Project Report

Data Mining Report Group 4

Benedetta Fiorillo, Chien-Yun Lin, Javier Miranda,
Jieun Park, Tatiana Nachev, Yujeong Kim

May 18, 2025

Submitted to
Data Mining Teaching Team
Prof. Dr. Sven Hertling
University of Mannheim

1 Introduction

Predicting flight prices has become increasingly important in the aviation industry of today, where the demand of flights has increased through the years, only in 2022, according to United States Department of transportation, 853 million passengers took a domestic flight (Bureau of Transportation Statistics 2023). Accurate price prediction is crucial not only for airlines aiming to optimize their revenue management but also for travelers seeking the best possible deals (Korkmaz 2024).

Estimating flight prices is a complex process shaped by a combination of economic, operational, and market-driven factors. Airlines set fares based on variables such as demand and supply, route popularity, seasonality, competition, fuel costs, and operational expenses (Wang et al. 2019). For example, prices often rise during peak travel seasons or on popular routes. Additionally, factors like the type of aircraft, baggage options, seat selection, and cabin class further influence ticket costs (Skyscanner 2025).

This report focuses on predicting domestic U.S. flight prices using a real dataset of flight information (dilwong 2021). We apply various data mining techniques and predictive models to determine which approach most accurately forecasts ticket prices. Our goal is to identify the most effective model, providing valuable insights for the airline industry and demonstrating how deeper analytical approaches can improve pricing strategies and decision-making.

2 Profile of Flight Price Datasets

This analysis is based on the publicly available `Flight Prices` dataset (dilwong 2021), which is provided in CSV format and sourced from the online travel agency Expedia. The dataset contains detailed records of flight ticket searches conducted between April 16th and October 5th, 2022, covering a period of nearly six months. It includes data on flights between several major airports across the United States. The dataset will be downloaded from Kaggle and processed in a Python-based analytical environment. It includes a wide range of variables such as departure and arrival airports, travel duration, ticket prices, seat availability, and information related to the airline and aircraft, offering a rich foundation for comprehensive analysis.

The objective of this study is to predict the base fare of each flight search, which will serve as the target variable. The dataset comprises a total of 82,138,753 rows, representing flight information from 16 U.S. airports over the specified time frame. Due to the substantial size of the dataset, the analysis will be narrowed to focus exclusively on flights departing from a single airport. Los Angeles International Airport (LAX) has been selected for this purpose, contributing 8,073,281 rows, approximately 9.83% of the total dataset.

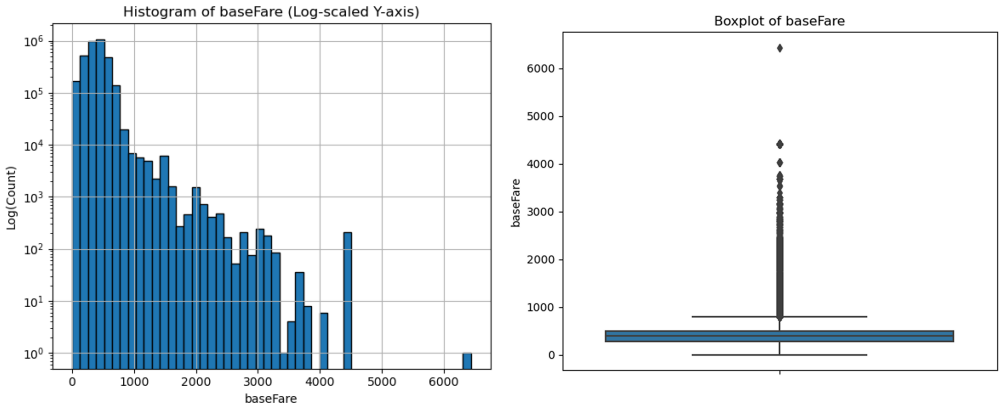
2.1 Target Variable Analysis: Base Fare

We began by examining the distribution of the target variable, `baseFare`, in the training data using both histogram and boxplot visualizations, along with descriptive statistics.

The summary statistics revealed a mean of approximately $398 and a median of $393, with a standard deviation of about $186, indicating a wide spread in pricing. Most values were concentrated between $200 and $600, but the distribution showed strong positive skewness of 3.04 and a high kurtosis value of 35.26, suggesting the presence of extremely high-end values.

Figure 1a shows the distribution of `baseFare` in histogram. We applied the logarithmic scale to the y-axis of the histogram to visualize these rare, high-end values more clearly. It reveals a highly right-skewed distribution, where the majority of `baseFare` values are concentrated below $600, with a steep drop-off in frequency as prices increase. However, a substantial number of low-frequency bins exist in the higher fare ranges, extending beyond $4000.

This distributional pattern is further confirmed by the boxplot of `baseFare` as shown in Figure 1b, which shows a relatively compact interquartile range (from $289.30 to $493.03), yet a large number of extreme values far beyond the upper whisker. The numerous outliers reflected as individual points above the box emphasize the heavy-tailed nature of the distribution. While these fares are statistical outliers, they may reflect valid cases such as last-minute bookings, premium travel classes, or international routes. Therefore, we chose to retain them in the dataset rather than remove them.



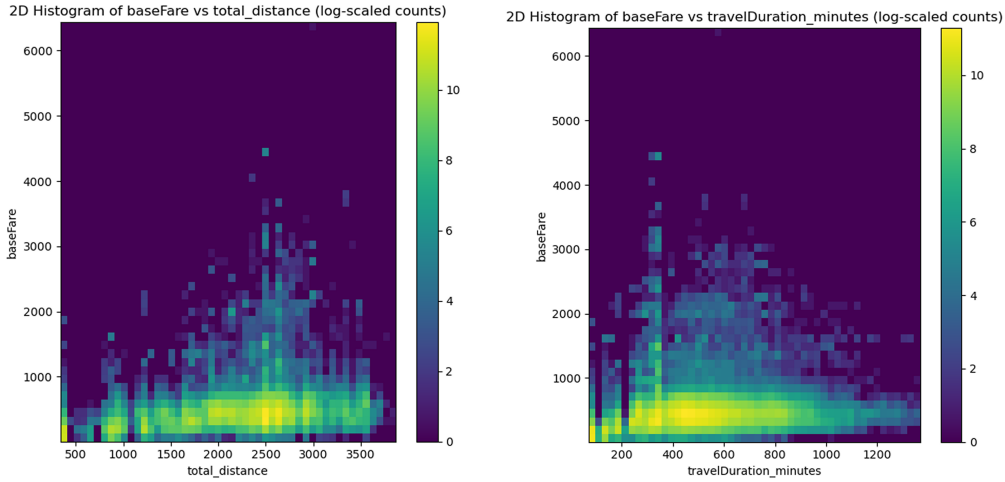
(a) Histogram (b) Boxplot

Figure 1: Distribution of `baseFare` in (a) histogram and in (b) boxplot (right).

2.2 2D Histogram Analysis

To further examine potential relational outliers and the underlying structure of the data, we visualized `baseFare` against two continuous predictors—`total_distance` and `travelDuration_minutes`—using 2D histograms as shown in Figure 2. To enhance interpretability, the bin counts were transformed using a logarithmic scale via log(1 + count).

In both plots, the majority of data points are concentrated in the lower fare ranges (approximately below $600), and correspond to moderate travel distances and durations. As expected, `baseFare` generally increases with both `total_distance` and `travelDuration_minutes`, although the relationships are not strictly linear and contain considerable variability. Several sparse, high-fare data points are visible—particularly above $3000—which deviate from the general trend. These points were not removed as outliers, as they may represent real-world pricing anomalies—such as last-minute bookings or dynamic pricing conditions—and could carry valuable information for the predictive model.



(a) `total_distance` (b) `travelDuration_minutes`

Figure 2: 2D histogram of `baseFare` vs. (a) `total_distance` and (b) `travelDuration_minutes`

2.3 Correlation Matrix

To explore the interrelationships among numerical features, a correlation matrix was utilized as shown in Figure 3. Since many features exhibit weak or negligible correlations in the full matrix, only those relevant to the target variable `baseFare` were included in the visualization to improve clarity.

This visualization allows for a preliminary identification of features that may exert a strong influence on the target. For instance, `total_distance` and `totalFlightTime` are highly correlated, with a coefficient of **0.96**, indicating that longer distances are generally associated with longer flight durations. Similarly, `is_holiday` and `isAroundHoliday`, both are related to holiday periods, show a moderate correlation of **0.38**.

In terms of flight structure, the number of legs (`num_legs`) is positively associated with `totalDuration_minutes` (**0.75**), `totalTransferTIme` (**0.65**) and `totalFlightTIme` (**0.59**), reflecting that multi-leg itineraries tend to be longer and involve more transfers. Conversely, `isNonStop` displays a strong negative correlation with `num_legs` (**-0.91**) and `travelDuration_minutes` (**-0.71**), suggesting that direct flights typically have no legs

and shorter travel times.

Regarding the target feature `baseFare`, several features appear to be influential. Both `travel_distance` and `totalFlightTime` demonstrate moderate positive correlations with `baseFare`, with coefficients of **0.44** and **0.43**, respectively, implying that longer routes and extended flight durations tend to increase base fare. On the other hand, `isCoach` (representing basic cabin class) is negatively correlated (**-0.38**), indicating that such tickets are generally more affordable.

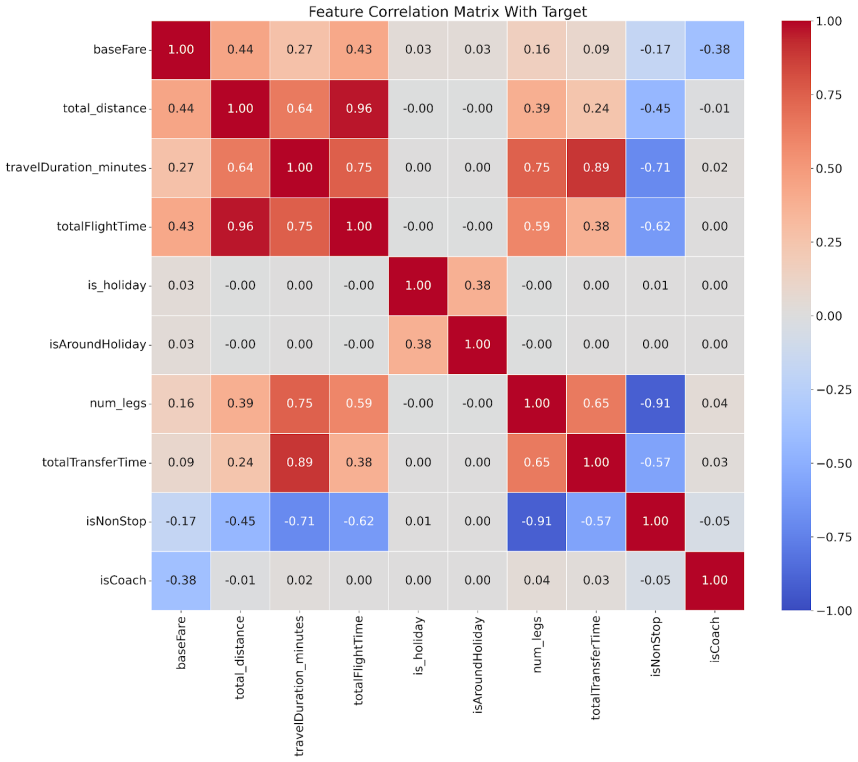


Figure 3: Correlation matrix of the Flight Price Dataset

3 Preprocessing

As mentioned before, the analysis focuses only on flights departing from one airport. Los Angeles International Airport (LAX) was chosen for this. To make the analysis more manageable, the dataset was reduced to include only flights from April to August. This smaller set was then divided into two parts: a training set with data from April to July (3,400,829 rows), and a test set with data from August (1,742,260 rows). This time-based split makes it possible to build models using data from the earlier months and test how well they perform on new, unseen data. Using August as the test month creates a more realistic forecasting situation, where the model is applied to future events. Keeping the order of the months also helps avoid data leakage and makes sure the results reflect how the model would work in real-world scenarios.

3.1 Data Cleaning

The cleaning phase started by removing rows with missing values. Beyond standard null entries, we also excluded structurally missing data, such as fields containing only delimiters (e.g., `||`) or placeholder strings like `None`. Additionally, rows with invalid values were discarded, particularly when numerical features related to time and distance contained negative or non-numeric entries. For multi-segment fields—those with values separated by `||`—each segment was validated, and if any segment was found to be invalid, the entire row was removed.

Next, we removed several features that were either redundant or not relevant to the prediction task. For example, columns such as `legId` and raw timestamp fields like `TimeEpochSeconds` were excluded because they contained information already represented in processed variables. Additionally, we dropped the `totalFare` column since it closely correlates with the target variable `baseFare`, differing only by tax amounts. Including `totalFare` could lead the model to simply memorize this value rather than learn meaningful patterns. Alongside feature removal, we also eliminated fully duplicated rows—those where all feature values were identical—to reduce redundancy and avoid biasing the model with repeated data points.

Overall, this cleaning process led to the removal of 710,940 rows from the training set (20.9%) and 358,847 rows from the test set (20.6%). To evaluate the potential for bias, we compared the distributions of `flightMonth` and `destinationAirport` before and after the removals. The results showed minimal changes, suggesting that the data removed was not concentrated in any specific period or destination. Thus, the overall representativeness of the dataset was preserved.

3.2 Feature Conversion

Following the data cleaning, we transformed several features to formats more suitable for modeling. Destination airport codes were one-hot encoded, resulting in 15 binary columns—one for each destination. Although we considered label encoding to reduce dimensionality, in the end, we chose one-hot encoding to maintain interpretability, as it allows us to assess the individual impact of each airport on flight pricing.

Travel durations were originally stored in ISO 8601 format (e.g., `PT8H5M`). We converted these into total minutes (e.g., 485 minutes), creating a uniform numerical representation that simplifies modeling and comparison. Similarly, both the search and flight dates were transformed from their original string formats (e.g., `yyyy-MM-dd`) into sequential integers. The earliest date was assigned the value 1, with subsequent dates incremented by 1. This conversion preserved chronological order and avoided the complications associated with raw date formats.

3.3 Feature Engineering

To enrich the dataset with contextual and temporal information, we engineered a variety of new features. Two binary variables, `isholiday` and `isAroundHoliday`, were introduced to capture the impact of public holidays. The first indicates whether the flight

date coincides with a U.S. public holiday in 2022, while the second flags flights occurring within three days before or after a holiday. These features aim to reflect holiday-driven fluctuations in pricing.

Several temporal indicators were also created to capture time-related patterns. These include `dayOfWeekNum` and `SearchDayOfWeekNum`, which represent the weekdays of the flight and the search, respectively (coded from 0 for Monday to 6 for Sunday), as well as `dayNum` and `monthNum`, which encode the day of the month and the month itself. All of these values were cast to the `int64` data type for consistency.

To incorporate time-of-day effects, we extracted the hour from both the departure and arrival times, creating the features `departureHour` and `arrivalHour`. We also calculated the number of days between the search and flight dates, stored as the `daysUntilFlight` feature, which can capture booking behavior and urgency, both of which are important factors in pricing.

Additional structural features were derived from the segmented nature of flight itineraries. The `num_legs` feature represents the number of flight segments by counting entries in the `segmentsArrivalTimeEpochSeconds` field. For rows with missing values, we assigned a default of zero. We also computed the `totalTransferTime` by summing the layover durations between segments, with direct flights assigned a value of zero. Subtracting this from the overall travel duration gave us the `totalFlightTime`, a measure of in-air time that better captures flight efficiency.

We also replaced the inconsistent `totalTravelDistance` field with a more accurate custom feature, `totalDistance`, which was calculated by summing the distances of individual flight segments. A comparison showed discrepancies between the two measures, validating our choice.

To explore aircraft influence, we computed three features—`relBoeing`, `relAirbus`, and `relOthers`—that represent the proportion of total flight distance covered by Boeing, Airbus, or other aircraft types. These were calculated by summing segment distances for each aircraft type and normalizing by the total trip distance.

Finally, we extracted airline information from multi-leg fields and computed frequency-weighted representations. Each airline involved in a trip was identified, and its occurrences were weighted by the corresponding segment distance. This led to the creation of ten new columns, each representing the weighted involvement of a specific airline. This approach balances frequency with impact, capturing both presence and relevance.

Together, these preprocessing steps resulted in a refined and enriched dataset comprising 49 well-defined features, 2,689,889 entries for the training set and 1,383,413 entries for the test set.

4 Data Mining

As previously mentioned, we sorted the dataset chronologically and used the data from August exclusively for testing. Before evaluating any model, we repeated this splitting approach and further divided the remaining data into 80% for training and 20% for validation using `train_test_split` with `shuffle=False`. This preserved the chronological

order and ensured consistency with the test set structure across all models, allowing for robust hyperparameter tuning and fair model comparison.

To ensure comparability across models, we standardized numerical features by normalizing the dataset—but only for models that required it. Tree-based models such as Decision Trees and Random Forests are insensitive to feature scaling, so we left their inputs unnormalized. In contrast, models like KNN, Linear Regression and MLP, which rely on distance or gradient-based calculations, benefited significantly from normalization.

For evaluation, we employed both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics are expressed in the same units as the target variable (in this case, dollars), making their interpretation intuitive.

4.1 Dummy Regressor

As a baseline, we employed the `DummyRegressor` using both the mean and median strategies. Unsurprisingly, both approaches performed poorly, as they fail to capture any underlying variance in the dataset. The mean strategy yielded a MAE of **132.28** and RMSE of **170.78**, while the median strategy achieved a slightly better MAE of **127.73** and RMSE of **166.76**.

To enhance these results, we partitioned the dataset based on the `isCoach` feature, which indicates whether a flight offers economy class. For each subgroup (economy and non-economy), we applied separate `DummyRegressor` models using both mean and median strategies, and then combined the predictions. This approach resulted in a marginal improvement: the grouped mean strategy achieved a MAE of **129.86** and RMSE of **165.10**, while the grouped median strategy achieved a MAE of **129.85** and RMSE of **165.11**. Although the improvements are minimal across both metrics, they offer a slightly stronger baseline compared to the ungrouped models.

4.2 K-Nearest Neighbors (KNN) Regressor

Our first non-trivial model is the K-Nearest Neighbors (KNN) Regressor. We initially experimented with feature selection strategies, including a filter-based approach and heuristic selection via Sequential Feature Selection (SFS). However, these alternatives underperformed compared to the full-feature model, suggesting that reducing dimensionality was not beneficial in this case. As a result, we focused on the KNN model without feature selection.

To determine the optimal number of neighbors k, we tested k values in the range [1, 20], selecting the best one based on the Mean Squared Error (MSE) on the validation set. The optimal k was k = **20**. The final KNN model (with no feature selection) achieved a MAE of **78.91** and RMSE of **113.00**, representing a significant improvement over all dummy baselines. While KNN may still struggle with predicting extreme price values, it effectively captures local structure in the feature space, delivering a much more accurate overall prediction and establishing a solid foundation for further model development.

4.3 Linear Regression

In order to predict the base fare of flight tickets, regression models (Linear Regression, Lasso and Ridge Regression) were implemented. For Lasso and Ridge regression, Hyperparameter tuning was conducted using `GridSearchCV` with 5-fold cross-validation to find optimal regularization strength (alpha).

Linear Regression achieved strong performance on the validation set with relatively low MAE (**87.56**) and RMSE (**122.57**). However, its performance degraded significantly on the test set (MAE: **151.11**, RMSE: **177.67**), indicating potential overfitting and poor generalization to unseen data.

Lasso Regression selected a very small alpha (**0.001**), indicating that the model benefited from minimal regularization. While its validation performance (MAE: 94.13, RMSE: 127.24) was slightly worse than that of Linear Regression, Lasso outperformed both Linear and Ridge models on the test set (MAE: **123.47**, RMSE: **152.59**). This indicates that Lasso was more effective at improving generalization and reducing overfitting, likely by shrinking unimportant feature coefficients to zero and focusing on the most relevant predictors.

Ridge Regression, in contrast, selected a moderate alpha (**7.196**). Its validation performance (MAE: **87.88**, RMSE: **122.77**) was comparable to that of Linear Regression, and its test performance (MAE: **149.39**, RMSE: **176.09**) showed modest improvement over the Linear model. This suggests that Ridge regularization was effective in addressing multicollinearity among features and slightly reducing variance, though it did not generalize as well as Lasso.

Analysis of the linear regression coefficients revealed that certain features had a more pronounced impact on flight prices. For example, the total flight distance had a strong positive coefficient, indicating that as higher the distance of the flight, higher the price of the flight. Additionally, variables such as `IsHoliday` or `isNonStop` were among the most influential predictors, with positive coefficients reflecting their respective effects on price. Nevertheless, many other features had coefficients close to zero, suggesting a limited effect on the outcome. Due to limited generalization of linear models on the test set, a second-degree Polynomial Regression was explored, yielding a test MAE of **126**, with no improvement over the linear approaches.

4.4 Decision Trees

As part of our predictive modeling phase for forecasting flight ticket prices, we also explored and implemented three decision tree-based approaches: a standard Decision Tree Regressor, a tuned version with hyperparameter optimization, and a Random Forest Regressor.

We began with a baseline Decision Tree Regressor, trained using default settings. This allowed us to quickly establish a performance benchmark. While simple and easy to interpret, the model showed limited generalization capability on the test set and indicated high prediction error and minimal explanatory power (RMSE: **146.20**, MAE: **88.71**). This suggested the model was likely overfitting the training data due to its

unrestricted depth.

To improve upon this, we applied `GridSearchCV` to perform hyperparameter tuning. Specifically, we optimized `max_depth` and `min_samples_leaf` using 5-fold cross-validation. The best configuration found was `max_depth` = `None` and `min_samples_leaf` = 20, which helped control overfitting by limiting the complexity of the tree. The tuned model significantly improved performance, achieving an RMSE of **121.68** and a lower MAE of **78.81** compared to the baseline. These improvements highlighted the importance of tuning when using individual tree-based models.

The final and most effective model was the Random Forest Regressor. This ensemble learning method constructs multiple decision trees and averages their outputs to enhance prediction accuracy and reduce overfitting. The Random Forest was configured with **100** estimators, striking a good balance between performance and complexity, and a fixed random state was set to ensure reproducibility.

On the test set, the model achieved an RMSE of **105.41** and an MAE of approximately **73.90**. The RMSE value of approximately $105 indicates that, on average, the model's predictions deviated from the actual flight prices by about $105. This provides an interpretable measure of the typical prediction error in the context of flight fares. The MAE of **73.90** further reflects the average magnitude of the errors, without penalizing larger deviations as strongly as RMSE does.

Among the three approaches tested, the Random Forest Regressor demonstrated the best performance in terms of both prediction accuracy and explained variance. Its ensemble nature proved more robust against overfitting compared to single-tree models.

4.5 Artificial Neural Network

We implemented a Multi-Layer Perceptron (MLP) to predict the base fare of flight prices. Our MLP architecture comprises three hidden layers, each containing **128** neurons. We apply the ReLU activation function after each hidden layer to add non-linearity to the model, followed by a dropout layer to help prevent overfitting. The final output layer is a linear layer, chosen for its suitability in regression tasks such as flight price prediction.

The training was performed with the following hyperparameter settings as a baseline: a learning rate of **0.001**, a batch size of **128**, and a maximum of **100** training epochs. Early stopping was employed with a patience of **15** epochs to prevent overfitting. We used the `MSELoss` function from PyTorch as our loss function, and the Adam optimizer for parameter updates. After 63 training epochs with early stopping, our model achieved a validation RMSE of **97.3**, test RMSE of **148.19**, and a test MAE of **114.99**. From the results, we can observe overfitting of the model.

We performed a random search to tune four key hyperparameters of our ANN model: learning rate (0.01, 0.001, 0.0001), batch size (64, 128, 256), dropout rate (0.0, 0.2, 0.5), and early stopping patience (10, 15, 20). The search space consisted of 81 possible combinations, and we randomly sampled **20** unique configurations.

Among all configurations, the model with the lowest validation RMSE (learning rate=0.0001, batch size=128, patience=15, dropout=0.0) achieved a validation RMSE of **91.16** but a test RMSE of **136.64** and a test MAE of **100**. In contrast, the model with

the lowest test RMSE (learning rate=0.001, batch size=256, patience=20, dropout=0.5) yielded a slightly higher validation RMSE of **105.56**, but performed better on the test set with a test RMSE of **124.88** and a MAE of **97.48**. This suggests that while the former model may have overfitted to the validation data, the latter demonstrated better generalization to unseen data.

The optimal performance is achieved with a moderate learning rate (0.1), larger batch sizes (128 or 256), longer early stopping patience (15–20 epochs), and lower dropout rates (0.0–0.2). This suggests that the model benefits from faster convergence and stable training, with minimal need for regularization, likely due to the large dataset size and low risk of overfitting.

4.6 Feature Importance Analysis on Random Forest Regressor

To gain insights into which features contribute most to the model's predictions, we computed feature importances using the impurity-based method provided by the `RandomForestRegressor`. The top three most important features were `isCoach`, `total_distance`, and `totalFlightTime`. The feature `isCoach`, indicating whether the ticket is in coach class, had the highest importance score—even though only around 0.002% of the instances in the dataset had `isCoach` set to false. This suggests that the presence of premium class tickets, while rare, has a strong impact on pricing patterns. This was followed closely by `total_distance`, which reflects the length of the journey and typically correlates with costs such as fuel and logistics. The third most important feature, `totalFlightTime`, may capture additional pricing factors related to flight duration, such as layovers or aircraft usage time.

Also notable are the fourth and the fifth feature, `flightDate_as_int` and `searchDate_as_int`, which encode the flight and search dates numerically. Their importance suggests that temporal aspects—such as seasonality or how far in advance a flight is searched—also play a meaningful role in shaping price behavior.

5 Results

The Random Forest Regressor achieved the best performance with an RMSE of **105.41**, meaning its predictions deviate from actual prices by about **$105** on average. It was followed by the KNN Regressor with an RMSE of **113**, and both models significantly outperformed the baseline RMSE of **165.1**. In contrast, the Linear Regressor and Multi-Layer Perceptron performed worse. The Linear Regressor likely struggled due to the non-linear nature of flight pricing, while the MLP's complexity may not have aligned well with the structure of this task. Although KNN achieved reasonable accuracy, its high prediction-time cost makes Random Forest a more practical option.

Considering the average flight price of **$398** and a standard deviation of **$186**, the Random Forest's error is relatively low—especially given the skewed distribution and presence of outliers—suggesting strong generalization. Feature importance analysis further supports its effectiveness in capturing key pricing patterns, reinforcing its value for accurate and interpretable flight price prediction.

Bibliography

Bureau of Transportation Statistics (2023, March). Full year 2022 u.s. airline traffic data. Accessed: 2025-05-16.

dilwong (2021). Flight prices. Accessed: 2025-05-16.

Korkmaz, H. (2024, November). Prediction of airline ticket price using machine learning method. *Journal of Transportation and Logistics 9*(2), 1–14. Accessed: 2025-05-16.

Skyscanner (2025, April). How does airline pricing work? Accessed: 2025-05-16.

Wang, X., Y. Chen, and Y. Zhang (2019). A framework for airfare price prediction: A machine learning approach. In *2019 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 123–130. IEEE. Accessed: 2025-05-16.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT4o	Rephrasing	Throughout	++

B. Fiorillo C. Lin J. Miranda,

Unterschrift
Mannheim, den 18. 05 2025 J. Park T. Nachev G, Kim