AeroSatisfaction Analytics

Fíonn Hourican & Eamon Cullimore

Student No's. 21477216, 21360701

Group 7

DCU

CSC1104 Data Warehousing & Data Mining

Instructor*: Prof. Mark Roantree*

Due Date*: 24/11/2024*

Table of Contents

Abstract

This project investigates predicting customer satisfaction on flights using data mining and machine learning. We aim to identify critical factors influencing satisfaction, applying preprocessing techniques to a substantial dataset to ensure compatibility with our prediction model. Using 80% of the data for initial training, we build and test a Random Forest Generator that is subsequently tested with the remaining 20% of the dataset to predict satisfaction levels. Through an iterative process, we adjust the factors included to determine their impact on prediction accuracy. Our findings offer insights that airlines can use to enhance customer experience by addressing areas that affect satisfaction most. This report details our dataset, data preparation steps, model design, results, and a thorough outcomes analysis.

1. Idea and Dataset Description

Our goal is to identify the factors that most significantly impact passenger satisfaction, enabling airlines to enhance the flight experience based on data-driven insights. By analysing both demographic details and aspects of the flight experience, we aim to pinpoint which factors, such as flight delays, seating comfort, or onboard services, play a crucial role in determining customer satisfaction. Understanding these factors helps build effective predictive models while providing actionable insights that airlines can implement to improve customer experience, potentially increasing customer retention and loyalty.

For our project, we utilise the **Airline Passenger Satisfaction** [Appendix: 1] Kaggle dataset, which contains 129,880 records and 23 features, to analyse and predict passenger satisfaction based on various service and demographic factors. The dataset is organised into three categories that provide insights into passenger experiences.

- **Demographic Information** includes passenger gender, customer type (loyal or new), age, type of travel (business or personal), and class of service (e.g., Business, Economy, etc). These features help us analyse satisfaction trends across diverse passenger profiles, enabling the model to assess whether factors like loyalty or age group influence satisfaction levels.
- **Flight Information** captures variables such as flight distance and departure and arrival delays, which can significantly affect passenger comfort. Longer distances may impact overall satisfaction due to prolonged seating, while delays are known to impact customer experience and potentially lower satisfaction ratings.
- **In-flight services and Amenities** cover a range of onboard experiences, each rated on a scale from 1 to 5. These include ratings for inflight Wi-Fi service, time convenience, food and drink quality, seat comfort, entertainment, onboard service, and cleanliness. By examining these factors, we aim to understand which amenities passengers value most, shedding light on service areas that may require improvement.

The dataset's structure supports a comprehensive analysis of the factors influencing passenger satisfaction, with the target variable being Satisfaction (either "satisfied" or "neutral/dissatisfied"). With a reasonably balanced split between satisfied (44%) and unsatisfied (56%) passengers, the dataset is well-suited for predictive models, such as a random forest, to determine the likelihood of passenger satisfaction.

For our analysis, we will use 20% of the data, **25,976** records, to train the model and 80%, **103,904** records, for testing its performance, ensuring robust validation of our predictions.

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	4	3	4	4	4	5	0	0.0	satisfied
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	3	4	4	3	3	3	0	0.0	satisfied

5 rows × 25 columns

Fig 1.1: Head of original data set.

2. Data Preparation

Our data preparation began with identifying and **addressing missing data** in the dataset. Using a Python script to scan the dataset, we found only 83 rows with missing values, primarily in the *Arrival Delay in Minutes* column. Since the missing data was minimal, we opted to drop these rows to retain overall data quality.[1]

```
# Show the shape of the dataset before dropping values
print(train2.shape)

# Drop the values without returning a new dataset
train2.dropna(inplace=True)

# Show the size of the dataset with missing values removed
print(train2.shape)
```

Fig 2: This code snippet shows how we checked for null values, dropped rows with missing values and rechecked.

We also considered other imputation methods. These included manually filling in missing values, though this was impractical given the dataset size; using a global constant, like setting missing values to zero, though this could misrepresent the data; applying the mean or median values, which would risk distorting the arrival delay distribution; and calculating the most probable value based on related fields like *Departure Delay in Minutes*, which, while potentially effective, added unnecessary complexity.[2]

Following the handling of missing data, we proceeded with **dimensionality reduction** to streamline our dataset before any further data preparation. First, we applied feature selection, removing irrelevant attributes like the ID and row number columns, as these unique identifiers had no impact on passenger satisfaction or flight experience. This step allowed us to focus solely on the most relevant features, enhancing the dataset's efficiency for our predictive model.

To enhance the accuracy and reliability of our machine learning model, we used an **outlier removal** technique based on the Interquartile Range (IQR). Outliers can introduce bias, especially in skewed datasets like ours, which may distort model performance.

```
# Function to remove outliers based on IQR
def remove_outliers_iqr(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1  # Interquartile range
    # Define lower and upper bounds for detecting outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Detect outliers in the specified column
    outliers = df[(df[column_name] < lower_bound) | (df[column_name] >
upper_bound)]

    # Create a cleaned dataset by removing the outliers
    cleaned_df = df[(df[column_name] >= lower_bound) & (df[column_name] <=
upper_bound)]
```

Fig 2.1: This block shows how we calculated the IQR and removed outliers.

Using the IQR method mentioned, we identified and removed outliers from key numerical features, including *Flight Distance, Age, Departure Delay in Minutes,* and *Arrival Relative to Departure*. This method effectively highlights extreme values outside the dataset's typical range.

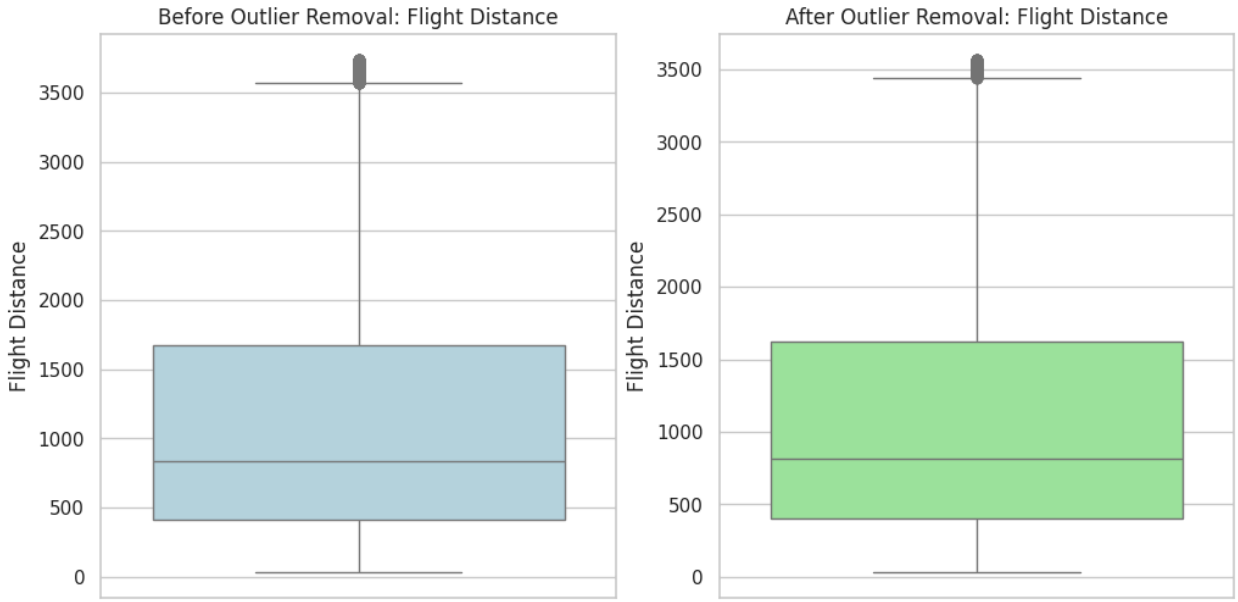


Fig 2.2: To assess the impact of outlier removal, we visualised the data with box plots before and after the process to confirm that the overall distribution remained consistent. This step effectively smoothed the data, reducing noise and ensuring our model would not be skewed by extreme, anomalous values.

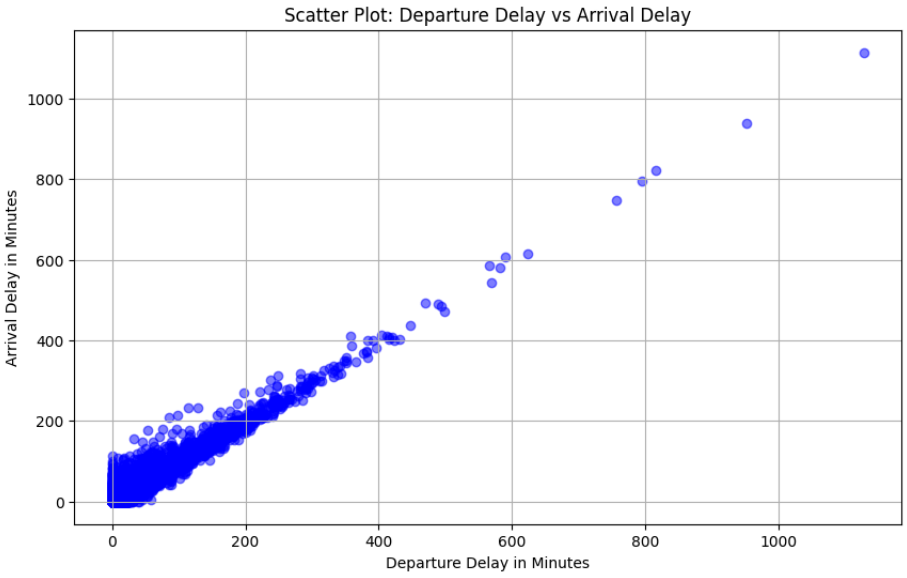


Fig 2.3: The Scatter Plot clearly shows a strong correlation between Departure Delay in minutes and Arrival Delay in minutes.

We initially debated removing the *Arrival Delay in Minutes* feature due to its strong correlation with *Departure Delay in Minutes*, which offered limited insights. Ultimately, we performed feature extraction to create a new feature, *Arrival Relative to Departure*, to gain deeper insights. This transformation distinguishes between flights that depart late but arrive on time versus those that experience delays in departure and arrival. For example, a flight delayed by 30 minutes at departure but arriving on time may affect customer satisfaction differently than a flight that departs 30 minutes late and arrives an hour late. The correlation between *Departure Delay in Minutes* and *Arrival Delay in Minutes* was .97, indicating a strong relationship, whereas the correlation with *Arrival Relative to Departure* dropped to -.07. This new feature should enrich our machine learning model with valuable information.

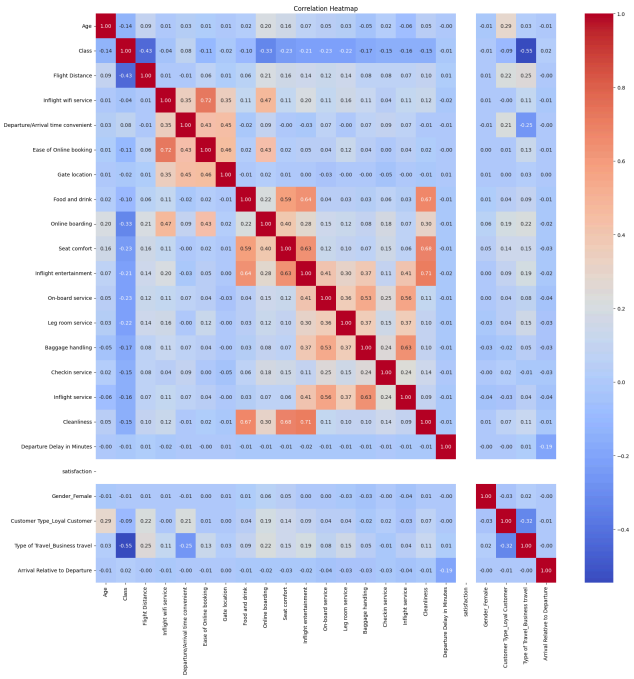


Fig 2.4: Heatmap showing correlation after feature extraction.

We applied binning as a **data smoothing** technique to enhance our dataset's clarity by converting continuous data into discrete intervals, or "bins." This approach generalises the data by grouping similar values, reducing the impact of minor fluctuations and outliers and making patterns easier to identify.[9] By transforming continuous variables into categorical bins, we simplified complex distributions, particularly in skewed datasets, improving interpretability and robustness for machine learning analysis.

To create the bins, we developed a function that determines the number of bins based on the square root of the total data points and calculates the bin width by dividing the range of values by the number of bins.[5]

For our analysis, we used equal-frequency binning for several columns, including *Flight Distance, Age, Departure Delay in Minutes,* and *Arrival Relative to Departure*. Initially, we attempted equal-width binning for *Age*, but it distorted the data's mean. Switching to equal-frequency binning allowed us to maintain similar observations across bins, capturing relationships within the data.

We addressed the left-skewed distributions in *Flight Distance, Departure Delay,* and *Arrival Relative to Departure* by applying Equal-Frequency Binning, ensuring each bin accurately reflected passenger experiences regarding delays. This was crucial for understanding how delays impact overall satisfaction.

For smoothing bin values, we explored three methods: Mean Replacement, which effectively reduced noise but sometimes sacrificed detail due to outliers; Median Replacement, which

proved more robust against outliers and provided a better central tendency for skewed data; and Bin Edges Replacement, which preserved extremes within the data but resulted in a loss of interpretability.

Finally, we chose to smooth the values using the **median** method for most features. The median provided a better central tendency in the presence of skewed data and outliers, making it the most suitable option for smoothing while preserving the integrity of the data distribution. This approach was particularly effective in improving the robustness of our machine learning models, as it reduced the impact of extreme values without distorting the overall structure of the data.

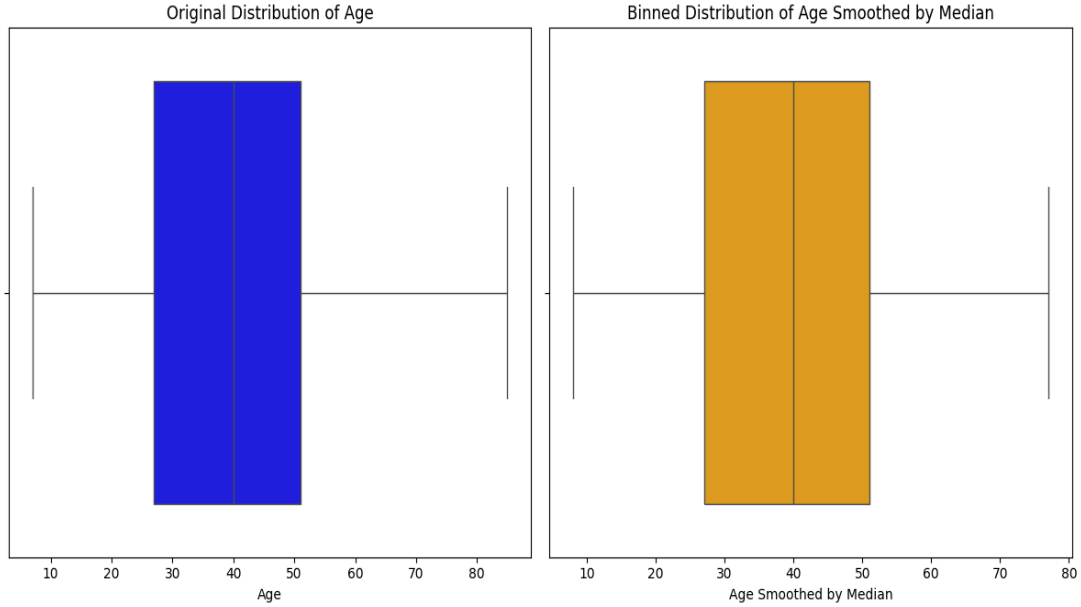


Fig 2.5: Box Plots show the data was smoothed and the shape was kept the same.

Upon examining the dataset, we identified several categorical fields in string format that were incompatible with our algorithm. To enable their use, we converted these strings into integers using two methods:

Ordinal Encoding: We used this for the *Class* column, which contained multiple ordered values. We defined a priority order, assigning ranks: Business (1), Eco Plus (2), and Eco (3). We then created a dictionary to map each class to its corresponding rank and employed a mapping function to replace string values with integers.[8]

One-Hot Encoding: This method was applied to most string columns, converting values into binary (1 or 0) and creating new columns to represent them. For columns like *gender*, *customer type*, and *type of travel*, we applied additional encoding to eliminate bias. We retained the first binary value in each column and dropped the other new columns to streamline the data.[6]

We employed a tree map to visualise the dataset hierarchy. While insights were somewhat limited, they indicated that satisfied customers tended to fly further than unsatisfied ones. Notably, the most satisfied customers were often business-class, loyal travellers. This suggests a potential connection between satisfaction and long-haul travel, where the enhanced amenities of

business class and loyalty benefits, such as lounge access or upgrades, may contribute to a more positive experience.

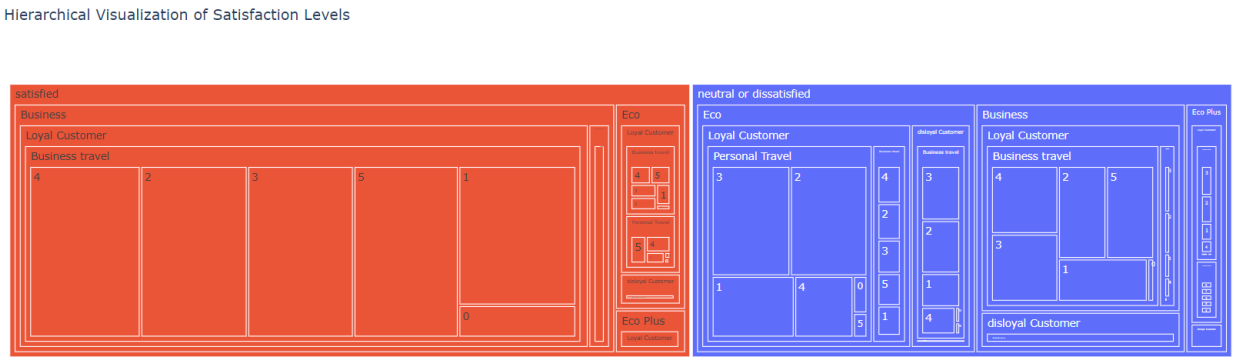


Fig 2.6: Treemap where the flight distance determines the size of the rectangles, and each rectangle represents a different category in a hierarchical form.

3. Algorithm Description

This project's chosen machine learning algorithm is the Random Forest classifier, a robust ensemble learning method. A Random Forest consists of multiple decision trees, each trained on a randomly sampled subset of the data with added noise. This randomness fosters diversity among the trees, enhancing the model's accuracy and robustness. [3]

Each tree is created using bootstrapping, where training data is sampled with replacement, allowing the model to average predictions across multiple trees. This reduces variance and overfitting, improving generalisation accuracy [6]. Additionally, Random Forests randomly select features at each split, preventing any single feature from dominating and promoting diversity among the trees, which is particularly beneficial for complex datasets with noisy features.

The out-of-bag (OOB) error provides a built-in validation measure, as each tree uses a portion of the data for training while leaving some for internal validation. Coupled with feature importance scoring, Random Forests deliver accuracy and interpretability, making them widely used for various classification and regression tasks. [4][7].

The Random Forest classifier is initialised with 100 decision trees, a standard choice that balances performance and computational efficiency. The maximum number of features considered for splitting at each node is set to the square root of the total number of features, a common practice to reduce overfitting and enhance model diversity. By specifying random_state=42, we ensure reproducibility by fixing the seed for random number generation.

```
rf = RandomForestClassifier(n_estimators=100,
max_features='sqrt', random_state=42)
rf.fit(train2, train2_target)
```

Fig 3.1

4. Results and Our Analysis

A **confusion matrix** offers valuable insights into a classifier's performance across different classes by breaking down its results into True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

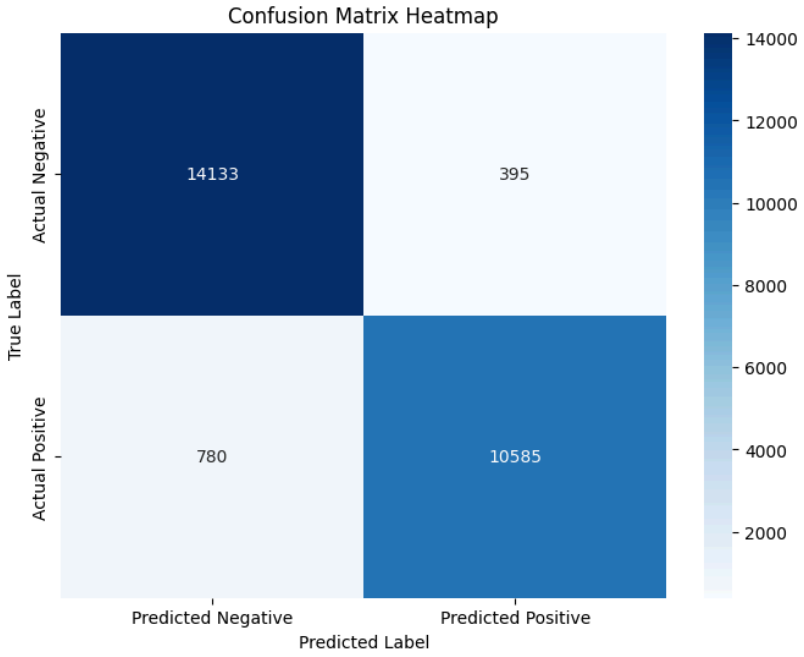
```
tn, fp, fn, tp = confusion_matrix(test2_target, test_pred).ravel()

# Create a data frame for the confusion matrix
data = {'Predicted Negative': [tn, fn],
        'Predicted Positive': [fp, tp]}
df = pd.DataFrame(data, index=['Actual Negative', 'Actual
Positive'])

# Create the heatmap
plt.figure(figsize=(8, 6))  # Adjust figure size if needed
sns.heatmap(df, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

Fig 3.2: This code calculates the confusion matrix for a classifier's predictions, extracts TN, FP, FN, and TP values, organises them into a data frame for readability, and then prints the matrix format to show actual vs. predicted classifications.

Fig 3.3: Confusion matrix



- **True Negatives (TN):** *14,133* cases were correctly classified as negative.
- **False Positives (FP):** *395* cases were incorrectly classified as positive when they were negative.
- **False Negatives (FN):** *780* cases were incorrectly classified as negative when they were positive.
- **True Positives (TP):** *10,585* cases correctly classified as positive.

The classifier achieved 24,718 correct predictions out of the total cases, demonstrating a high overall accuracy. However, the 1,175 misclassifications, split between 395 false positives and 780 false negatives, reveal specific areas for improvement. The higher count of false negatives suggests that the classifier may struggle more with correctly identifying positive cases. This imbalance could point to an opportunity to fine-tune the model to make more accurate predictions.

Accuracy, or recognition rate, measures how closely a model's predictions match actual values in a dataset. Calculated as **(TP+TN)/(TP+TN+FP+FN)(TP + TN) / (TP + TN + FP +**

FN)(TP+TN)/(TP+TN+FP+FN), it indicates the percentage of correct predictions, with values closer to 1 signalling more robust performance.

```
# Calculate Accuracy
accuracy = accuracy_score(test2_target,
test_pred)
print("Accuracy:", accuracy)
```

FIG 3.4: utilising sklearns accuracy_score() function to return a decimal representing our accuracy rating for our data.

Accuracy: 0.9546209400223998

Our model achieved an accuracy of 95%, correctly classifying 95% of ~25,000 rows with a misclassification rate of only 5%. This high accuracy, backed by a 96.8% F1 score, suggests the model is robust in predicting airline satisfaction and can offer reliable insights for data-driven improvements.

The **error rate** quantifies the percentage of incorrect predictions made by an algorithm on a dataset. It can be calculated either as **(FP+FN)/(TP+TN+FP+FN)(FP + FN) / (TP + TN + FP + FN)(FP+FN)/(TP+TN+FP+FN)** or, if accuracy is known, as simply **1−accuracy**.

Our model achieved an accuracy of 95%, correctly classifying 95% of ~25,000 rows with a misclassification rate of only 5%. This high accuracy, backed by a 96.8% F1 score, suggests the model is robust in predicting airline satisfaction and can offer reliable insights for data-driven improvements.

```
# Calculate Error Rate
error_rate = 1 - Accuracy
print("Error Rate:", error_rate)
```

Fig 3.5: Since our accuracy was ~0.95, we used the latter formula

Error Rate: 0.04537905997760017

Sensitivity (or recall) measures how effectively a model identifies true positives within a specific category, such as passenger satisfaction. In this context, sensitivity is calculated by dividing the number of true positives by the sum of true positives and false negatives.

```
# Calculate Sensitivity
sensitivity = tp / (tp + fn)
print("Sensitivity:",
sensitivity)
```

Fig 3.6: Calculating Sensitivity.
Sensitivity: 0.9313682358117026

At 93%, sensitivity shows the model effectively captures true positives, minimising missed satisfied customers. This level of sensitivity provides airlines with reliable insights into which customers are satisfied, supporting efforts to enhance the passenger experience.

Specificity (or true negative rate) measures how well a classifier correctly identifies negative cases.

Specificity: 0.9728111233480177

Fig 3.7: Calculate Specificity

A specificity of 97.3% demonstrates the model's strength in identifying dissatisfied customers and avoiding false positives.

```
# Calculate Specificity
specificity = tn / (tn + fp)
print("Specificity:",
specificity)
```

Precision is a measure of accuracy that represents the percentage of instances labelled as positive that are, in fact, true positives.

Precision, at 96.4%, further validates the model's accuracy in labelling satisfied customers, ensuring that predictions are trustworthy. This high precision helps airlines understand their genuinely satisfied customers, forming a clear basis for maintaining satisfaction.

```
# Calculate Precision
precision = tp / (tp + fp)
print("Precision:",
precision)
```

Fig 3.8: Calculate Precision
Precision: 0.9640255009107468

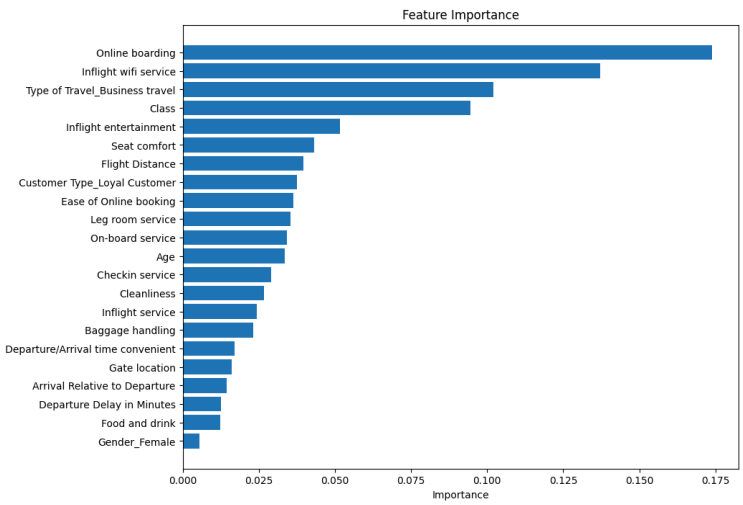
The **F1 score** combines precision and recall (or sensitivity) to provide a balanced measure of the model's accuracy. It is particularly valuable when both false positives and false negatives need to be managed. It is calculated as the harmonic mean of precision and specificity.

The F1 score of 96.8% reflects a balanced performance in precision and specificity, indicating the model's ability to distinguish satisfied from dissatisfied customers. This comprehensive score underscores the model's effectiveness as a reliable tool for customer satisfaction analysis.

```
# calculate F, F1, F-score
f1_score = 2 * (precision *
specificity) / (precision +
specificity)
print("F1 Score:", f1_score)
```

Fig 3.9: Calculate F1-score
F1 Score: 0.9683983860394262

These performance metrics collectively provide a clear, multi-dimensional picture of the model's effectiveness. Accuracy and error rate give a general sense of correct classifications, while sensitivity and specificity show how well the model differentiates between satisfied and dissatisfied customers. Precision adds confidence that the satisfied predictions are genuinely accurate, and the F1 score balances both precision and specificity, underscoring the reliable identification of satisfied and unsatisfied customers. The consistently high values across these metrics confirm that the model is accurate and effective at minimising misclassifications. This makes the model a robust tool for customer satisfaction predictions in an airline context.



In analysing **feature importance** within our dataset, we initially expected *Flight Distance* to rank highly due to its large numerical values across rows. However, the results revealed a different picture: the top five features were *Online Boarding, Inflight Wifi, Type of Travel (specifically Business Travel), Class, and Inflight Entertainment.* This ranking suggests that customers prioritise convenient online boarding and quality inflight wifi above other factors.

Fig. 3.10: Feature Importance Graph

Features like *Class* and *Customer Type* performed as anticipated, reflecting their importance in determining higher satisfaction ratings. Contrary to expectations, *Flight Distance* ranked less highly despite its wide range of values across rows. We assumed not normalising this feature would lead to bias; however, the final rankings indicated normalising was unnecessary, confirming an acceptable model outcome.

4.1 Experiment

We experimented with refining the dataset by dropping specific columns. First, we remove features airlines cannot control, like *Flight Distance, Age,* and *delays*, to focus on factors directly impacting satisfaction(rf2). Next, we drop the four low-importance features to reduce noise and emphasise key predictors(rf40). Finally, we combine both sets(rf5), creating a streamlined dataset of influential, actionable features to evaluate any gains in predictive accuracy and interpretability. Interestingly, reducing to 16 features did not affect our model performance, as shown.

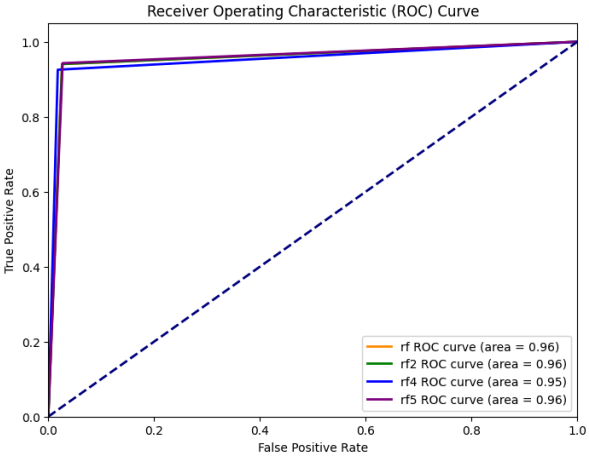


Fig 3.11: ROC Curve

4.2 Reflection

We learnt a lot about data preprocessing during this project. We initially simply removed entries with missing values and encoded the data, which was enough to train our model. However, we learnt the importance of eliminating outliers, smoothing the data, and removing unnecessary features. Removing outliers and smoothing the data through binding helped reduce noise. Reducing the number of features reduces computational power. All these techniques helped make our model more efficient and reliable.

Another critical feature we learned was our dataset's suitability for machine learning. When we researched our dataset, it was discovered that there was not much preprocessing since most of the data was in ranges of 0-5. Out of the 23 features in our dataset, only four features needed encoding to numerical values. This highlighted how suitable our data was for this assignment and machine learning in general. Most importantly, this dataset was instrumental in a real-world application. People always complain about airlines, and this dataset was perfect for training a model to predict customer satisfaction.

Finally, using this model, we see what features are essential to the consumer. The ease of online boarding is evident in hindsight, allowing customers to skip queues. Similarly, having Inflight Wi-Fi makes trips more enjoyable, as customers can enjoy things like Netflix or texting friends. Although these insights seem obvious, we never would have ranked them as the most essential features. This shows us how data analysis can uncover valuable, real-world insights that might otherwise go overlooked.

Appendix

1. Database on Kaggle
    - https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction
2. Google collab containing our project
    - https://colab.research.google.com/drive/12Lj1qRirPaDhZGgVDSBtA4VP6VDhdY?usp=sharing

References

1. Datacamp (2023), Top Techniques to Handle Missing Values Every Data Scientist Should Know [Online] Available At: https://www.datacamp.com/tutorial/techniques-to-handle-missing-data-values [Accessed 7th October 2024]
2. DataDaft (2020) How to Detect and Fill Missing Values in Pandas (Python) [Online] Available at: https://youtu.be/AbBZYHNYFaY?si=OCPuF4ggAk9H_EAH [Accessed 7th October 2024]
3. Google. (2024). Random forests. [Online]. developers.google. Available at: https://developers.google.com/machine-learning/decision-forests/random-forests#:~:text=A%20random%20 [Accessed 7 November 2024].
4. Liaw, Andy & Wiener, Matthew. (2002). Classification and Regression by randomForest. R News. 2(3), pp.18-22. [Online]. Available at: https://journal.r-project.org/articles/RN-2002-022/RN-2002-022.pdf [Accessed 25 October 2024].
5. Qimacros. (nd). Struggling with Histogram Bin Width and Bin Intervals?. [Online]. Qimacros. Available at: https://www.qimacros.com/histogram-excel/how-to-determine-histogram-bin-interval/#:~:text=Calculate%20the%20number%20of% [Accessed 4 October 2024].
6. QuantStart. (2024). Bootstrap Aggregation, Random Forests and Boosted Trees. [Online]. quantstart.com. Available at: https://www.quantstart.com/articles/bootstrap-aggregation-random-forests-and-boosted-trees/ [Accessed 25 October 2024].
7. scikit-learn. (2024). Supervised Learning. [Online]. scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/ensemble.html#random-forests [Accessed 25 October 2024].
8. SitePoint (2023) An Introduction to Data Encoding and Decoding in Data Science (Online) Available at: https://www.sitepoint.com/data-encoding-decoding-data-science-introduction/ [Accessed 7th October 2024]

9. Utkarsh. (12 June 2023). What is Binning in Data Mining?. [Online]. Scaler. Available at: https://www.scaler.com/topics/binning-in-data-mining/ [Accessed 4 October 2024].