Predicting Airline Passenger Satisfaction Using Machine Learning Models

Anthony Pham
University of Minnesota/FOM
pham0520@umn.edu

Abstract— This study aims to develop a predictive model to classify airline passenger satisfaction based on survey responses. By analyzing key factors that contribute to satisfaction and dissatisfaction, this research provides insights into areas where airlines can improve customer experience. The dataset, sourced from Kaggle, consists of various categorical and numerical features representing passenger demographics, flight conditions, and service quality. The project applies machine learning classification techniques to identify patterns and correlations, offering actionable insights for the airline industry.

I. Introduction

In recent years, the airline industry has faced increasing competition, making passenger satisfaction a critical factor in customer retention and business success. Understanding the factors that influence passenger satisfaction can help airlines allocate resources effectively and improve service quality. Machine learning classification techniques have proven useful in analyzing large datasets and identifying patterns that may not be immediately apparent through traditional methods.

This study aims to develop a classification model to predict passenger satisfaction based on various factors, such as service quality, flight conditions, and passenger demographics. The model classifies passengers as either "satisfied" or "neutral/dissatisfied" using 24 service-related and demographic features. The dataset, sourced from Kaggle, underwent data analysis, preprocessing, feature engineering, and model training and evaluation to determine the most influential factors affecting satisfaction. By identifying key determinants, airlines can implement targeted improvements to enhance customer experience and operational efficiency.

II. Preprocessing the Dataset

The dataset initially contained 103,904 samples and 25 features. During preprocessing, the "Unnamed" index column and the ID column were removed, as they provided no meaningful information for predictive modeling. To easier facilitate feature engineering, all categorical variables were explicitly set to the *category* data type.

Next, the dataset was examined for missing values, revealing that the *Arrival Delay in Minutes* feature had 310 missing values. Since this variable was highly right-skewed (Figure 1), the missing values were imputed using the median to prevent further distortion in the distribution.

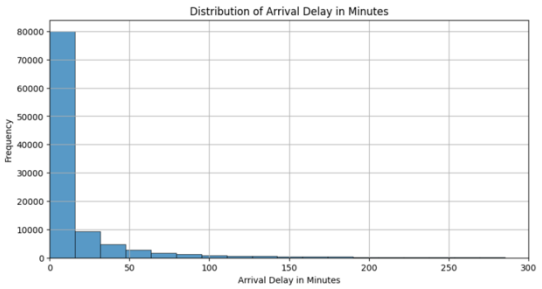

Figure 1: Right Skewed distribution of arrival Delay -> filling this feature with Median

Them using Bar Graphs for Exploratory data analysis, visual cues revealed key insights into passenger satisfaction (Figure 2). Features such as gender, customer type, and gate location had minimal impact on overall satisfaction. However, age played a significant role, as older passengers (above 38 years old) tended to report higher satisfaction levels. Additionally, business travelers were far more likely to be satisfied than leisure travelers, with nearly all satisfied passengers flying for business purposes. Class of service also significantly influenced satisfaction, with business class passengers reporting the highest satisfaction levels, whereas economy class passengers showed lower satisfaction ratings. Furthermore, service-related factors— Wi-Fi quality, in-flight entertainment, food, seat comfort, baggage handling, and cleanliness; were found to have a substantial impact on passenger satisfaction.

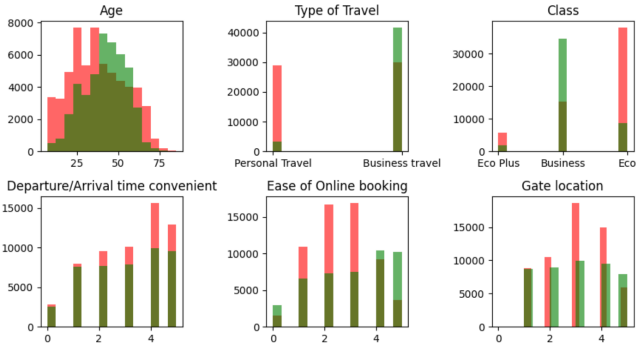

Figure 2: Sample of Plotted features, shows distribution of satisfied(green) to neutral/dissatisfied(red) passengers for each feature

III. Feature Enginering and Selection

Since the dataset contains both categorical and numerical features, the features were categorized into three groups: numerical, categorical binary, and categorical non-binary. This separation was implemented to ensure a more organized and efficient feature engineering process.

1. *Encoding Categorical Features:*
- Binary categorical features - *Gender, Customer Type,* and *Type of Travel*; were encoded by assigning values of 0 or 1.
- Non-binary categorical features (multiclass categories) were encoded using one-hot encoding, which generated dummy variables, and which each feature into the number of classes the feature contained into columns.

This method was selected to prevent the model from assuming an ordinal relationship between categories.

2. *Scaling Numerical Features:*
- The numerical features - *Age, Flight Distance, Departure Delay in Minutes,* and *Arrival Delay in Minutes*; were standardized using *StandardScaler*.

Standardization ensured that all numerical features had a mean of 0 and a standard deviation of 1, preventing features with larger ranges from dominating the model.

3. *Combining and Normalizing Features:*
- After preprocessing, the binary and non-binary categorical features were combined.
- MinMax scaling was applied to normalize these categorical features.

This results in 2 data frames separated by their data type- numerical and categorical: for the purpose of preprocessing methods.

4. *Dimensionality Reduction Using Chi-Square Selection:*
- One-hot encoding increased the number of features by 69 additional columns.
- To reduce dimensionality, the Chi-Square feature selection method was applied. Features with a score below 1000 were removed, reducing the categorical feature set by approximately 39%.

Reducing dimensionality speeds up computation time and eliminates less relevant features, preventing the model from being overwhelmed by unnecessary data, enhancing model efficiency.

5. *Final Dataset Preparation:*
- The selected categorical features were merged with the numerical features, completing the dataset preparation for model training and testing.

IV. Model Training

The dataset originally contained 103,904 samples and 56 features before training. The data was split into 70% training and 30% testing because an 80/20 split led to longer training times and slight overfitting in some models.
The models were trained on Google Colab's cloud GPU's and CPU' and on a personal Mac M1 Pro chip on heavier models. The training time for each model is reported in the results section.

The following versions of third-party libraries were used:
- Scikit-learn: v1.2.2
- XGBoost: v1.7.6
- Pandas: v1.5.3
- NumPy: v1.23.5

To evaluate model performance, the following metrics were used:
- Accuracy
- ROC Curve and AUC Score
- Precision, Recall, and F1-Score
- Confusion Matrix
- Learning Curve graphs

1. Logistic Regression

Logistic Regression was chosen as a baseline classifier to determine linear relationships and because of its efficiency and interpretability. Hyperparameters were tuned using Gridsearch as follows:
- C = 10 (regularization strength)
- class_weight = 'balanced' (to address class imbalance initially shown in Confusion Matrix) (Figure 3)

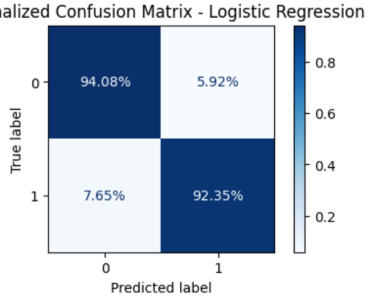

Figure 3: False positives were at a 10.3% to 4.9% ratio before setting class_wight = 'balanced'

This classifier ran quickly and completed in 7.36 seconds.

2. K-Nearest Neighbors (KNN)

KNN was selected to try testing a non-parametric model, attempting to possibly capture more complex relationships in the data. The following parameters were optimized by GridSearch:
- n_neighbors = 11
- jobs = -1 (use all CPU cores for efficiency)

This classifier ran the slowest, completing in 190 seconds.

3. Decision Tree Classifier

Decision Trees were implemented to test Tree models and analyze hierarchical relationships between features. GridSearch determined the best parameters as follows:
- max_depth = 20 (maximum tree depth)
- min_samples_leaf = 10
- min_samples_split = 10

This classifier ran the fastest, completing in 1.50 seconds.

4. Random Forest Classifier

Given the strong performance of Decision Trees, Random Forest was employed as an ensemble learning approach to reduce overfitting. The parameter was chosen by hand through trial and error:

- n_estimators = 100 (# of decision trees in the forest)

This classifier ran decently and completed in 52.4 seconds.

5. XGBoost Classifier

XGBoost was chosen for its ability to efficiently handle complex feature interactions. The final hyperparameters were found by GridSearch:

- n_estimators = 500 (# of boosting rounds)
- learning_rate = 0.05 (step size shrinkage)
- eval_metric = "logloss"
- max_depth = 7

This classifier also ran decently and completed in 47.1 seconds.

V. Discussion

A. Performance Analysis

Tree-based models- Decision Trees, Random Forest, and XGBoost; significantly outperformed traditional methods like Logistic Regression and KNN. This suggests that passenger satisfaction is influenced by non-linear and complex feature interactions**,** which tree-based models can capture effectively.

1. Logistic Regression: Served as a strong baseline but struggled with complex relationships in the data (Figure 4)

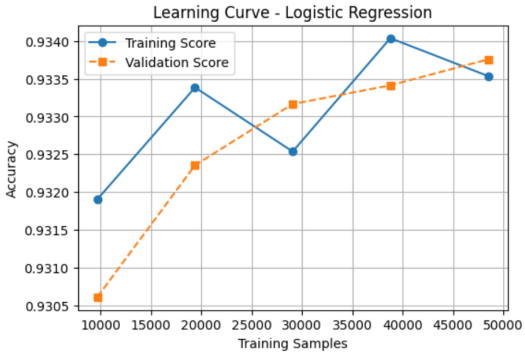

Figure 4: Learning curve of Log. Reg., showing volatile training score changes

2. KNN: Achieved similar performance to Logistic Regression but was very computationally expensive due to its distance calculations.
3. Decision Trees: Performed well while maintaining efficiency, showing steady improvements in both training and test scores.
4. Random Forest: Demonstrated robustness and generalization power but required more training time.
5. XGBoost: Achieved the best results, balancing accuracy, generalization, and efficiency

B. Overfitting and Learning Curve Observations

1. Random Forest showed training accuracies over 99%, but its test accuracies were 4-5% lower. This suggests that more training data could improve performance (Figure 5).

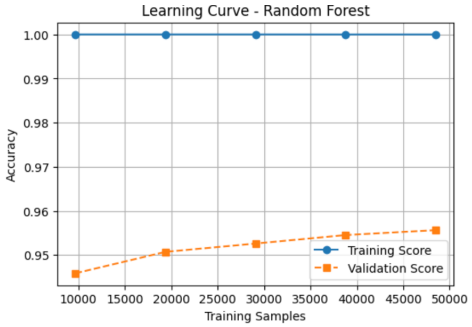

Figure 5: Training Curve of Random Forest shows effective relationship comprehension

2. Decision Trees showed steady learning without significant overfitting, making them a reliable mid-tier choice.
3. KNN continued to improve with more data, but its computational cost made it impractical for large-scale datasets.

C. Future Improvements

To further improve classification performance, future work could include:

1. More Hyperparameter Tuning: Further refining parameters for tree-based models.
2. Feature Selection Optimization: Investigating different selection methods beyond Chi-Square.
3. Early Stopping in XGBoost: To prevent unnecessary training beyond optimal performance.
4. Exploring Deep Learning Approaches: Applying neural networks to see if they outperform tree-based methods.

VI. Conclusion

This study explored multiple machine learning models to predict airline passenger satisfaction, evaluating their performance using accuracy, AUC, precision, recall, and F1-score.

Among all models, XGBoost achieved the highest performance (95.92% accuracy, 0.9940 AUC), demonstrating strong predictive power**.** Random Forest followed closely, offering slightly lower accuracy but high generalization ability.

Decision Trees provided a balance between efficiency and accuracy, while KNN and Logistic Regression struggled a little more to capture complex relationships in the data. (Figure 6)

Final Model Performance Summary:

	Model	Accuracy	AUC	Precision	Recall	F1-Score
0	Logistic Regression	0.9333	0.9777	0.9224	0.9235	0.9230
1	K-Nearest Neighbors	0.9301	0.9788	0.9364	0.8995	0.9176
2	Decision Tree	0.9483	0.9823	0.9506	0.9287	0.9395
3	Random Forest	0.9568	0.9918	0.9647	0.9343	0.9493
4	XG Boost	0.9595	0.9941	0.9676	0.9378	0.9525

Figure 6: Final performance results of each model

Overall, ensemble methods like Random Forest and XGBoost proved to be the most effective for this classification task. Future improvements, including further hyperparameter tuning, feature selection refinement, and deep learning exploration, could further enhance model accuracy and efficiency.

REFERENCES

[1] T. Mahal, "Airline Passenger Satisfaction Dataset," Kaggle, 2021. Available: https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction/data.

[2] Scikit-learn Developers, "Model selection: choosing estimators and tuning hyperparameters," *Scikit-learn*, 2024. Available: https://scikit-learn.org/stable/model_selection.html.

[3] OpenAI, *ChatGPT*, "AI-based grammar assistance and quick technical queries," OpenAI, 2025. Available: https://openai.com/chatgpt.

[4] A. Sanmartín, *MADR 4511W: Introduction to Artificial Intelligence*. Course materials and example code, University of Minnesota/ FOM, 2025.