Predicting Airline Passenger Satisfaction Using Machine Learning Techniques

Andreea Diana Gherghescu
Digital Transformation Management
University of Bologna
Email: andreea.gherghescu@studio.unibo.it
Matricola: 0001120955

Xiuyue Zhang
Digital Transformation Management
University of Bologna
Email: xiuyue.zhang@studio.unibo.it
Matricola: 0001113673

***Abstract*—One of the main goals of airline companies is to retain existing customers and attract new ones by continuously improving the quality of their services. Passenger satisfaction depends on numerous factors, from the airport experience to in-flight services, punctuality, and the off boarding process.**

This project focuses on classifying airline passengers as either satisfied or unsatisfied based on various aspects of their journey. The analysis is based on a survey dataset that includes features such as travel class, type of travel, flight delays, onboard services, and demographic information.

The workflow involves preprocessing of the dataset, data mining for gathering insights and understanding the context, and modeling and comparing various classifiers to identify the most capable to distinguish between satisfied and unsatisfied passengers.

1. Introduction

Airline companies aim to enhance the customer experience to remain competitive and drive profitability. With customer expectations rising and travel options expanding, satisfaction has become a key performance indicator, influencing brand loyalty and future choices. Satisfaction is influenced by multiple touch points throughout the journey — from airport arrival and check-in procedures to in-flight comfort and the overall quality of service provided. This machine learning project proposes to build a reliable classification model that can predict whether a passenger is satisfied or unsatisfied, based on various features captured from a post-flight survey. The dataset includes responses from thousands of airline passengers, with information such as type of travel, travel class, flight delays, service ratings, and demographic attributes.

The project consists of four main stages, further summarized.

1.1. Data Loading

The first stage involves loading the Airline Passenger Satisfaction dataset from Kaggle. It contains 103904 answers to a satisfaction survey comprising 24 features, plus the label column which categorizes the passengers' experience as *Satisfaction* or *neutral or dissatisfaction*.

1.2. Data Profiling

After importing the dataset, we need to infer some information about its structure and the context of the data. This includes visualizing the number of missing values, the distribution of data, and the semantics of the features.

1.3. Data Preprocessing

At this stage, it is necessary to explore the data to extract and keep only the relevant features for modeling, and make sure their values are standardized to match the models. It includes visualizing the data through plots, dropping irrelevant features, checking correlations between features, encoding categorical values, standardizing the data, and splitting it into a training and a testing set.

1.4. Modeling and Evaluation

Machine Learning models are leveraged to create the most effective classifier for the given purpose. Various models are trained and tested (Decision Tree, Random Forest Classifier, K-Nearest Neighbors Classifier, LGBM Classifier, and a deep learning model, specifically the Multiple-Layer Perceptron Classifier). Techniques as feature extraction or hyperparameter tuning further improve and support the chosen models, which are evaluated in terms of accuracy.

The ultimate goal is to develop a robust model capable of predicting satisfaction labels with high accuracy, potentially helping airlines better understand customer pain points and tailor services accordingly. The findings from this project not only contribute to the technical understanding of classification in structured datasets but also provide actionable insights for customer retention and service enhancement in the airline industry.

1.5. Group Organization

The project was completed as a collaborative effort, with each of the members contributing to different parts of the workflow according to their individual strengths and interests. The process started with both members working together to select the project topic. After discussing several ideas, a consensus was reached to focus the analysis on predicting passenger flight satisfaction.

After choosing the topic, Andreea Gherghescu took the initiative to find a suitable dataset. She researched various sources and successfully identified and obtained the airline passenger satisfaction dataset, which became the foundation of our work.

Once having the dataset, Xiuyue Zhang was responsible for setting up the project environment. This included importing the necessary libraries, loading the data, and ensuring everything was ready for analysis. Then, Xiuyue led the data understanding phase, exploring the structure of the dataset, examining the characteristics, and gaining a complete understanding of the content and context of the data.

Xiuyue also handled the analysis of the data distribution, using descriptive statistics and visualizations to better understand how the data was spread across different features. Following this, she managed all aspects of data preprocessing. This involved cleaning the data, handling missing values, encoding categorical variables, and normalizing numerical features. In addition, Xiuyue also did the task of splitting the data into training and test sets to prepare them for modeling.

After the data was prepared, Andreea took over the remaining phases of the project. She was responsible for implementing and evaluating various machine learning models, including Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Classifier, and Neural Network. Andreea also performed feature selection, hyperparameter tuning, and compared the performance of different models to determine the most effective approach.

Throughout the project, the team maintained regular communication and coordinated their efforts to ensure a smooth workflow. This clear division of responsibilities allowed the members to efficiently complete each stage of the project while supporting each other and maintaining a high standard of work.

Andreea Gherghescu, Xiuyue Zhang
April 30, 2025

2. Related Work

Customer satisfaction in the services industry has been an active research area, especially with the growing availability of customer feedback. Airline passenger satisfaction is no exception, with multiple related datasets available online. Various machine learning models have been applied to predict satisfaction levels using structured data collected from surveys or transactional records.

In [1], the author used multiple classification models, such as Decision Trees, Random Forest, and Support Vector Machine, to predict whether passengers were satisfied or unsatisfied based on survey data. The study is aimed at the interpretability of machine learning models to identify the features that drive customer satisfaction. Among the models tested, the Random Forest Classifier proved to be the best, obtaining high accuracy and having the option to provide interpretable results on feature importance. The study highlights Random Forest's high performance on high-dimensional data and its suitability for real-world classification tasks in domains such as airline passenger satisfaction. The model's ability to handle both categorical and continuous features efficiently makes it a top choice for similar tasks.

Similarly, in [2], a comparative study was conducted on a set of classifiers, including Logistic Regression, k-Nearest Neighbors, and Gradient Boosting. The study concluded that ensemble models, more specifically Gradient Boosting, significantly outperformed individual classifiers due to their ability to handle feature interactions and class imbalance. Gradient Boosting's ability to combine weak learners into a strong predictor by iteratively correcting mistakes from previous models in the sequence enabled it to pick up on complex patterns in the data. Furthermore, it is observed in the paper that while Logistic Regression and k-NN were well-performing for less complex cases, ensemble models were more consistent in handling complexities in the given challenge, namely customer satisfaction.

Another notable approach, explored in [3], was specifically focused on the performance optimization of k-NN through ensemble techniques. Although k-NN is a straightforward algorithm based on the computation of point-to-point distances, its performance is inconsistent when applied to noisy or high-dimensional data. The study addressed these issues by leveraging bagging, which builds multiple models and aggregates their predictions to reduce variance and model instability. The results demonstrated that, while k-NN alone provided moderate performance, its ensemble version with bagging significantly enhanced accuracy and resulted in a more stable prediction, reflecting the benefits of combining multiple simple classifiers to boost overall model performance in airline customer satisfaction prediction.

These studies demonstrate the effectiveness of combining feature analysis with robust classifiers in building predictive systems for passenger satisfaction and driving customer-centric strategies, forming the foundation upon which this project is based.

3. Proposed Method

3.1. Data analysis

The first step was to understand the semantics of the 24 features given, which are also described in the Data Card[1] of the dataset.

1. https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction/data

When profiling the data, we first found two columns that were not relevant for the process (*Unnamed: 0* and *id*) and dropped them. Then, we started to analyze the categorical features, their number of unique values, and their distribution. The label class was more or less balanced, while others, like *Class* or *Customer Type*, were not. Some conclusions were that the majority of customers are loyal and travel for business purposes, and the *Eco Plus* class is likely not going to contribute a lot of information to our purpose since it represents less than 10% of class choices.

Further, we checked the correlations between features using a correlation matrix. Following this step, we dropped the *Arrival Delay in Minutes* feature as it was highly correlated with the departure delay and had some missing values. We also observed other correlations, some being logical from a semantics point of view, but we decided to keep all the features for the moment, as the correlation values were not as high.

Plotting the distribution of some numerical features, we gained some insights about potential relevant features, but not enough to take any action at this step.

One of the attributes that did not seem to be correlated with any other attribute but usually expresses some degree of information is *Age*, so we decided to take a closer look at it. We plotted the binned distribution, which was normal, and continued to plot two boxplots of *Age* combined with *Customer Type* and *Class* (see Fig. 1). Considering that more than 90% of passengers chose to fly in *Business* or *Eco* class, we could see from the box plots that the age box of *Business Class* was almost the same as that of *Loyal Customers*, while the *Eco Class* included younger customers and had a lower age average. So, we concluded that the *Loyal Customers* usually fly in *Business Class*, regardless of age. Some of the models we were going to use do not

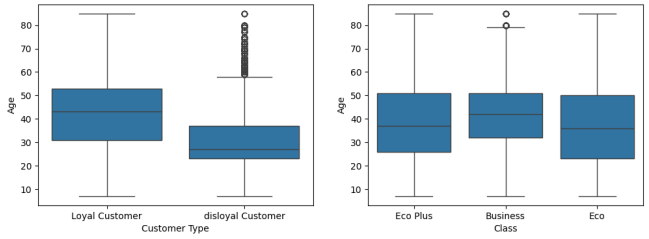

Figure 1. Boxplots showing the distribution of passenger age grouped by (a) Customer Type and (b) Travel Class.

work well with categorical features, so we needed to do some encoding. Attributes with only two unique values were transformed into boolean values. The *Class* attribute had 3 features, so we used one-hot encoding. The last step of data preprocessing was to normalize the data, for which we used the `StandardScaler`, and to split it into training and testing sets, with a distribution of 80% to 20%.

Another attempt to explore the data distribution involved visualizing it in 2D and 3D using Principal Component Analysis (PCA). However, due to the large size and complexity of the dataset, these visualizations did not lead to meaningful conclusions, as shown in Fig. 2.

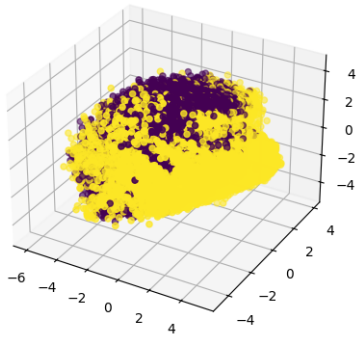

Figure 2. Visualizing the data in 3D using Principal Component Analysis.

3.2. Classification

After analyzing the data, we focused on training, testing, and improving multiple machine learning models.

For hyperparameter tuning, i.e., finding the best-performing model configuration, we used `GridSearchCV` and `RandomizedSearchCV`. **`GridSearchCV`** exhaustively searches through a specified grid of hyperparameters by evaluating all possible combinations using cross-validation. It ensures thoroughness but can be computationally expensive, especially with large grids. **`RandomizedSearchCV`**, on the other hand, selects a fixed number of parameter settings from the grid at random. While it does not guarantee the absolute best combination, it is significantly faster and often finds a near-optimal solution with much lower computational cost, making it a practical choice for large or complex search spaces.

The libraries used for all models and subsequent techniques are `scikit-learn` and `lightgbm` for the LightGBM Classifier.

3.2.1. Logistic Regression. Logistic Regression is a widely used statistical method for binary classification tasks that estimates the probability that a given input belongs to a particular class. It uses the logistic (sigmoid) function to map predicted values to probabilities between 0 and 1. This makes it particularly suitable for problems where the goal is to distinguish between two categories.

3.2.2. K-Nearest Neighbors Classifier. KNN Classifier is a simple algorithm used for classification and regression tasks. In classification, it assigns a label to a data point based on the majority label of its k closest neighbors in the feature space. KNN is non-parametric, meaning it makes no assumptions about the underlying data distribution, and its performance heavily depends on the choice of k and the distance metric used.

To get a larger coverage over the space of the k parameter, we used a randomized search, which concluded that the best configuration for k is 9.

3.2.3. Decision Tree. A Decision Tree models decisions and their possible consequences as a tree-like structure, where internal nodes represent feature-based conditions, branches represent their outcomes, and leaf nodes represent class labels. Entropy was used as `criterion` to measure impurity.

3.2.4. Random Forest Classifier. The Random Forest Classifier is an ensemble learning method that builds multiple Decision Trees and combines their outputs to make more accurate and stable predictions. Each tree is trained on a random subset of the data and considers a random subset of features when splitting nodes, which helps reduce overfitting and improves generalization. The final prediction is made by majority voting among the trees.

Randomized search was again used for finding the best configuration with the following parameters:

- `n_estimators`: The number of trees to be constructed, which had the values [50, 100, 150].
- `max_depth`: The maximum depth of each tree, used for controlling the complexity of the model and reducing overfitting. We set it to [5, 10, 15].
- `min_samples_split`: The minimum number of samples required to split an internal node in a decision tree, also used for controlling overfitting. Set to [2, 5].
- `min_samples_leaf`: The minimum number of samples required to be at a leaf node, improves generalization. Set to [1, 2].

The feature importance outputted by this model showed a variation among features, from the most important one with a score of 0.20, to the least important one having a score below 0.01. This gave us the intuition that it may be better to drop some features that are not important and may produce noise.

For this, we used **Recursive Feature Elimination with Cross-Validation**, a feature selection technique used to identify the most important features in a dataset. It works by recursively removing features and evaluating the model's performance using cross-validation at each step. The goal is to reduce the number of features while maintaining or improving the model's performance. After obtaining the new set of features, we proceeded to retrain the forest with the best parameters, as well as search through a larger grid.

3.2.5. Support Vector Classifier. SVC works by finding the hyperplane that best separates the data into different classes with the largest margin, being effective in high-dimensional spaces. A randomized search was again applied with the following parameters:

- `C`: Regularization parameter in charge of the bias-variance trade-off, set to [0.1, 1, 10].
- `kernel`: Set to ['linear', 'rbf'], with the first one allowing a linear decision boundary, while the second one can capture more complex relationships.
- `gamma`: Defines the influence of a single training example, where 'scale' sets it as the inverse of the number of features.

After these attempts, we decided to use **AutoML**, which predicted the LightGBM Classifier as the best for our purpose. The metric used was accuracy.

3.2.6. Neural Networks. After trying a few machine learning models, we tried a deep learning model, namely the **Multi-Layer Perceptron Classifier**. It consists of one or more hidden layers and learns complex patterns by adjusting weights through backpropagation. MLP Classifier is well-suited for classification problems where relationships between features are non-linear. The parameters used were:

- `hidden_layer_sizes`: Specifies the number of neurons in each hidden layer, with the values (128, 64, 32) constructing a three-layer architecture.
- `activation`: Defines the activation function to be used in the hidden layers, with "relu" (Rectified Linear Unit) being chosen for introducing non-linearity.
- `solver`: The optimization algorithm used for training the network, with "adam" (Adaptive Moment Estimation) being an efficient and widely-used solver.
- `alpha`: A regularization parameter to prevent overfitting, with 0.01 indicating a moderate level of regularization.
- `learning_rate`: Controls how much the model weights are updated during training, with "adaptive" meaning the learning rate adjusts based on performance.
- `max_iter`: The maximum number of iterations (500) for training the model, controlling how long the training process will run.

Hyperparameter tuning was further conducted, enlarging the search space.

3.2.7. Light Gradient Boosting Machine Classifier. LGBMC is a fast, efficient gradient boosting framework based on decision trees. Given that it was the model suggested by AutoML, we tried improving it through a grid search.

4. Results

In this section, we present the results obtained with each implemented model.

4.1. Logistic Regression

The Logistic Regressor is a quite rigid model, trying to find a decision boundary for the dataset. Given that our data is very complex, with over 20 features, it did not perform as well, obtaining an accuracy of 0,87 on the test set. It ended up as the worst-performing of our models.

4.2. K-Nearest Neighbors Classifier

The KNN rendered almost the same accuracy on the test set for different configurations of the k parameter, with a value of 0.92. Our intuition is that it would take a much larger k value to accurately separate the data.

4.3. Decision Tree

The Decision Tree performed better than expected, with an accuracy value of 0.94, outperforming both the Regressor and the KNN. Still, further exploring Random Forests did not incentivize us to continue exploring the Decision Tree.

4.4. Random Forest Classifier

The Random Forest had an overall good performance, better than all the models tried before. The initial grid already provided 0.959 accuracy, with 150 estimators and a maximum depth of 15. The RFECV further selected an optimal number of 18 features for the best forest, which then obtained a score of 0.961. Looking at the importance scores, we could observe that some of the features that were not selected, like the *Eco Plus* class, confirm our earlier intuitions that they are not correlated to the satisfaction levels.

Encouraged by this significant improvement, we continued with a second grid with the 18 features, which scored 0.962, having 150 trees and a maximum depth of 25.

In the second grid, the maximum number of estimators we set was 200, and the maximum depth was 25, leading us to assume that a more complex forest with a bigger depth would have produced even better results. Unfortunately, our computing capabilities and time were limited, so we could not follow this lead. Given all this, Random Forest still provided satisfactory results.

4.5. Support Vector Classifier

The SVC randomized search obtained a 0.958 accuracy score, which is better than how the Decision Tree performed, but it is worse than all the Random Forests we tried. Also, the computation time was much higher than for any other model, so we decided not to dig more into this model.

4.6. Multi-Layer Perceptron Classifier

The MLP's first configuration for the hidden layers was (128, 64, 32), with an underwhelming accuracy score of 0.951. Following the grid search, the same activation function `relu` was kept, but the hidden layers' structure changed to (256, 128, 64, 32). This configuration resulted in a big jump in the accuracy score to 0.958. Again, it is a computationally heavy model, but we do believe that a more complex structure of the layers would have learned the data patterns better and would have resulted in a higher accuracy.

Still, given that limitations exist, for this particular context, the Random Forest performed better.

4.7. Light Gradient Boosting Machine Classifier

The LGBM chosen by AutoML had an accuracy of 0.964, which is from the start better than all our tries. Given that, we wanted to see if by providing more estimators, bigger depth, and various values to the parameters, we would obtain a better performance. After the hyperparameter tuning, we remained at the accuracy of 0.964. These two configurations obtained from the MLP Classifier remain our best solutions.

5. Conclusions

In conclusion, this machine learning project aimed at building and providing robust classifiers for distinguishing between satisfied and unsatisfied passengers of airline companies. Through data profiling, exploration, and modeling, we implemented, tested, and improved various models, successfully answering the given challenge.

The machine learning methods we used involve Linear Regression, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Classifier, LightGBM Classifier, and Multi-Layer Perceptron, as well as the use of different methods for hyperparameter optimization, feature extraction, and automated modeling.

In particular, the LightGBM model exhibited impressive results, closely followed by the Random Forest. They showed a good understanding of the dataset and effective capabilities to classify passenger satisfaction.

References

[1] T. Chavan, "Predicting and Optimizing Airlines Customer Satisfaction Using Machine Learning," *RIT Scholar Works*, 2022. Available: https://repository.rit.edu/cgi/viewcontent.cgi?article=12513&context=theses.

[2] A. Mishra and R. Sharma, "Airline Customer Satisfaction Prediction," *International Journal for Multidisciplinary Research*, 2024. Available: https://www.ijfmr.com/papers/2024/2/18640.pdf.

[3] R. A. Ibrahim and R. Rao, "Predicting Airline Customer Satisfaction using k-NN Ensemble Regression Models," *International Journal of Scientific and Technology Research*, vol. 8, no. 10, pp. 1356–1360, 2019. Available: https://www.researchgate.net/publication/336603226.

[4] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.