Airlines Customer Satisfaction

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

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.

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.

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)
```

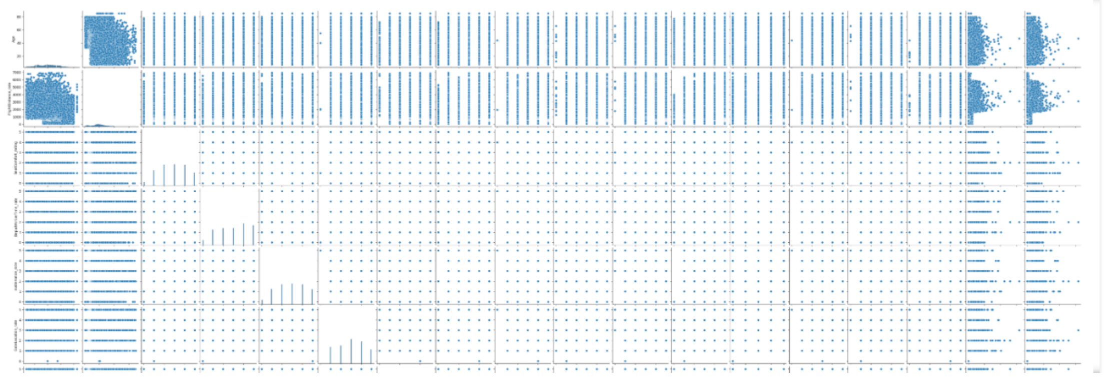


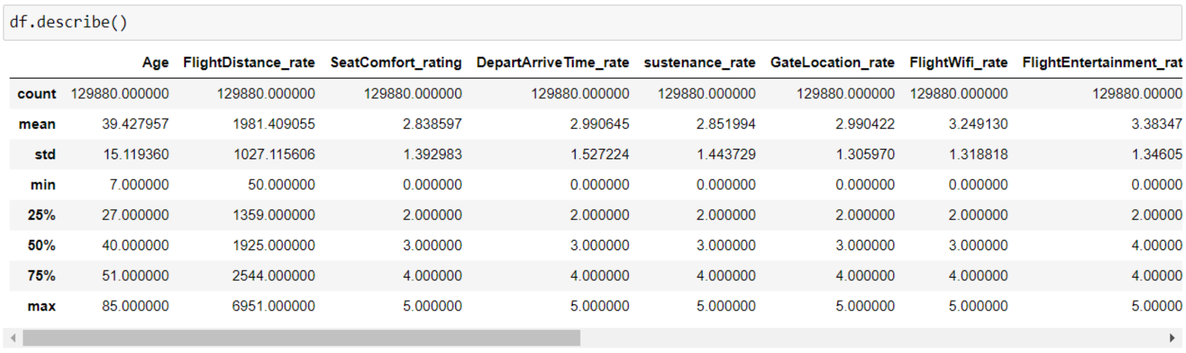
Figure 1

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()
```

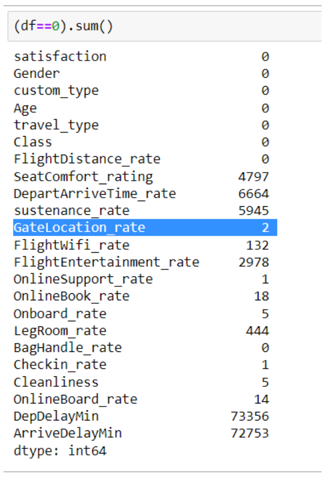
	Age	FlightDistance_rate	SeatComfort_rating	DepartArriveTime_rate	sustenance_rate	GateLocation_rate	FlightWifi_rate	FlightEntertainment_rat
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.00000
mean	39.427957	1981.409055	2.838597	2.990645	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

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)

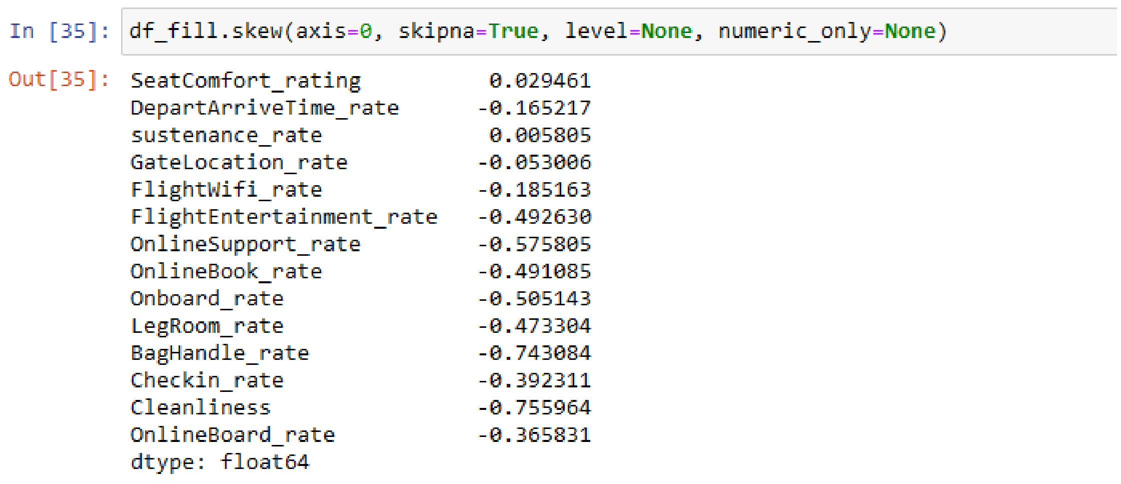


```
(df==0).sum()
```

```
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
```

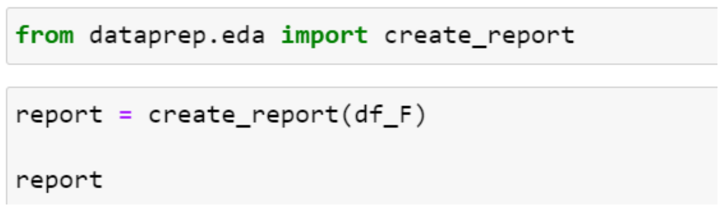
Figure 11

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.

V. Data Prep Library

Dataprep automated analysis 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)



```
from dataprep.eda import create_report
```

```
report = create_report(df_F)

report
```

Figure 12

VI. Feature Selection

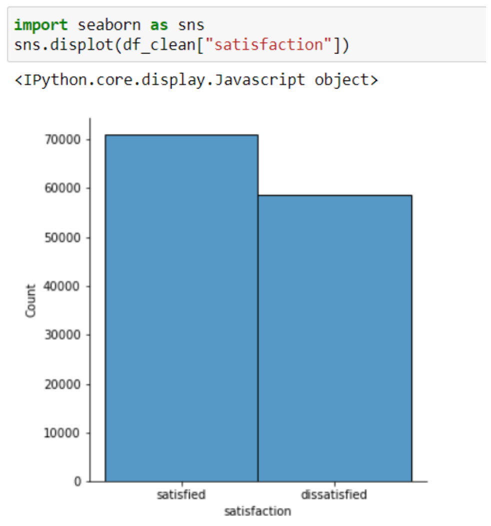
Feature selection as opposed to feature extraction has been determined to be the most optimal choice for reducing noise and maintaining important data for ML classification.

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.

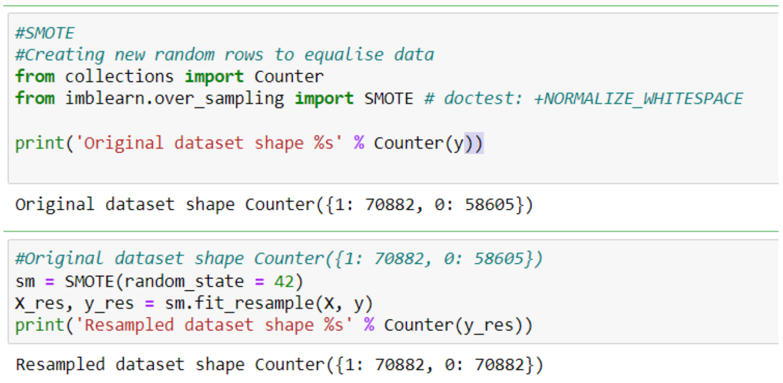
However, through visualisation imbalances were noted in the label category. To give the classification model an equal 50/50 chance of classifying for both classes, SMOTE was implemented as a method to balance this feature. (Figure 13), (Figure 14)



```
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})
```

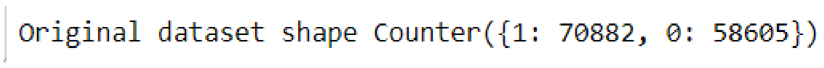
```
#Original dataset shape Counter({1: 70882, 0: 58605})
sm = SMOTE(random_state = 42)
X_res, y_res = sm.fit_resample(X, y)
print('Resampled dataset shape %s' % Counter(y_res))
```

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

Figure 14

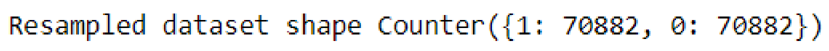
From a business perspective, the imbalance represents in favour of a higher count of satisfied customers as opposed to unsatisfied which is a favourable, although not an optimal, outcome for results.

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})
```

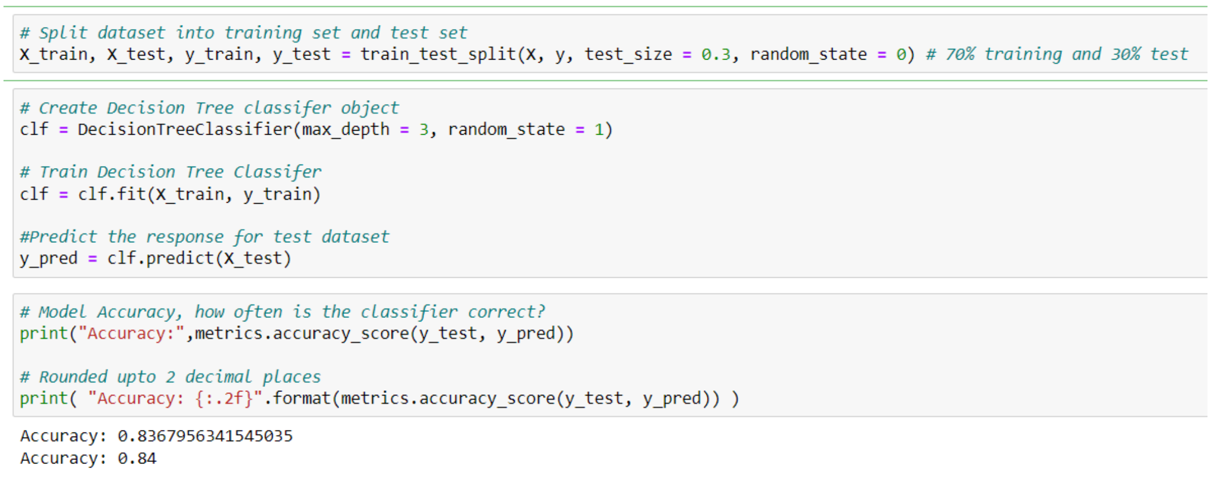
Figure 15



```
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)



```
# 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
```

```
# 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)
```

```
# 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)

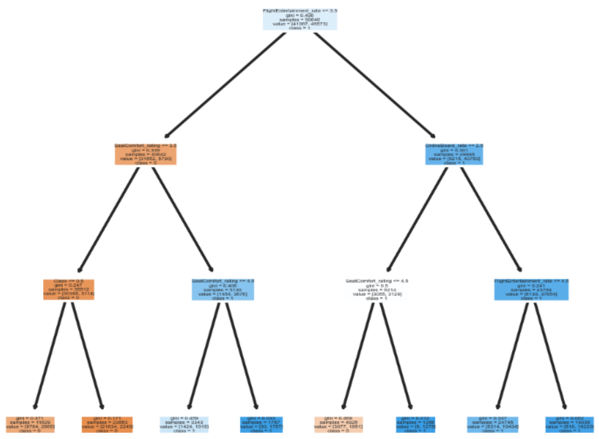


Figure 18

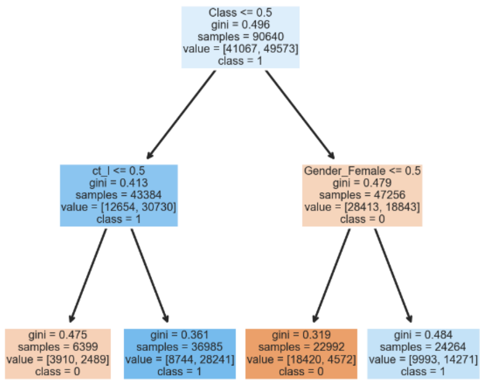


Figure 19

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)

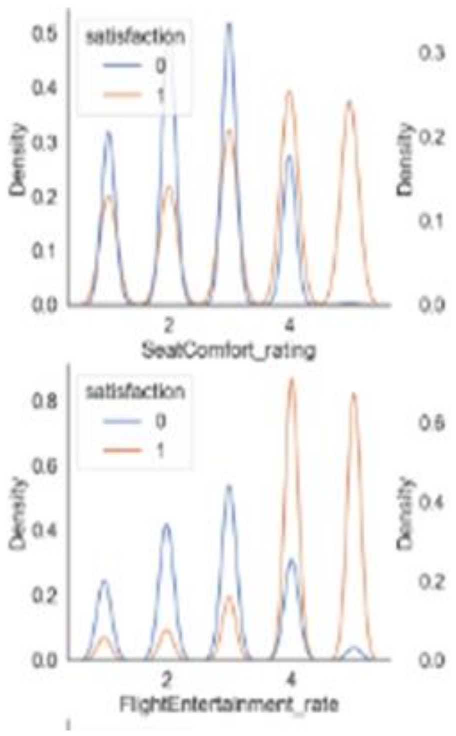


Figure 20

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)

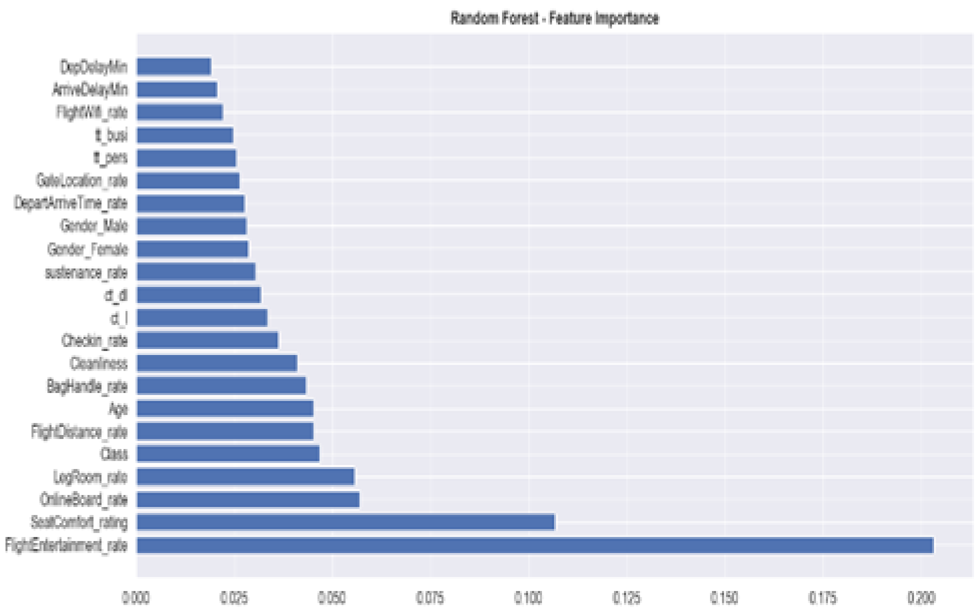


Figure 21

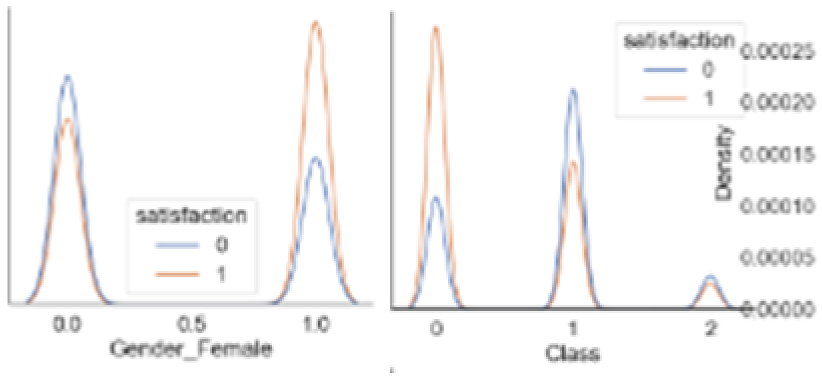
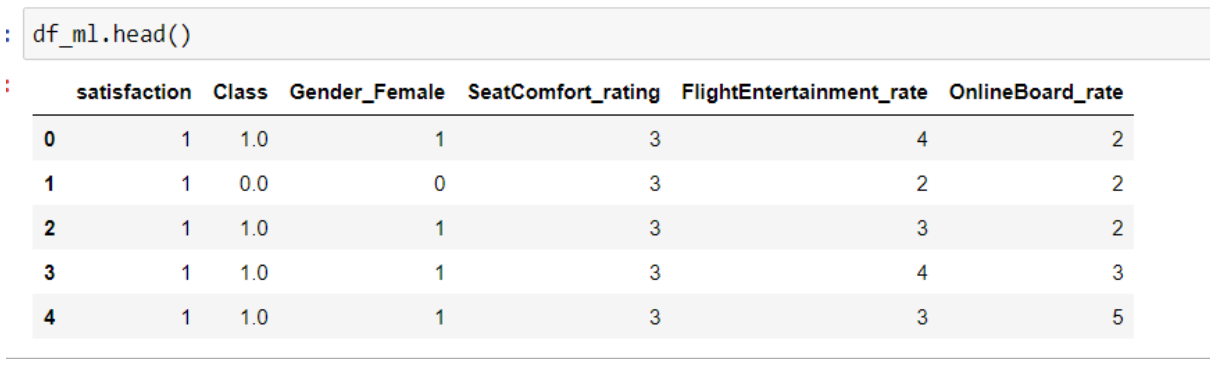


Figure 22

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

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:

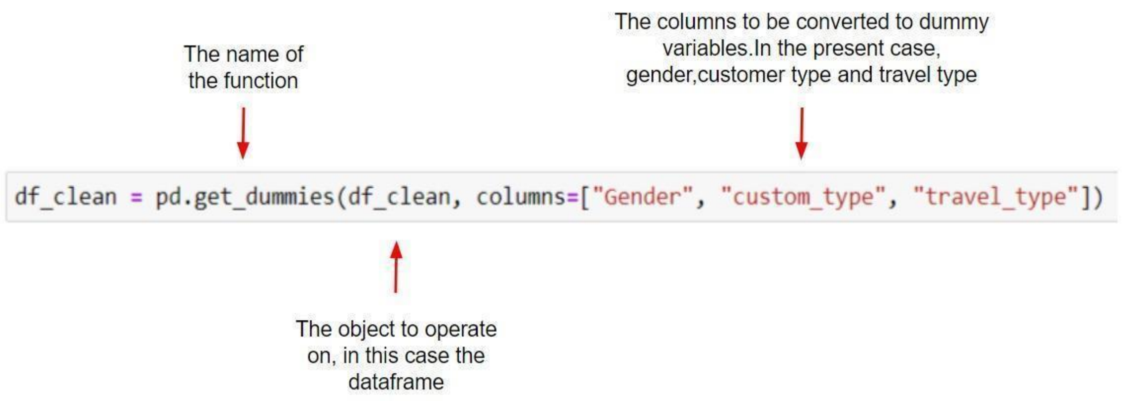
- A 1 encodes the presence of a category
- A 0 encodes the absence of a category

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)


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

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

Figure 24

ORDINAL ENCODING

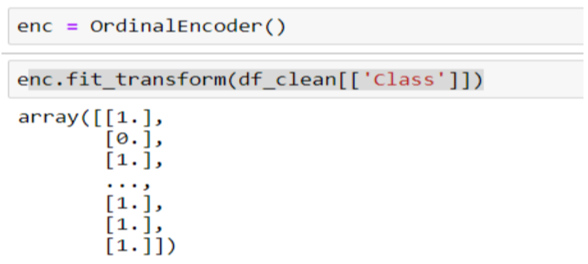
Once getdummies was applied, Ordinal Encoding was then used.

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.

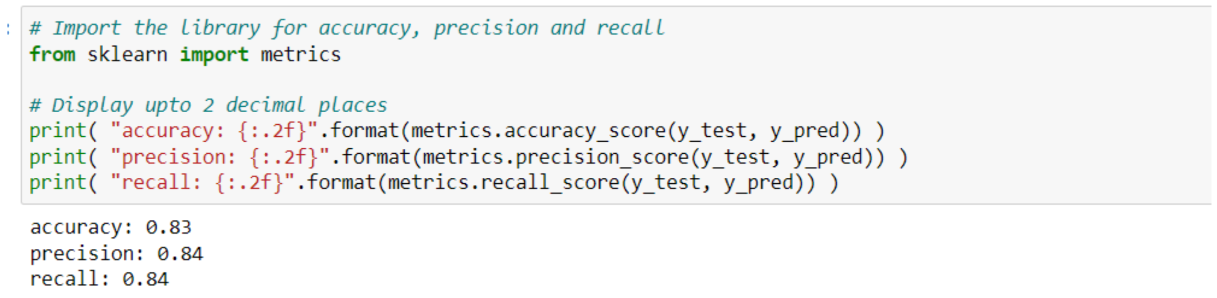
VIII. Scaling

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?)*

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)

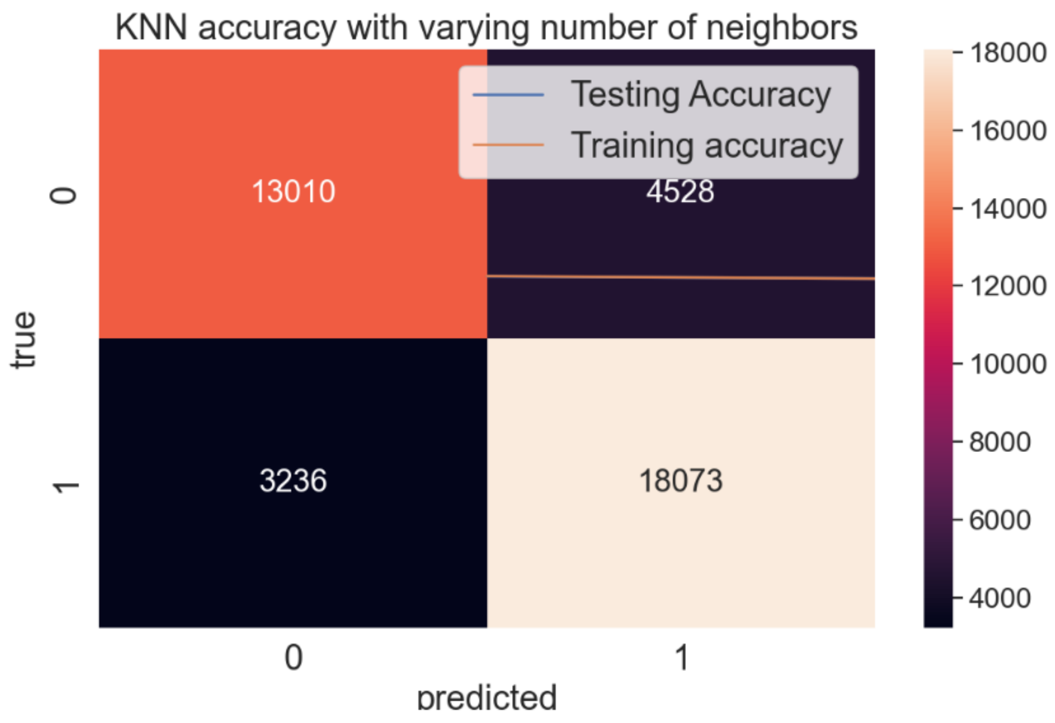
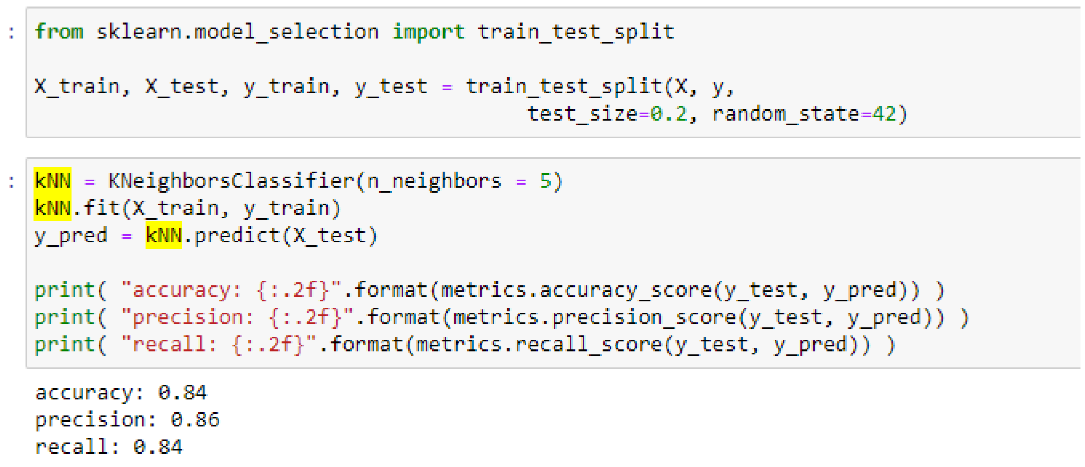


Figure 27

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)
```

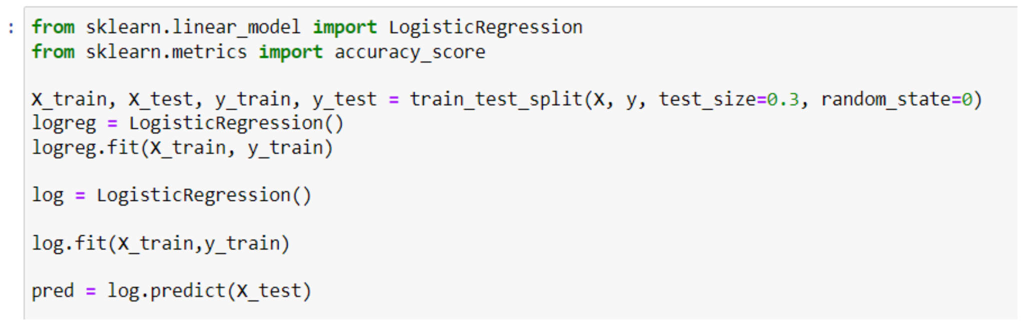
```
kNN = KNeighborsClassifier(n_neighbors = 5)
kNN.fit(X_train, y_train)
y_pred = kNN.predict(X_test)

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

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)

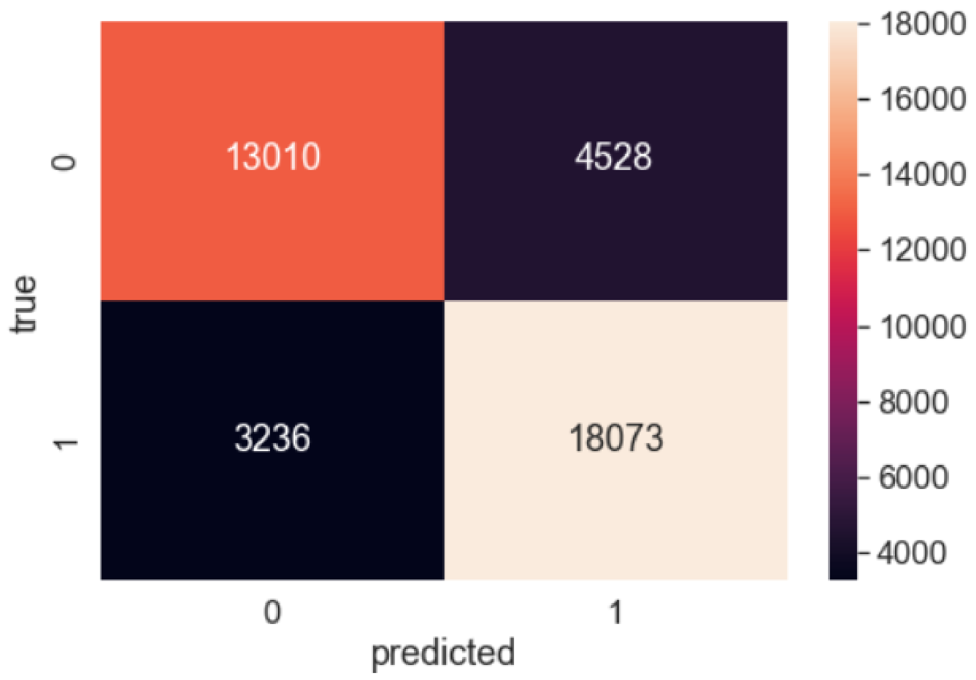
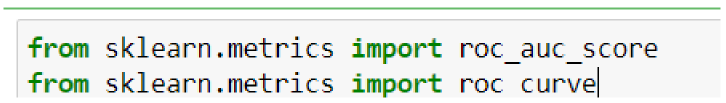


Figure 30

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

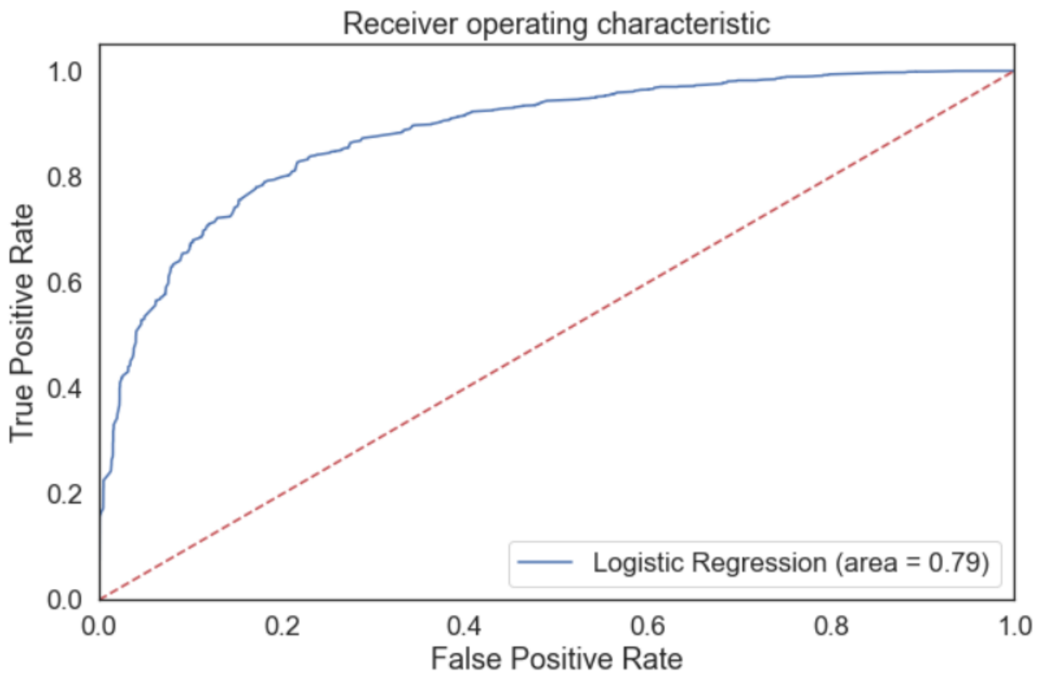


Figure 32

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.

<u>Feature Selection models:</u> Decision Tree and Random Forest
<u>Classification models:</u> 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.

XII. 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).