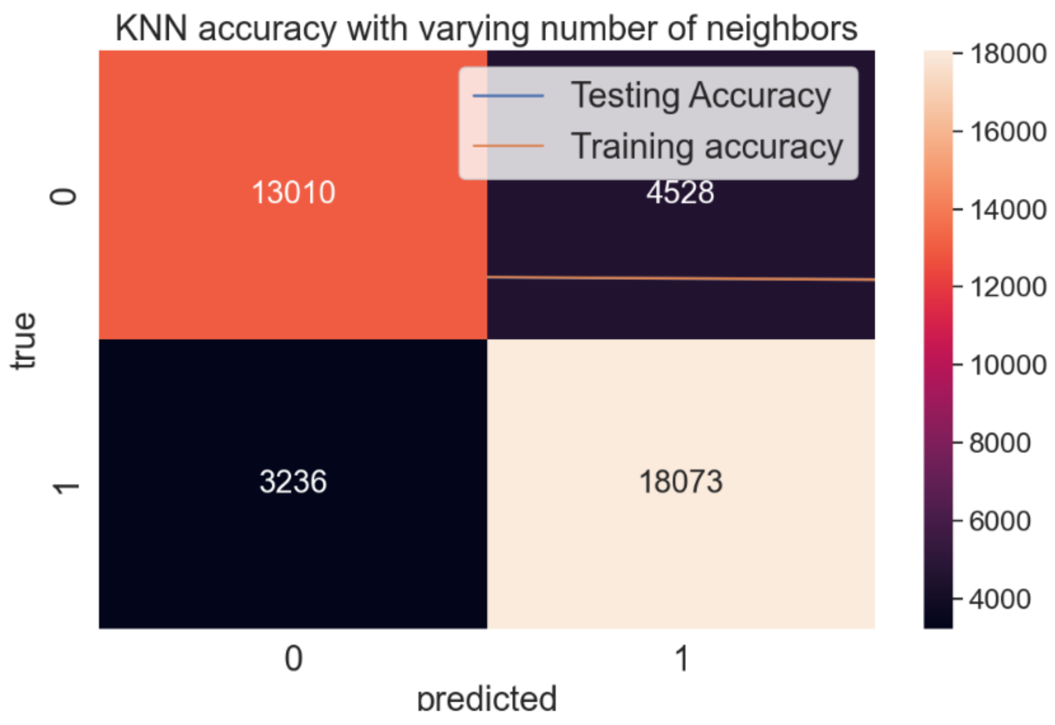
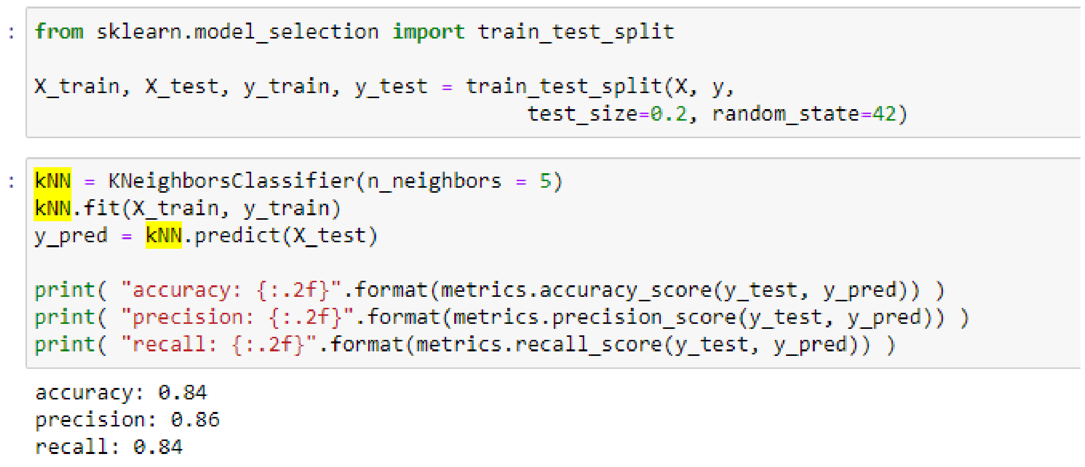


Figure 27

Various test sizes were used to get the highest accuracy and it 20% was the best outcome for this (Figure 28)



```
: from sklearn.model_selection import train_test_split

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

```
: kNN = KNeighborsClassifier(n_neighbors = 5)
  kNN.fit(X_train, y_train)
  y_pred = kNN.predict(X_test)

  print( "accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)) )
  print( "precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)) )
  print( "recall: {:.2f}".format(metrics.recall_score(y_test, y_pred)) )
```

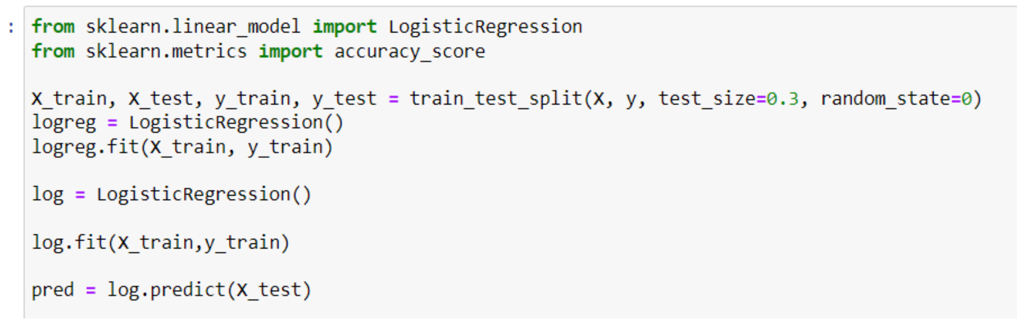
```
accuracy: 0.84
precision: 0.86
recall: 0.84
```

Figure 28

Words (91)

X. LOGISTIC REGRESSION

Logistic regression was chosen as one of the models to handle classifying satisfied and unsatisfied customers. It was chosen for this dataset as it works on calculating probabilities. As this model works best with correlated features, this model has been given features selected by importance, by both random forest and decision tree, to predict the probability of classifying satisfied and unsatisfied customers, set out under certain conditions. Logistic regression is also most optimal when working with ordinal data and binary data. As it is not one of the most complex and thus accurate models, it has been used in conjunction with KNN (Figure 29)



```
: from sklearn.linear_model import LogisticRegression
  from sklearn.metrics import accuracy_score

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
  logreg = LogisticRegression()
  logreg.fit(X_train, y_train)

  log = LogisticRegression()

  log.fit(X_train,y_train)

  pred = log.predict(X_test)
```

Figure 29

To visualize this a confusion matrix is effective at giving the performance of this algorithm (Figure 30)

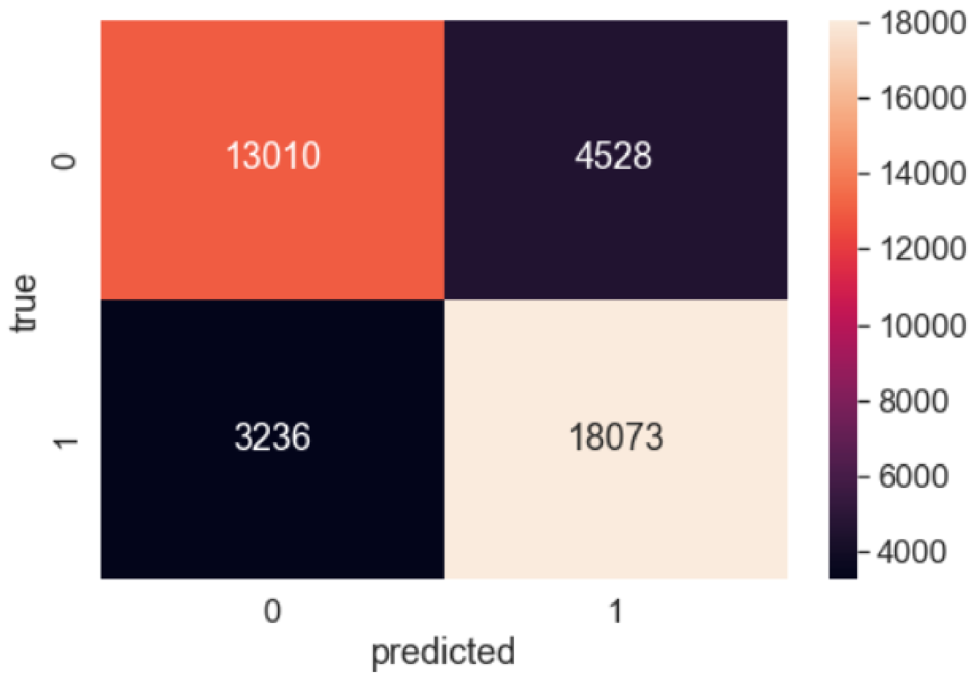
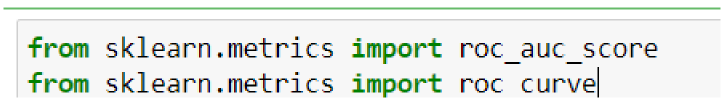


Figure 30

Even though the accuracy score is less than that than the prior algorithms achieved, an estimate can show how effective it is by applying it to some of the training data and comparing the prediction to the known value(Raj, 2020) The confusion matrix can be further used for determining various important metrics including Accuracy, ROC Score, Precision, F Score(Raj, 2020)

An ROC plot is used to tell how good the model is for distinguishing between the given classes(AskPython," 2021) To plot this additional libraries are employed (Figure x) and the curve demonstrated by plot (Figure 31),(Figure 32)



```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

Figure 31

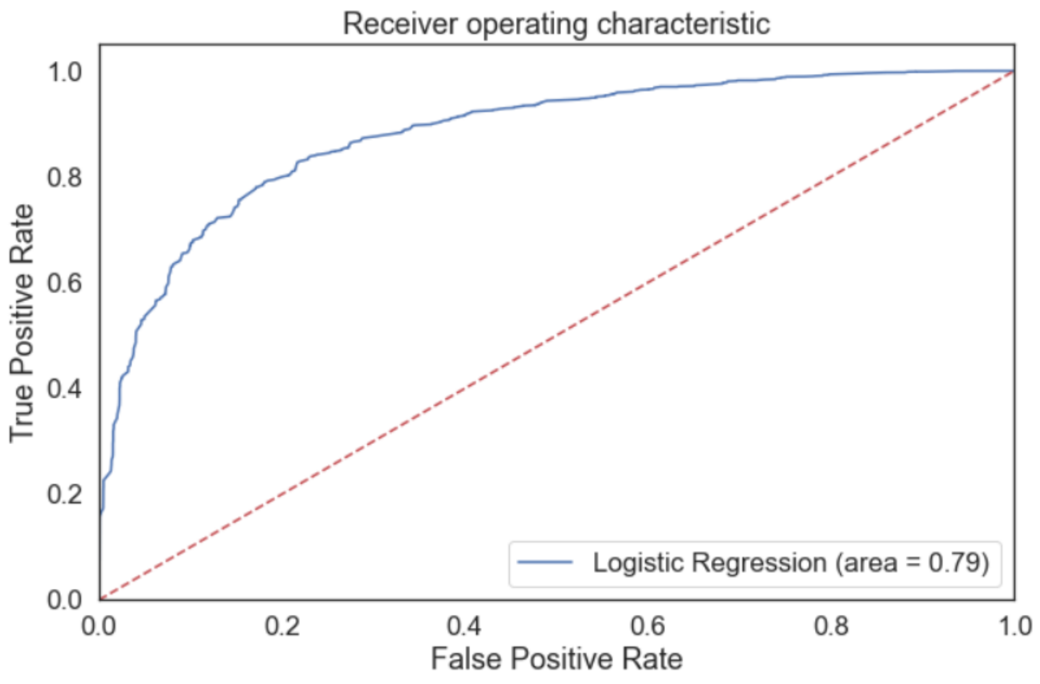


Figure 32

Words (208)

XI. CONCLUSION

The goal is to predict whether the airline customers were satisfied or unsatisfied based on the features selected for the model.

Machine Learning models were applied, two of them to select the most important features which would be used for the other two classification models.

Feature Selection models: Decision Tree and Random Forest
Classification models: kNN and Logistic Regression

During the Feature Selection phase both models predicted similar results and therefore our approach on features to be used for the classification models was a mix of the two models results. Decision Tree was applied twice in two different datasets, first over all features which brought the most important features related to the customer rating, 3 features were selected from the results ("Flight Entertainment", "Seat Comfort" and "Online Board"), and a second round of Decision Tree classification was applied over Customer Type features only to select important features which would give information about the customers profile. From the second round the selected features were "Class" and "Gender_Female" (which was also representative of the Male population as the feature was OneHot encoded).

Random Forest model was used to cross-check the Decision Tree results and also to look at the most important features ranking at a more granular level. After running the RF we also tried to pass "Age" as one of the features to be used on the classification models however we saw accuracy between Train and Test decrease and decided to leave "Age" out of the model.

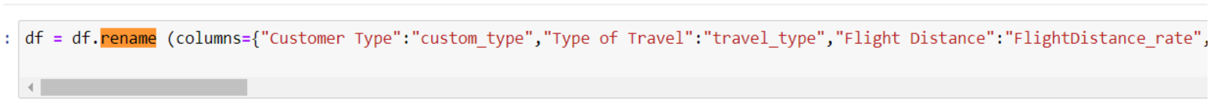
The first classification prediction model used, to predict whether the customer was satisfied or unsatisfied based on the features selected, was kNN (k Nearest Neighbours), 3 different test samples were used (30% / 20% / 10%) and a cross-check loop to find the optimal number of k was also applied. Best results were achieve using 20% for testing with 5k Neighbours for classification where accuracy reached 84%, precision 86% and recall 84% and the model was able to generalise predictions on a reasonable accuracy level based on the features selected.

The second classification model applied was Logistic Regression which is based on probability prediction. It was observed that the model was not able to achieve accuracy greater than 80% and based on precision, recall and f1-score, was not as strong in generalising as kNN for this purpose and therefore the model chosen to be used to classify the customer satisfaction as Satisfied or Unsatisfied based on the evaluation of the criteria: Flight Entertainment, Seat Comfort and Online Board, in conjunction with customer profile: Gender and Flight Class was kNN.

Words (425)

XII. Individual Contribution: Melissa Miskell

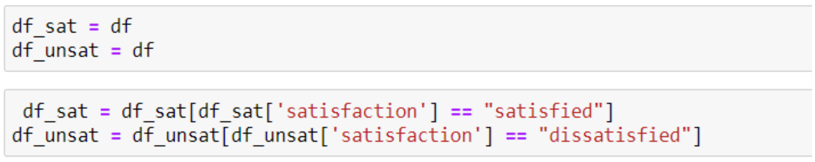
I began by running an early descriptive analysis on the dataset and renaming columns for ease of sharing and maintaining good standard practice (Figure 33)



```
: df = df.rename (columns={"Customer Type":"custom_type","Type of Travel":"travel_type","Flight Distance":"FlightDistance_rate",
```

Figure 33

I separated the df to run descriptive statistics on the columns as a dataset as a whole and for the satisfied and unsatisfied customers. The results demonstrated that ratings and age were different for each category (Figure 34)

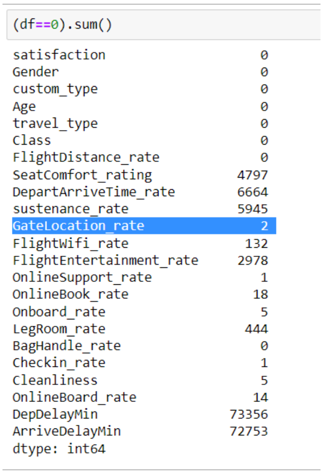


```
df_sat = df
df_unsat = df
```

```
 df_sat = df_sat[df_sat['satisfaction'] == "satisfied"]
df_unsat = df_unsat[df_unsat['satisfaction'] == "dissatisfied"]
```

Figure 34

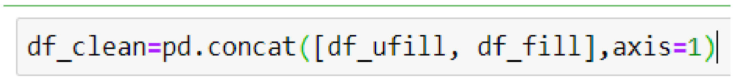
I checked the sum of 0 values to determine how much of the data in columns was left unanswered and the sparsity of the delay columns (Figure 35)



```
(df==0).sum()

satisfaction                  0
Gender                        0
custom_type                   0
Age                           0
travel_type                   0
Class                         0
FlightDistance_rate           0
SeatComfort_rating         4797
DepartArriveTime_rate      6664
sustenance_rate            5945
GateLocation_rate             2
FlightWifi_rate             132
FlightEntertainment_rate   2978
OnlineSupport_rate            1
OnlineBook_rate              18
Onboard_rate                  5
LegRoom_rate                444
BagHandle_rate                0
Checkin_rate                  1
Cleanliness                   5
OnlineBoard_rate             14
DepDelayMin               73356
ArriveDelayMin            72753
dtype: int64
```

Figure 35

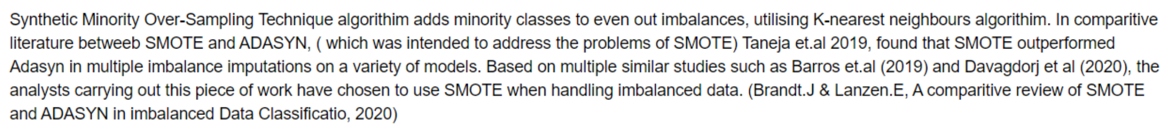
After I, and the team, ran a quick calculation cross checked with relevant literature, we determined that we could drop the 393 null values from the df. When handling unanswered rating data, I read a substantial amount of literature to aid in a group decision. I split the dataset for ease of filling these values and concatenated the df when clean (Figure 36)



```
df_clean=pd.concat([df_ufill, df_fill],axis=1)
```

Figure 36

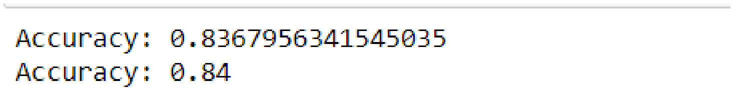
I used heatmap visualisations to determine early correlations both together and as a divided df between labels and as well as for visualising imbalances. All team members ran appropriate encoding of categorical data. I then used decision trees to aid in feature selection which utilises Gini calculations to determine the most important features. I read through research to help with this decision vs feature extraction. To ensure the model returned with 50/50 chance of either satisfied or unsatisfied, I used SMOTE to balance the label (Figure 37)



Synthetic Minority Over-Sampling Technique algorithim adds minority classes to even out imbalances, utilising K-nearest neighbours algorithim. In comparitive literature betweeb SMOTE and ADASYN, (which was intended to address the problems of SMOTE) Taneja et.al 2019, found that SMOTE outperformed Adasyn in multiple imbalance imputations on a variety of models. Based on multiple similar studies such as Barros et.al (2019) and Davagdorj et al (2020), the analysts carrying out this piece of work have chosen to use SMOTE when handling imbalanced data. (Brandt.J & Lanzen.E, A comparitive review of SMOTE and ADASYN in imbalanced Data Classificatio, 2020)

Figure 37

The split was settled at 70/30 for highest accuracy re-ran from two other splits of 60/40 and 80/20 (Figure 38)



```
Accuracy: 0.8367956341545035
Accuracy: 0.84
```

Figure 38

I chose the max depth at 3 for readability and the outcome was cross validated by random forest. The team ran Logistic Regression and Knn models to predict problem areas, solutions and characteristics of customers. In terms of report writing, I contributed to handling missing values, decision trees, random forest and rationale for logistic regression as well as bibliography.

Words (308)

XIII. Individual Contribution: Leandro Marigo

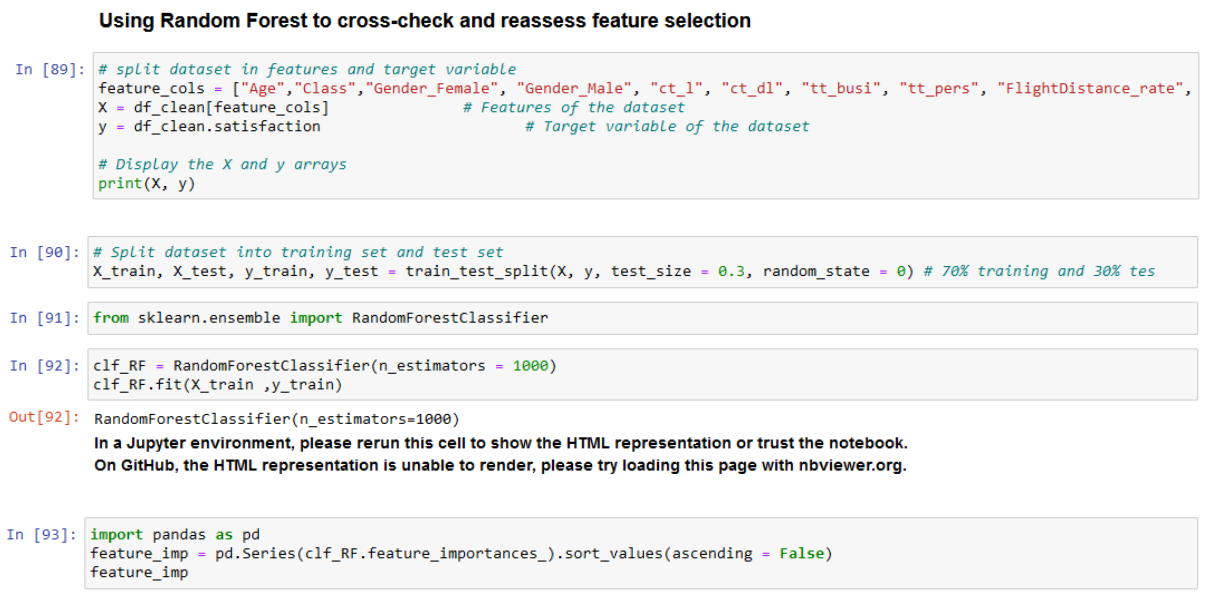
I contributed with the search and choice of the dataset in Kaggle considering it fits the requirements for the CA1 and would be an interesting dataset to work with. I have also contributed with codes for the initial EDA and understanding of the dataset discussing with the team better approaches for handling data cleaning. I have initially dropped the entire column containing missing values ("Arrival Delay in Minutes) but after discussing with the team we have decided to drop only the missing values which wouldn't represent much of the dataset (aprox. 0.3%). On the Encoding section I applied the get.dummies method for encoding some of the categorical variables, initially have applied the get.dummies encoding for the features ("Gender", "Customer Type", "Type of Travel" and "Class"), but after discussing with the team we have decided to apply the Ordinal Encoder for the "Class" feature as it represents a order of the flight classes and this would reduce the number of features / dimensions for analysis, and applied Label Encoder for the target variable "satisfaction".

After data cleaning and preparation, we worked through visualizations techniques and some of the Machine Learning models (Decision Tree and Random Forest) in order to decide which features would be used for the classification models (kNN and Logistic Regression).

When first applied, the kNN model was overfitted with a very high accuracy on the training set and lower accuracy on the testing, but improved after feature selection and MinMax scaling.

In the report dedicated time on writing about missing values, Random Forest for feature selection, Scaling and Conclusions.

Below are some of the codes I worked on the notebook (Figure 39)



Using Random Forest to cross-check and reassess feature selection

```
In [89]: # split dataset in features and target variable
         feature_cols = ["Age","Class","Gender_Female", "Gender_Male", "ct_l", "ct_dl", "tt_busi", "tt_pers", "FlightDistance_rate",
         X = df_clean[feature_cols]                # Features of the dataset
         y = df_clean.satisfaction                        # Target variable of the dataset

         # Display the X and y arrays
         print(X, y)
```

```
In [90]: # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0) # 70% training and 30% tes
```

```
In [91]: from sklearn.ensemble import RandomForestClassifier
```

```
In [92]: clf_RF = RandomForestClassifier(n_estimators = 1000)
         clf_RF.fit(X_train ,y_train)
```

```
Out[92]: RandomForestClassifier(n_estimators=1000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [93]: import pandas as pd
         feature_imp = pd.Series(clf_RF.feature_importances_).sort_values(ascending = False)
         feature_imp
```

Scaling using MinMax

```
In [103]: from sklearn.preprocessing import MinMaxScaler
```

```
In [104]: cols = ['Class', 'Gender_Female', 'SeatComfort_rating', 'FlightEntertainment_rate', 'OnlineBoard_rate']
          min_max = MinMaxScaler()

          X_train = min_max.fit_transform(X_train)
          X_test = min_max.fit_transform(X_test)
```

```
In [105]: X_train
```

```
Out[105]: array([[0.  , 1.  , 0.75, 1.  , 0.5 ],
                 [0.5 , 0.  , 0.5 , 0.5 , 0.  ],
                 [0.5 , 1.  , 0.5 , 0.25, 0.25],
                 ...,
                 [0.5 , 1.  , 0.5 , 0.5 , 0.  ],
                 [0.5 , 1.  , 0.75, 0.75, 0.  ],
                 [0.  , 0.  , 0.5 , 1.  , 1.  ]])
```

Applying kNN model for k = 3

```
In [106]: from sklearn.neighbors import KNeighborsClassifier
          from sklearn import metrics
          from sklearn.metrics import accuracy_score
```

```
In [107]: kNN = KNeighborsClassifier(n_neighbors = 3)
          kNN.fit(X_train, y_train)
          y_pred = kNN.predict(X_test)

          print( "Accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)) )
```

```
Accuracy: 0.83
```

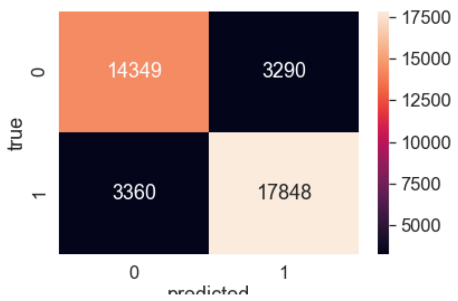
```
In [108]: # Import the library for accuracy, precision and recall
          from sklearn import metrics

          # Display upto 2 decimal places
          print( "accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)) )
          print( "precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)) )
          print( "recall: {:.2f}".format(metrics.recall_score(y_test, y_pred)) )
```

```
accuracy: 0.83
precision: 0.84
recall: 0.84
```

kNN - Confusion Matrix

```
In [109]: from sklearn.metrics import confusion_matrix
          import seaborn as sns

          y_predict = kNN.predict(X_test)
          cm = confusion_matrix(y_test, y_predict)

          sns.heatmap(cm, annot = True, fmt='g')
          plt.xlabel("predicted")
          plt.ylabel("true")
```

```
<IPython.core.display.Javascript object>
```



```
Out[109]: Text(25.5, 0.5, 'true')
```

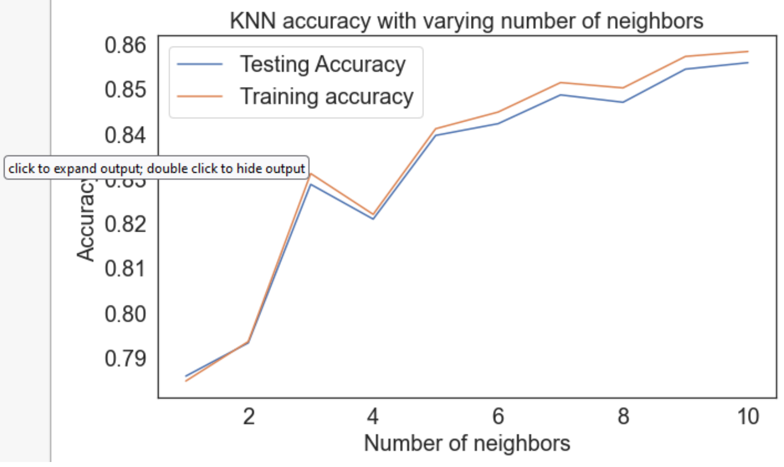
kNN Hyperparameters and k number selection

In [110]:
```python
from sklearn.metrics import classification_report

# Display the classification report
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81     17639
           1       0.84      0.84      0.84     21208

    accuracy                           0.83     38847
   macro avg       0.83      0.83      0.83     38847
weighted avg       0.83      0.83      0.83     38847
```

In [113]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize = (10, 6))
plt.title('KNN accuracy with varying number of neighbors', fontsize = 20)
plt.plot(neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training accuracy')
plt.legend(prop={'size': 20})
plt.xlabel('Number of neighbors', fontsize = 20)
plt.ylabel('Accuracy', fontsize = 20)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.show()
```

```
<IPython.core.display.Javascript object>
```



Words (272)

```
<IPython.core.display.Javascript object>
```

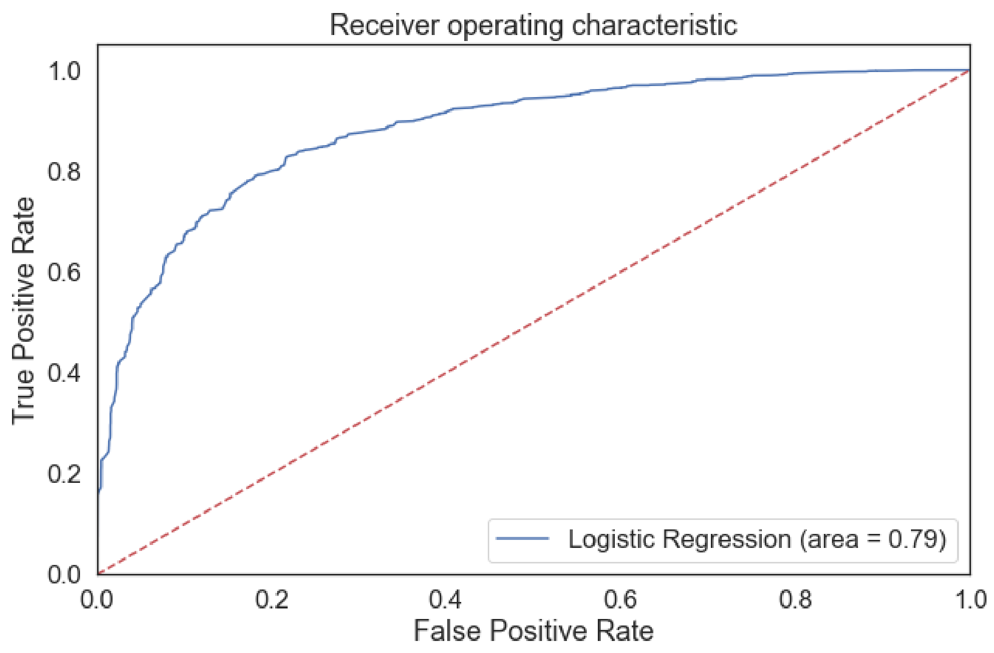
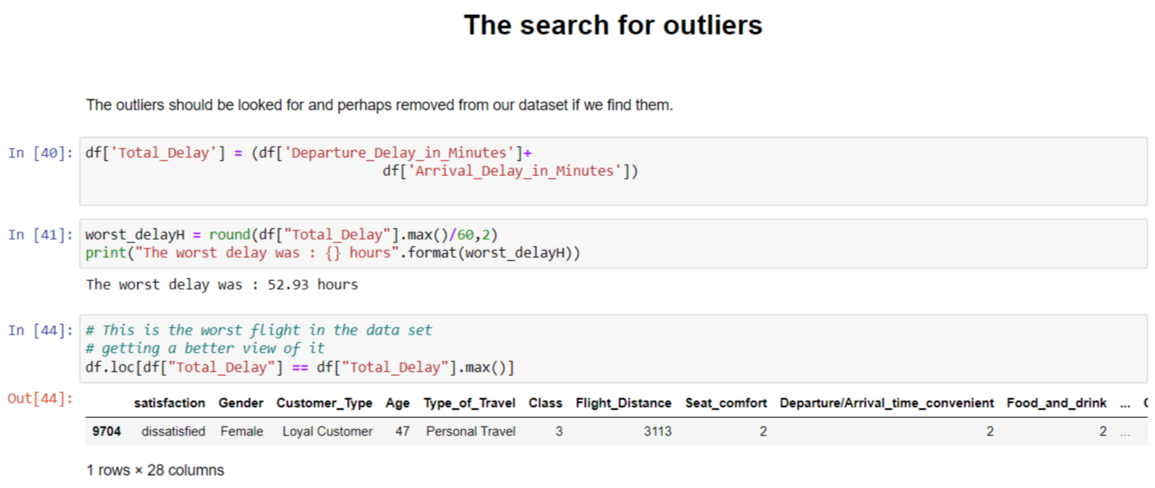


Figure 39

XIV. Individual Contribution: Damaris Nascimento de Souza Vargova

As agreed within our group, everyone created their own analysis in the Jupyter Notebook file of the data set chosen by the team. Then decide what information to keep and what to discard each contributor and then it was built the main report. Tasks were divided evenly, and each member was fully engaged, resulting in an equally effort.

I participated in the weekly meetings that the group held. Usually it was 2 meetings weekly. In my own analysis of the Airlines Customer Satisfaction data set, I started by importing the libraries, loading and checking the data. Following this, I performed basic analysis as usual, outlier detection, cleaning of the data. (Figure 40)



The search for outliers

The outliers should be looked for and perhaps removed from our dataset if we find them.

```
In [40]: df['Total_Delay'] = (df['Departure_Delay_in_Minutes']+
                              df['Arrival_Delay_in_Minutes'])
```

```
In [41]: worst_delayH = round(df["Total_Delay"].max()/60,2)
         print("The worst delay was : {} hours".format(worst_delayH))

         The worst delay was : 52.93 hours
```

```
In [44]: # This is the worst flight in the data set
         # getting a better view of it
         df.loc[df["Total_Delay"] == df["Total_Delay"].max()]
```

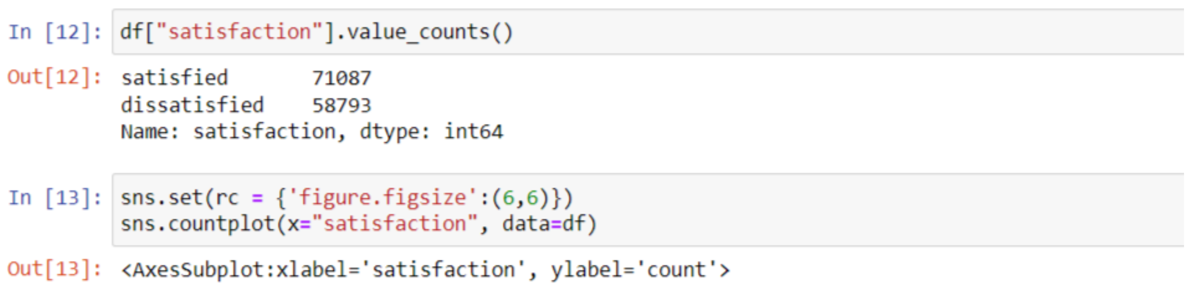
Out[44]:

	satisfaction	Gender	Customer_Type	Age	Type_of_Travel	Class	Flight_Distance	Seat_comfort	Departure/Arrival_time_convenient	Food_and_drink	...
9704	dissatisfied	Female	Loyal Customer	47	Personal Travel	3	3113	2	2	2	...

1 rows × 28 columns

Figure 40

Visualisation techniques were used (Figure 41)



```
In [12]: df["satisfaction"].value_counts()

Out[12]: satisfied       71087
         dissatisfied    58793
         Name: satisfaction, dtype: int64

In [13]: sns.set(rc = {'figure.figsize':(6,6)})
         sns.countplot(x="satisfaction", data=df)

Out[13]: <AxesSubplot:xlabel='satisfaction', ylabel='count'>
```

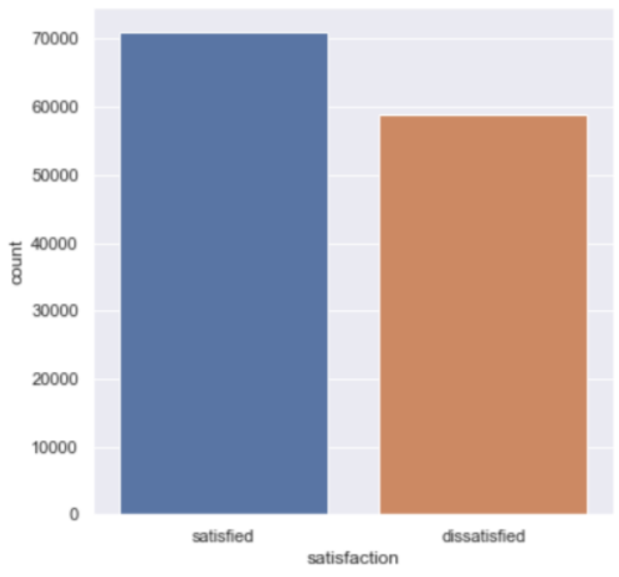
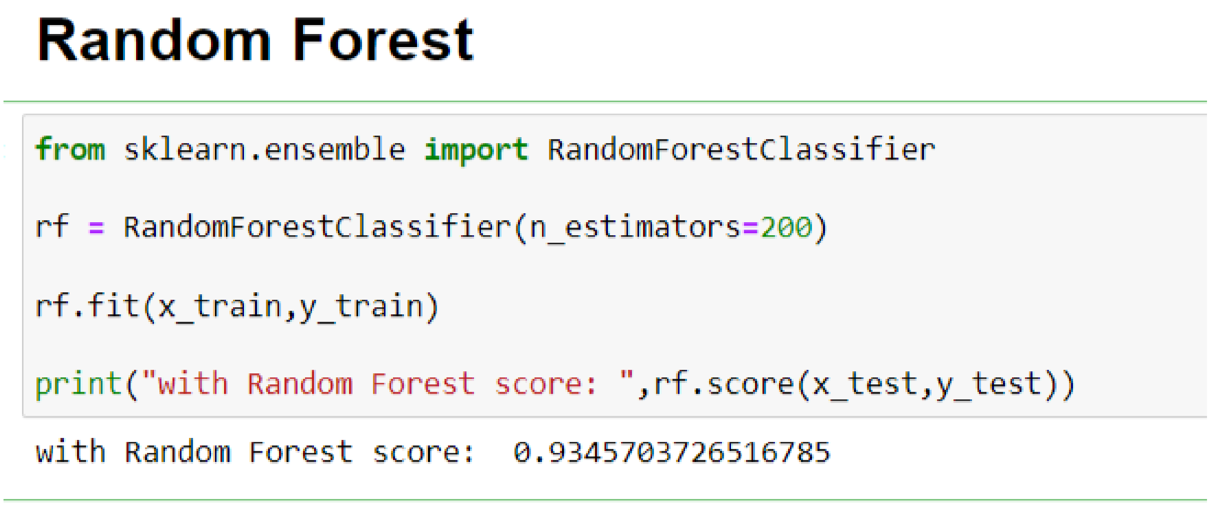


Figure 41

Missing values were also taken into account and handled appropriately.

As well, work on feature engineering, encoding, normalising, scaling, and preparing the data set to apply the models. The algorithms that gave me better results were Random Forest, and KNN (Figure 42), (Figure 43)



Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=200)

rf.fit(x_train,y_train)

print("with Random Forest score: ",rf.score(x_test,y_test))

with Random Forest score:  0.9345703726516785
```

Figure 42

KNN

```
In [39]: from sklearn.neighbors import KNeighborsClassifier

         knn = KNeighborsClassifier(n_neighbors=3)

         knn.fit(x_train,y_train)

         pred_for_knn = knn.predict(x_test)

         print("with KNeighborsClassifier score: ",knn.score(x_test, y_test))
```

```
with KNeighborsClassifier score:  0.9580304896827841
```

We get an accuracy score of 95%, but we can change the K value and look at the success results of the algorithm and find the optimum K value.

We get an accuracy score of 95%, but we can change the K value and look at the success results of the algorithm and find the optimum K value.

```
In [40]: score_list = []

         for x in range(1,20):
             knn2 = KNeighborsClassifier(n_neighbors=x)
             knn2.fit(x_train,y_train)
             score_list.append(knn2.score(x_test,y_test))
             print("if k values is",x,"value score",knn2.score(x_test,y_test))


         plt.plot(range(1,20),score_list)
         plt.xticks(range(1,20))
         plt.xlabel("k values")
         plt.ylabel("score")
         plt.legend()
         plt.grid()
         plt.show()
```

```
if k values is 1 value score 1.0
if k values is 2 value score 0.9524253156760086
if k values is 3 value score 0.9580304896827841
if k values is 4 value score 0.9436710809978441
if k values is 5 value score 0.9468971358176779
if k values is 6 value score 0.9387357560825377
if k values is 7 value score 0.94125346473668
if k values is 8 value score 0.9354712041884817
if k values is 9 value score 0.9368417000307977
if k values is 10 value score 0.9332768709578072
if k values is 11 value score 0.934678164459501
if k values is 12 value score 0.930890052356021
if k values is 13 value score 0.931952571604558
if k values is 14 value score 0.9290421927933477
if k values is 15 value score 0.930073914382507
if k values is 16 value score 0.9274638127502309
if k values is 17 value score 0.9288805050816138
if k values is 18 value score 0.926632275947028
```

```
if k values is 19 value score 0.9275562057283646
```

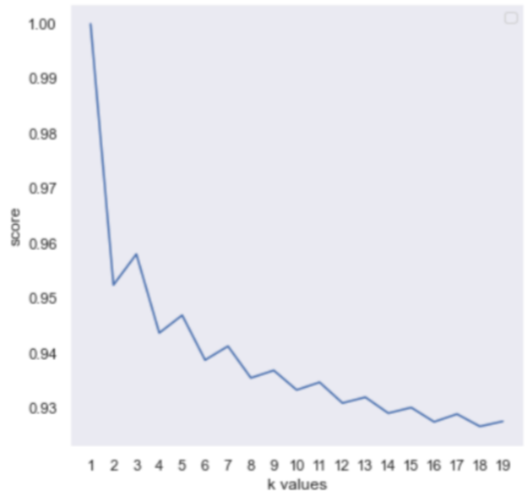


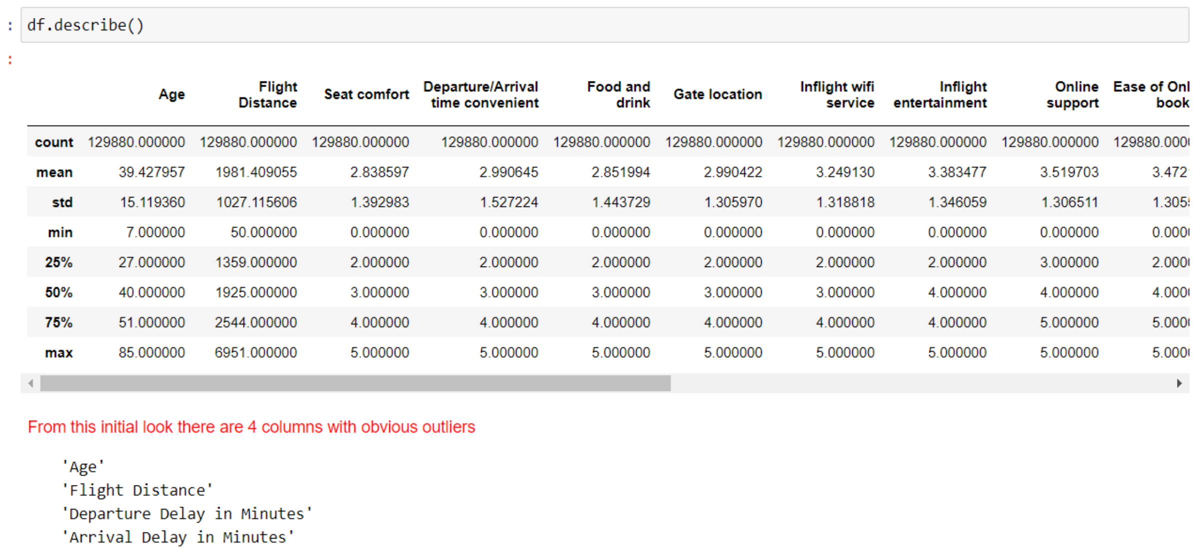
Figure 43

It was the task of each group member to write the reasoning behind some of the decisions taken. Mine was to put together comments on encoding categorical data, and work with the group in the comments on the conclusion of the task.

Words (196)

XV. Individual Contribution: Bobbi McDermott

The assignment was a great learning experience from my perspective. I reviewed the dataset and description and put forward method's on how to approach the problem by a) establishing the target variable and b) what machine learning algorithms would be best to handle a categorical problem. I suggested we look at Logistic Regression, Decision Tree, Random Forest, KNN and Gausian NB. We all developed our own notebooks and took the best aspects of each. I initially viewed the statistical distribution to establish any outliers (Figure 44)



```
df.describe()
```

	Age	Flight Distance	Seat comfort	Departure/Arrival time convenient	Food and drink	Gate location	Inflight wifi service	Inflight entertainment	Online support	Ease of Onl book
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000
mean	39.427957	1981.409055	2.838597	2.990645	2.851994	2.990422	3.249130	3.383477	3.519703	3.472
std	15.119360	1027.115606	1.392983	1.527224	1.443729	1.305970	1.318818	1.346059	1.306511	1.305
min	7.000000	50.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	27.000000	1359.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	3.000000	2.000
50%	40.000000	1925.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000	4.000
75%	51.000000	2544.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000	5.000
max	85.000000	6951.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000

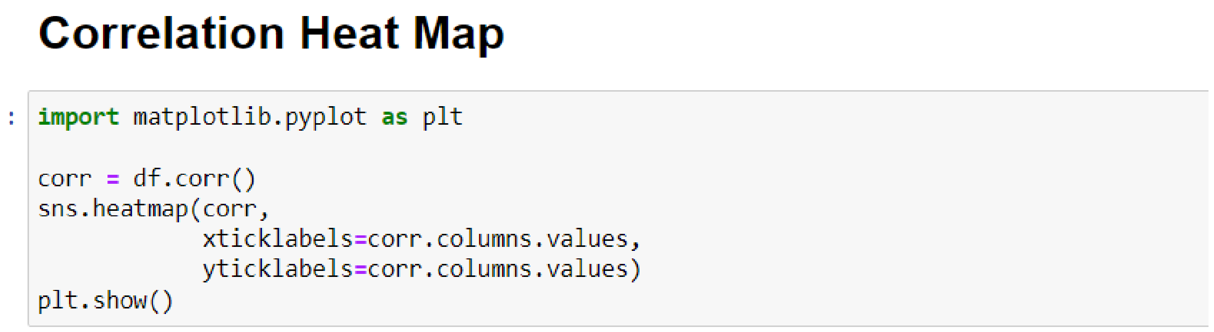
From this initial look there are 4 columns with obvious outliers

```
'Age'
'Flight Distance'
'Departure Delay in Minutes'
'Arrival Delay in Minutes'
```

Figure 44

I suggested to drop 'Arrival delay in Minutes' as it had null values and the Departure Delay in minutes would provide almost similar data. To make the notebook binary for analysis, I explored OneHot encoding but we decided on dummy encoding as it was the cleanest approach. Ordinal encoding was used for 'Class'. It was also important from my point of view to visualize the balance of the target variable in order to see if we need to manipulate it to give the training data a 50/50 weight to work with.

Further visualisation I employed was a heat map(Figure 45), (Figure 46)



Correlation Heat Map

```
import matplotlib.pyplot as plt

corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
plt.show()
```

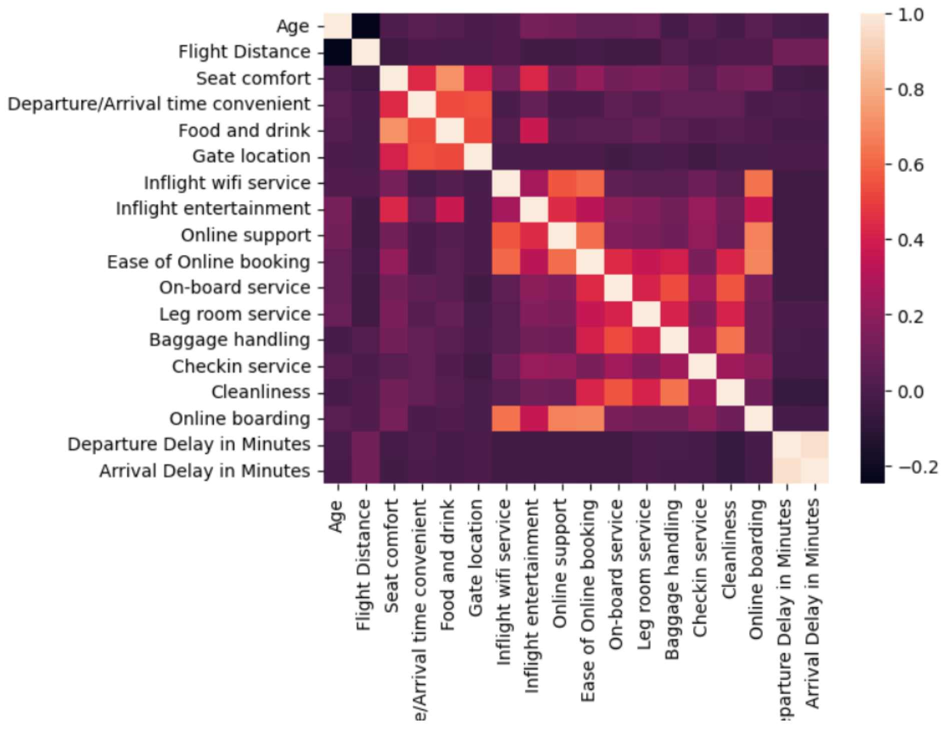


Figure 45

```
: df['satisfaction'].hist()

: <AxesSubplot:>
```

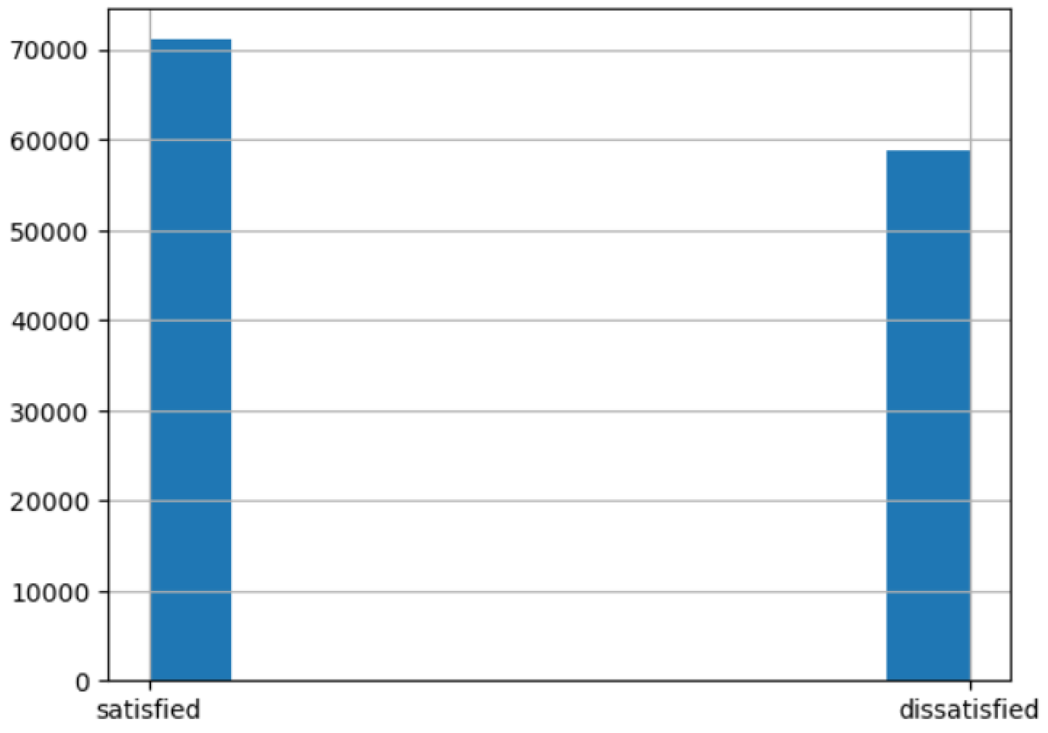


Figure 46

I assisted in feature selection and visualising the important features after Random Forest (Figure 47)

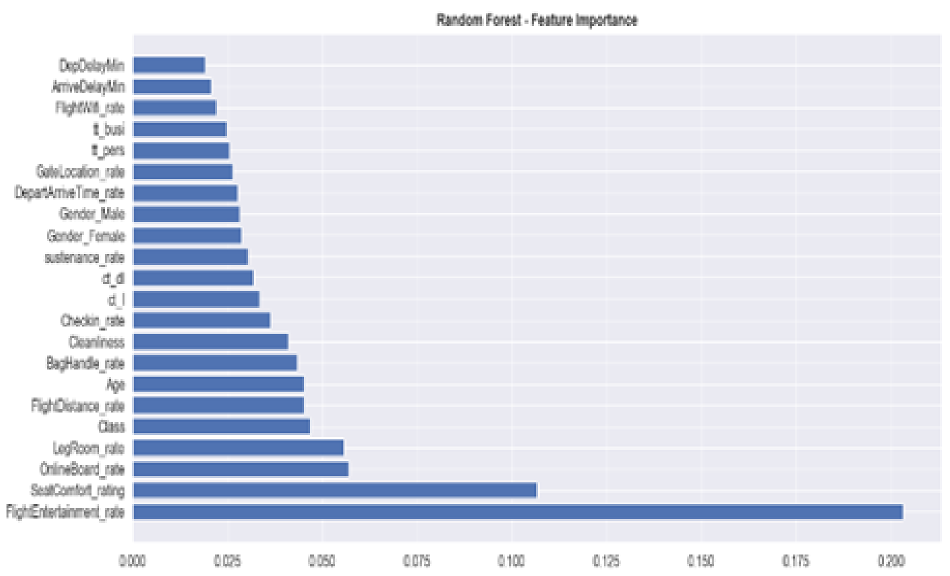
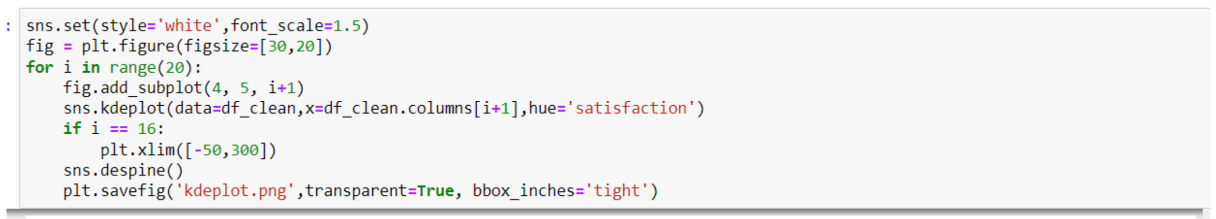


Figure 47

Another visualisation I employed was to look at the curves of all features after Random Forest between the dependant variables (Figure 48)



```
sns.set(style='white',font_scale=1.5)
fig = plt.figure(figsize=[30,20])
for i in range(20):
    fig.add_subplot(4, 5, i+1)
    sns.kdeplot(data=df_clean,x=df_clean.columns[i+1],hue='satisfaction')
    if i == 16:
        plt.xlim([-50,300])
    sns.despine()
    plt.savefig('kdeplot.png',transparent=True, bbox_inches='tight')
```

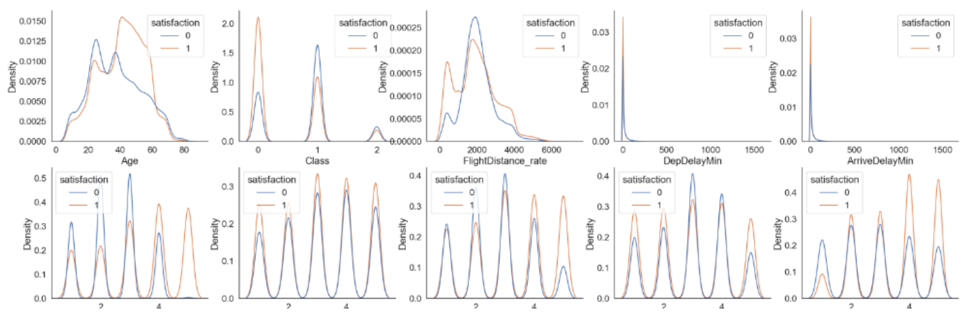
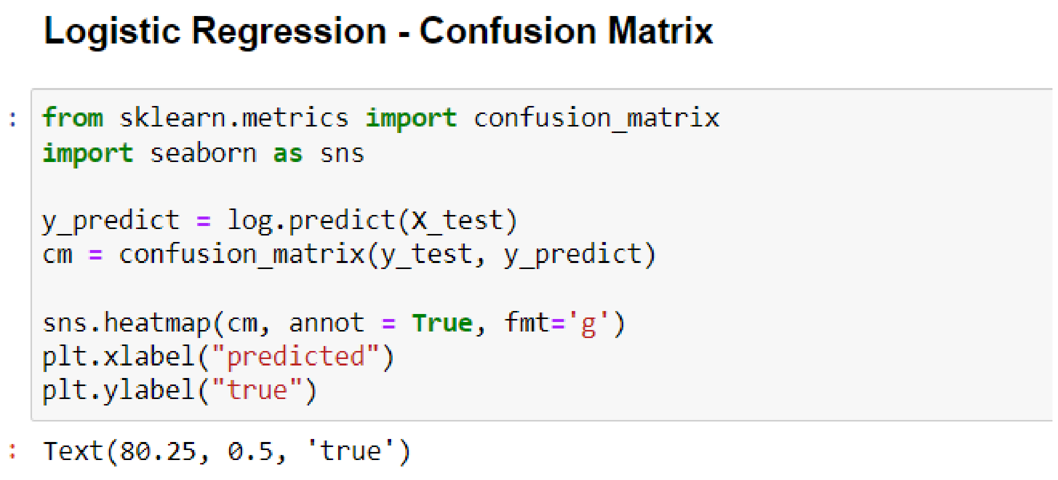


Figure 48

From a Machine Learning perspective, I concentrated on Logistic Regression as another algorithm to estimate the relationship between the dependant and independent variables and compare the accuracy of this method versus the prior two (Figure 49)



Logistic Regression - Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_predict = log.predict(X_test)
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm, annot = True, fmt='g')
plt.xlabel("predicted")
plt.ylabel("true")
```

```
Text(80.25, 0.5, 'true')
```

Figure 49

XVI. TEAM CONTRIBUTION

The effort for this project was very collaborative. We each had our strengths and unified our knowledge for the best outcome. A shared google drive was set up to exchange work and for version control. We each had a notebook of our own and plotted the best analysis, after regular meetings. If one team member had a better outcome with a method that was different, that method was applied to the master notebook. We had agendas after every meeting as to what our goal was for the next meeting, and we employed crisp dm techniques as well as roadmaps for our processes. Each team member has put their personal contribution effort into this report; however it was very even in that we interfaced on refining the best code for the outcome we wanted to achieve, therefore distribution of the workload was even (Figure 50)

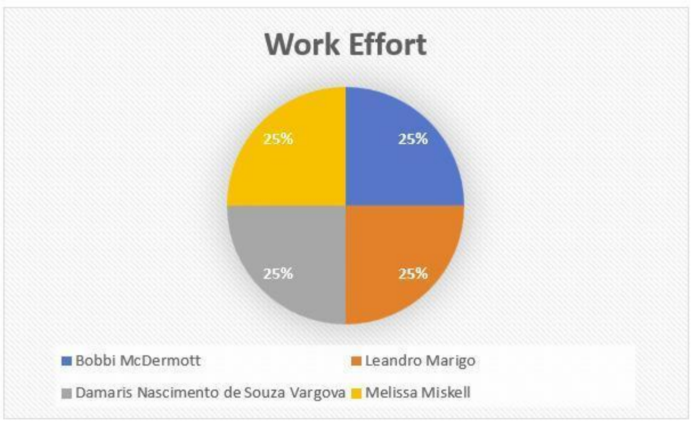


Figure 50

Words (142)

XVII. BIBLIOGRAPHY

Arya, N. (2022) *Logistic regression for classification*, *KDnuggets*. KDnuggets . Available at: https://www.kdnuggets.com/2022/04/logistic-regression-classification.html (Accessed: November 26, 2022).

Bewick, V., Cheek, L. and Ball, J. (2005) “Statistics review 14: Logistic regression,” *Critical Care*, 9(1), pp. 112–118. Available at: https://doi.org/10.1186/cc3045.

Dubey, A. (2018). *Feature Selection Using Random forest*. [online] Medium. Available at: https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f.

George, D., Mallery, P., 2010. IBM SPSS Statistics 25 Step by Step: A Simple Guide and Reference. Routledge.
Gogia, N. (2019). *Why Scaling is Important in Machine Learning?* [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/why-scaling-is-important-in-machine-learning-aee5781d161a.

LaValley, M.P. (2008) “Logistic regression,” *Circulation*, 117(18), pp. 2395–2399. Available at: https://doi.org/10.1161/circulationaha.106.682658.

Mahmood, M.S., 2022. Practical implementation of outlier detection in python [WWW Document]. Medium. URL https://towardsdatascience.com/practical-implementation-of-outlier-detection-in-python-90680453b3ce (accessed 11.25.22).

Michael P. Notter | Advanced exploratory data analysis (EDA) [WWW Document], n.d. URL https://miykael.github.io/blog/2022/advanced_eda/ (accessed 11.24.22).

Montelpare, W.J., Read, E., McComber, T., Mahar, A., Ritchie, K., 2020. Applied Statistics in Healthcare Research.

Raj, A., 2020. The Perfect Recipe for Classification Using Logistic Regression [WWW Document]. Medium. URL https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592 (accessed 11.25.22).

ROC curves in Machine Learning - AskPython, 2021. URL https://www.askpython.com/python/examples/roc-curves-machine-learning (accessed 11.26.22).