Basic Flight Fare Prediction System Final Report

Members: Subhan Mehmood, Hashirul Quadir, Dionissios Kakissis, Mohammad Bakeer, Abiha Fatima

Contributions:

Group Members	**Contributions**
Subhan	Code/Notebook, Report
Hashirul	Report, Poster
Dionissios	Code/Notebook, Report
Mohammad	Report, Web App
Abiha	Code/Notebook, Report

Introduction:
Predicting the price of flight tickets can be challenging, especially considering the numerous complex factors that affect airfare, such as airline, travel time, and duration. It can be difficult for travelers looking for a flight to predict prices reliably. As such, the goal of this project is to train a series of machine learning models that can predict the price of tickets based on user-specified details, like origin, destination, date, etc., in which the best model would be extracted and implemented into a web application to be utilized by various users to get a predicted price of their flight.

Literature Review:
A related work we found is a project titled 'Flight Fare Prediction' by Kunal Dhavale. In this project, he scrapes data from the online travel website EaseMyTrip via BeautifulSoup. Once he collects the data, he cleans and preprocesses the dataset. Afterward, he performs exploratory data analysis to extract the significant features of the dataset and convert them to the relevant types needed for model training. To predict flight prices, Dhavale trains four models: Linear Regression, Random Forest, XGBoost, and Extra Trees Regression. After such training, he outputs the statistics for each model, such as accuracy, and concludes that the Random Forest and Extra Trees Regressor models are the best. Finally, he extracts the Random Forest model and implements it in his web application, which he created from scratch, where users can enter their flight details to get a predicted price.

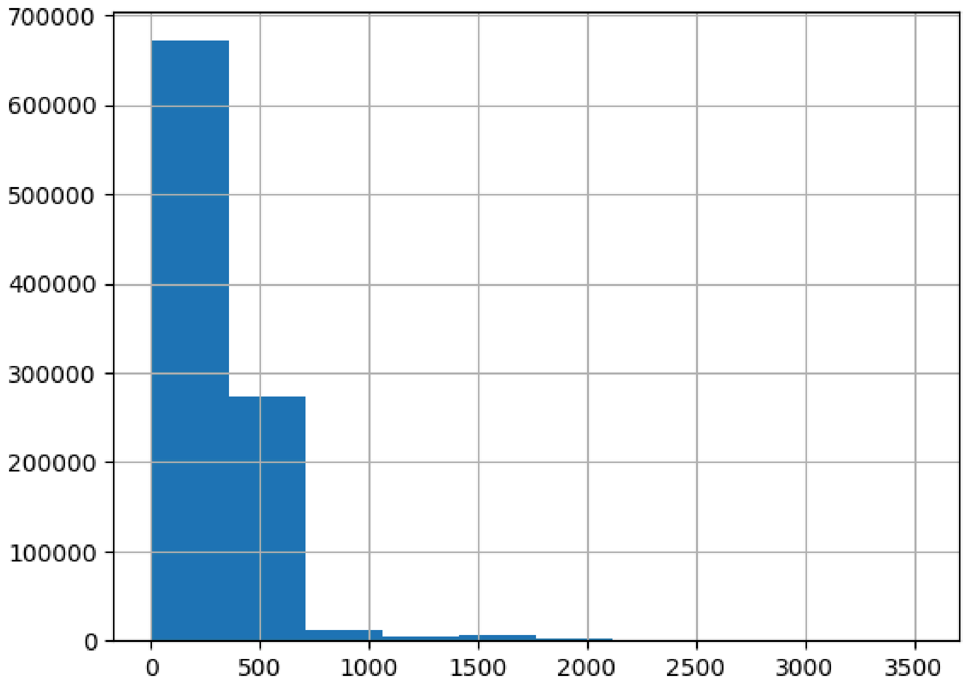
Comparing our work to his, we have a complementary system using similar approaches and models, but the main difference is the complexity and expansiveness of our dataset. Dhavale uses a broad dataset, while our dataset is more complex and contains many in-depth observations and features about flights. Additionally, preprocessing and cleaning our dataset was a more lengthy process compared to Dhavale's, since we had to take into account more features that were missing, insufficient, or unnecessary. Furthermore, we trained the same models as Dhavale but acquired vastly different statistics for our models' performances, notably the error rate and accuracy. However, both of our projects came to the same conclusion about the best model, which was Random Forest. Lastly, our web application will be similar, but Dhavale's application contains basic features, such as origin and destination, while our application will include similar features, as well as more in-depth features, like flight distance, travel duration, and a few others. Overall, our and Dhavale's project are similar, but different approaches are taken in some circumstances, such as dataset and feature extraction/engineering.

Dataset:
Our chosen dataset comes from Kaggle, titled Flight Prices, and includes data on one-way flights found on Expedia between 04-16-2022 and 10-05-2022. The dataset contains 5999739 unique values and 27 columns. The dataset contains both quantitative and qualitative values.

To preprocess and clean the data, we first chose the first 1 million observations, which later decreased due to preprocessing and data cleaning, from the dataset’s approximate 6 million, so that it would be more manageable and less computationally expensive. Then, we printed out the observations that were missing from the dataset. We filled the missing observations with the median of the column because the median would be more robust to outliers, and we could not delete such observations, as they would come in handy for our project. Afterward, we converted the duration feature, which was in hours and minutes, into total seconds and also summed the flightDurationInSeconds feature where pipe-separated values existed. Next, we deleted rows that had empty segmentsEquipmentDecription and segmentsCabinCode observations since we could not impute them. We converted the flightDate feature into an ordinal format in the month/date/year format for ease of use in our models. We also label encoded categorical variables for Random Forest and XGBoost, and one-hot encoded for Linear Regression. Finally, we saved our preprocessed dataset and zipped it so that it would be easy to push it to GitHub and so that we would be able to open it on our laptops. Both the original and preprocessed datasets are in zipped format so that they are manageable on our computers.

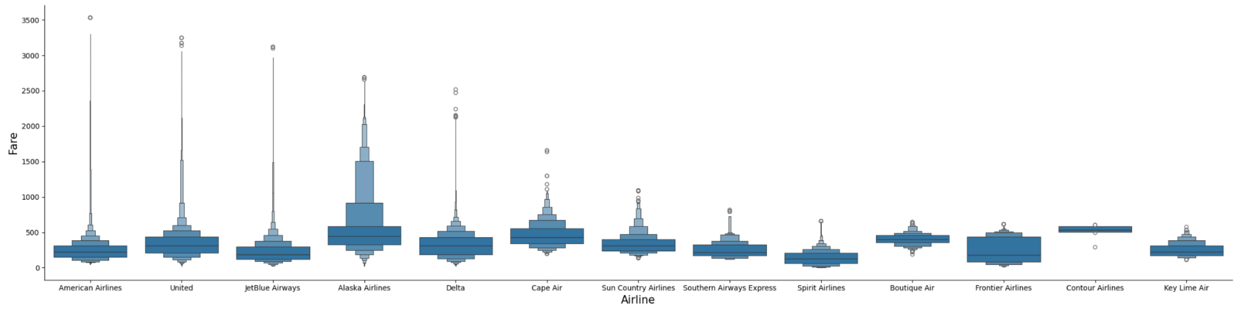
Exploratory Data Analysis and Feature Importance:
Before we trained and implemented our models, we went through exploratory data analysis to learn more about the dataset.

We plotted the basicFare feature on a histogram to learn how the prices were distributed.



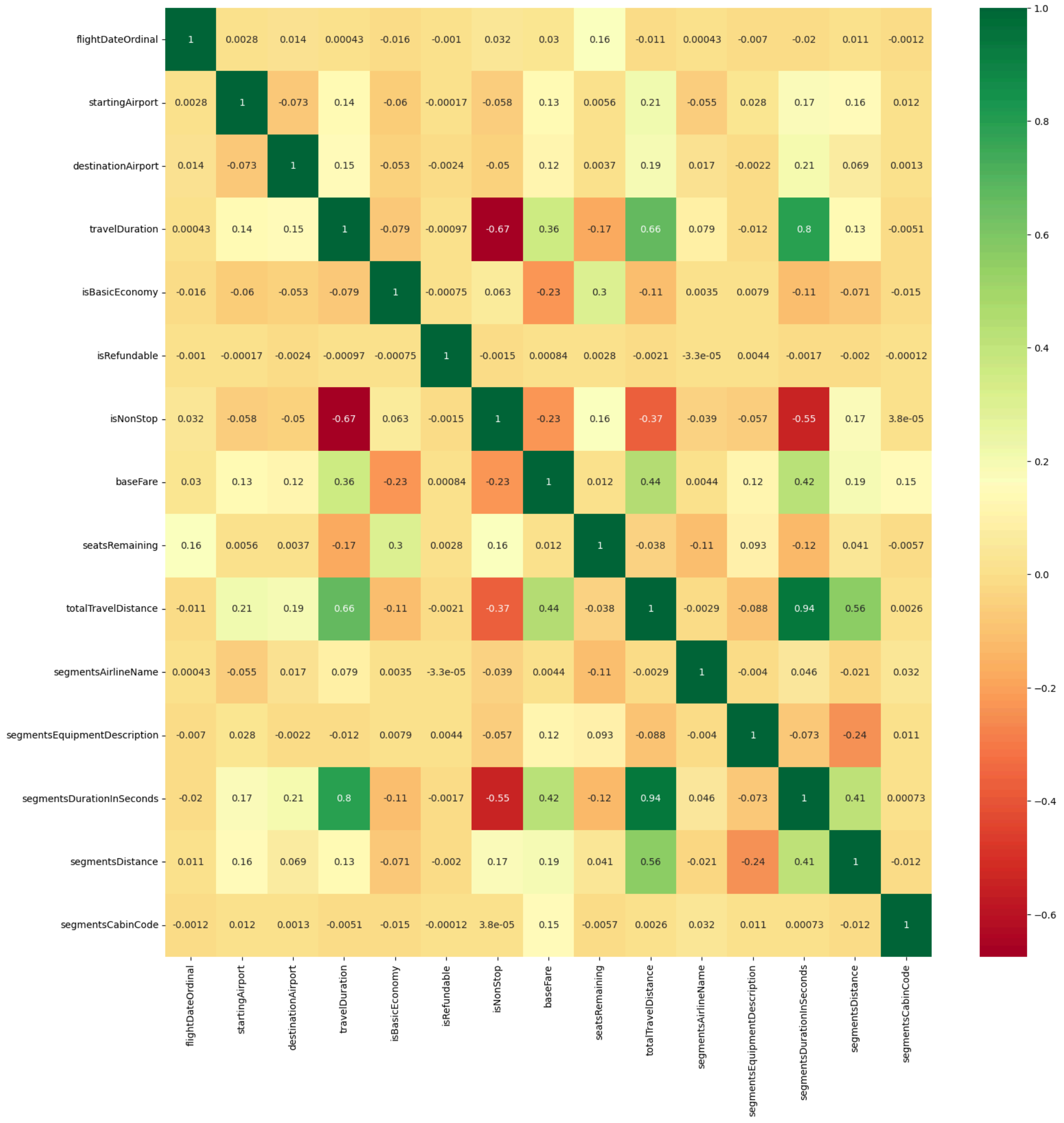
The histogram reveals that most of the prices are under $500, with only a small portion of the prices being on the high end, indicating that the flight prices are relatively low. Additionally, the histogram shows that the prices are right-skewed towards the high-end prices.

We also created a boxen plot, which visualizes the distribution of baseFare between different airlines.



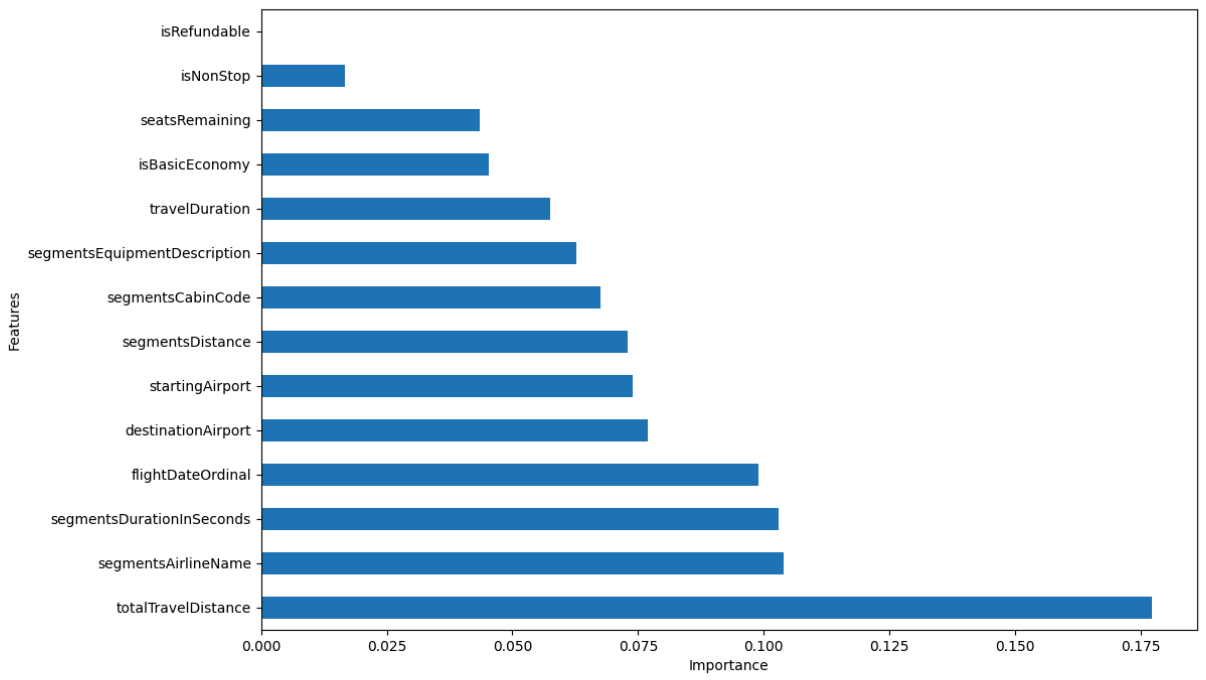
The plot shows that some airlines had wide fare ranges and outliers while others did not, which suggests that specific airlines can affect the price of tickets and, in turn, affect the model's ability to generalize.

Furthermore, we created a correlation heatmap to ascertain our important features for model training to see the correlations between features.



We found that features such as totalTravelDistance, segmentsDurationInSeconds,segmentsDistance, and travelDuration, had a strong correlation. Hence, we considered them significant. There were other features that we took into account.

Additionally, we implemented an Extra Trees Regressor Model to identify the most impactful features via the relative importance scores.



From the model, we totalTravelDistance, segmentsDurationInSeconds, travelDuration, flightDateOrdinal, isBasicEconomy, segmentsAirlineName, startingAirport, destinationAirport, segmentsDistance, and segmentsCabinCode as our confirmed significant features for our dataset and models.

Lastly, we implemented a Recursive Feature Elimination (RFE) algorithm via Random Forest to rank feature importance and reinforce our confirmed significant predictors, which resulted in the final list of feature importance being: totalTravelDistance, segmentsDurationInSeconds, travelDuration, flightDateOrdinal, isBasicEconomy, segmentsAirlineName, startingAirport, destinationAirport, segmentsDistance, and segmentsCabinCode.

Baseline:
We implemented a Multiple Linear Regression Model as our baseline. It was a basic linear model trained on one-hot encoded data from the dataset. This baseline was implemented to see if there was a linear relationship between the baseFare, our target variable, and the numerous features, such as flightDuration, flightDate, etc. We trained our Multiple Linear Regression Model on a one-hot encoded version of our dataset, which was necessary to handle the categorical variables, such as airline names. Based on our exploratory data analysis, we found ten features from the flight dataset to be important in predicting airfare. These features were totalTravelDistance, segmentsDurationInSeconds, travelDuration, flightDateOrdinal, isBasicEconomy, segmentsAirlineName, startingAirport, destinationAirport, segmentsDistance, and

segmentsCabinCode. Again, categorical variables were one-hot encoded, and then all important features were utilized in training the Multiple Linear Regression Model.

When implementing this model, we applied a 70/30 split, where 70% of the sampled dataset we were working with was used for training the model, and the other 30% was used for testing the model for validation. The model was not trained with regularization and was a pure baseline.

The LinearRegression() function was used to implement the Linear Regression Model via the scikit-learn library in Python.

Main Approach:
In our minds, the three models we are training, Multiple Linear Regression, Random Forest, and Gradient Boosting Regression, are our main approaches, but Multiple Linear Regression is classified as our baseline, and the Gradient Boosting Regression is our oracle. In this scenario, the Random Forest Model is our main approach. We chose this specific model because of its robustness to outliers and noisy data and its ability to handle and show complex and non-linear relationships, which is needed in our case due to the nature of flight prices.

Similar to the Multiple Linear Regression Model, the Random Forest utilizes the 10 significant features we noted during our exploratory data analysis as input, which were totalTravelDistance, segmentsDurationInSeconds, travelDuration, flightDateOrdinal, isBasicEconomy, segmentsAirlineName, startingAirport, destinationAirport, segmentsDistance, and segmentsCabinCode. In this case, label encoding is used for the categorical variables, and the features are scaled appropriately wherever necessary. The output is the predicted flight ticket price.

The Random Forest was implemented via a 70/30 split, where 70% of the sampled dataset is used for training the Random Forest and the other 30% for testing the model (unseen data).
Additionally, we specified three parameters for our Random Forest: the n_estimators, which is set to 100, signifying there will be 100 trees built for stable prediction and better performance, random_state = 42 used for reproducibility of results, and n_jobs = -1 to use all CPU cores for faster training.

The RandomForestRegressor() function was used to implement the Random Forest Model via the scikit-learn library in Python.

Furthermore, we implemented another model, an XGBoost Model. This is the oracle of our project. We chose this model because of its computational efficiency and speed, its ability to capture non-linear and complex relationships, and so that it can be used as a benchmark model. Similar to the Multiple Linear Regression Model and Random Forest, the XGBoost model used the 10 significant features- totalTravelDistance, segmentsDurationInSeconds, travelDuration, flightDateOrdinal, isBasicEconomy, segmentsAirlineName, startingAirport, destinationAirport, segmentsDistance, and segmentsCabinCode- within our model. These important features were also labeled and encoded for the categorical variables and scaled accordingly when needed. The output for this model was again the predicted airfare.

This model was also implemented with a 70/30 split, where 70% of the sampled dataset was used for training, and the other 30% was used for testing (unseen data). The parameters specified were similar to the Random Forest, where n_estimators was set to 100, indicating that 100 trees will be built for steady predictions and improved performance, random_state was set to 42 to reproduce results whenever necessary, and n_jobs was set to -1 to utilize all CPU cores for faster training time.

In this case, the XGBRegressor() function was used to implement the XGBoost Model through the scikit-learn library in Python.

Evaluation Metric:
For our metrics, we are using the Root Mean Squared Error (RMSE) because it will allow us to interpret and compare models and penalize larger errors more heavily than minor errors, and the Mean Absolute Error (MAE) because it will enable us to measure the average error of the ticket prices in dollars. RMSE will work well here because the prices of flight tickets have a broad range, and slight differences can impact what the user chooses. MAE will work well because it will help interpret the models' errors reasonably and practically. R^2 is also used to check how well the models explain the pricing variations.

Equation for Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Equation for Mean Absolute Error (MAE):

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Equation for R^2:

$$1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

Moreover, as previously hinted, we defined a custom accuracy function based on the Mean Absolute Percentage Error (MAPE) to evaluate our models' performance further. In this case, the custom accuracy function measures how close each model's predictions are to the actual values, expressed as a percentage. As accuracy is used to evaluate models concerning classification problems, this accuracy percentage is similar but adapted for our regression problem.

Equation for Mean Absolute Percentage Error (MAPE):

$$\frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

The custom accuracy function is based on the MAPE equation. Here, the custom accuracy is calculated by subtracting 100 from the determined MAPE of a model. Higher percentages are indicative of better model performance. Utilizing such a metric is easy and understandable for us, as it is familiar.

Accuracy = 100 - MAPE

Results & Analysis:

<u>Multiple Linear Regression:</u>

This is what we achieved in terms of metrics and performance for the Multiple Linear Regression Model (Baseline):

Mean Absolute Error (MAE): 100.0955
Root Mean Squared Error (RMSE): 164.8059
R^2: 0.4332

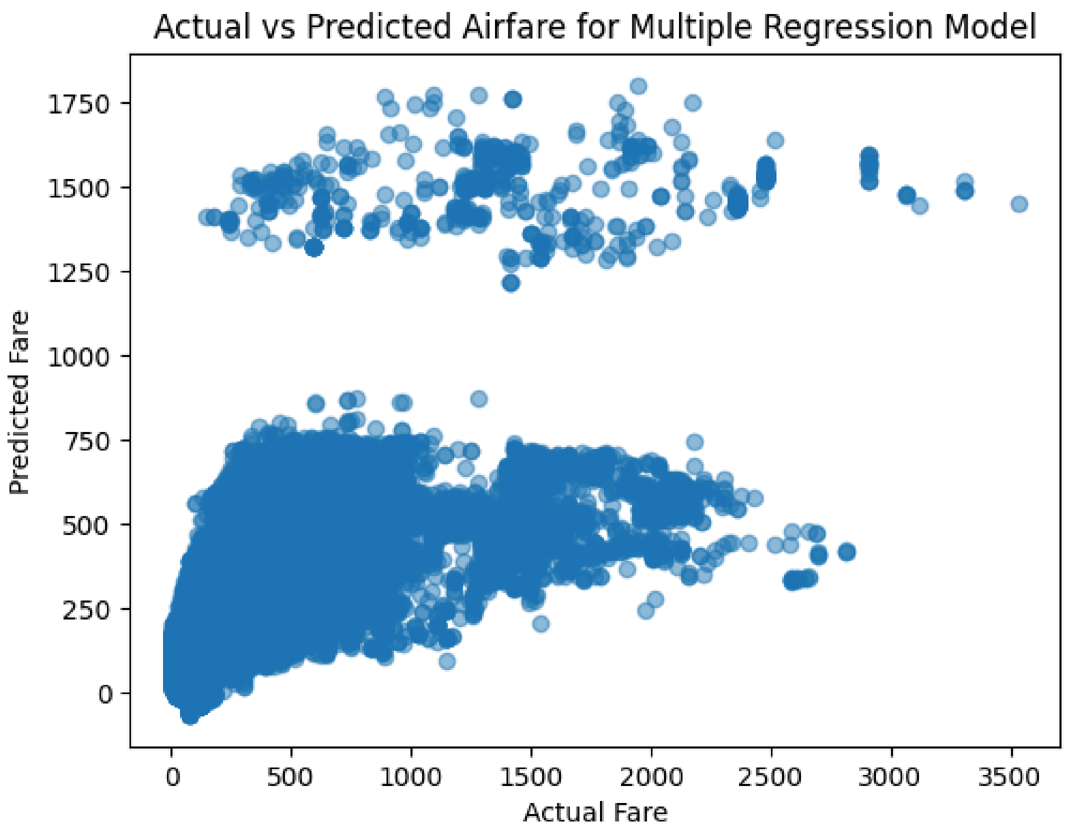
Custom Accuracy Based on MAPE: 51.2122 → 51.21%

The R^2 of 0.4332 indicates that approximately only 43.32% of the variance in the airfare prices can be explained by the linear model. This indicates that the linear model is relatively moderate in predicting airfare prices and is unsuitable.

Moreover, the MAE of 100.10 means that, on average, the Multiple Linear Regression Model's predicted airfare is off by about $100.10 compared to the true price. The RMSE of 164.81 shows that big mistakes/errors, such as outliers, heavily influence the model and that some predictions are very off. Both of these metrics are high, indicating considerable errors and that the linear model struggled to predict flight prices.

Additionally, the custom accuracy that was calculated for the Multiple Linear Regression Model results in the value of 51.21%, the lowest of the three models trained, which indicates that the accuracy of this particular model on this dataset is relatively moderate and not great, which enforces the idea of this linear model not being suitable in this scenario.

Below is the actual vs predicted scatter plot of the Multiple Linear Regression Model:



From this plot, it can be noted that the points are widely spread out, and there is a lot of variability. The predictions are not consistent, and this can be seen especially for the higher actual fares. Additionally, it can be seen from the plots that there are two separate clusters, so the data is being separated into two groups, one for cheaper flights, most likely economy class, and one for expensive flights, most likely business and premium classes from the dataset; and is predicting similar prices within each group. Also, the model sometimes overpredicts lower prices, assessing them as higher prices when they are not, and sometimes underpredicts higher prices as lower prices.

All in all, it is clear that the features in the dataset do not have a linear relationship with airfare prices. Instead, a complex and non-linear relationship exists. Therefore, the baseline model we implemented performed inadequately in this case. However, this is to be expected since, in the real world, the price of flight tickets does not have a simple linear relationship with other attributes but has a convoluted relationship with various factors affecting prices.

Random Forest:

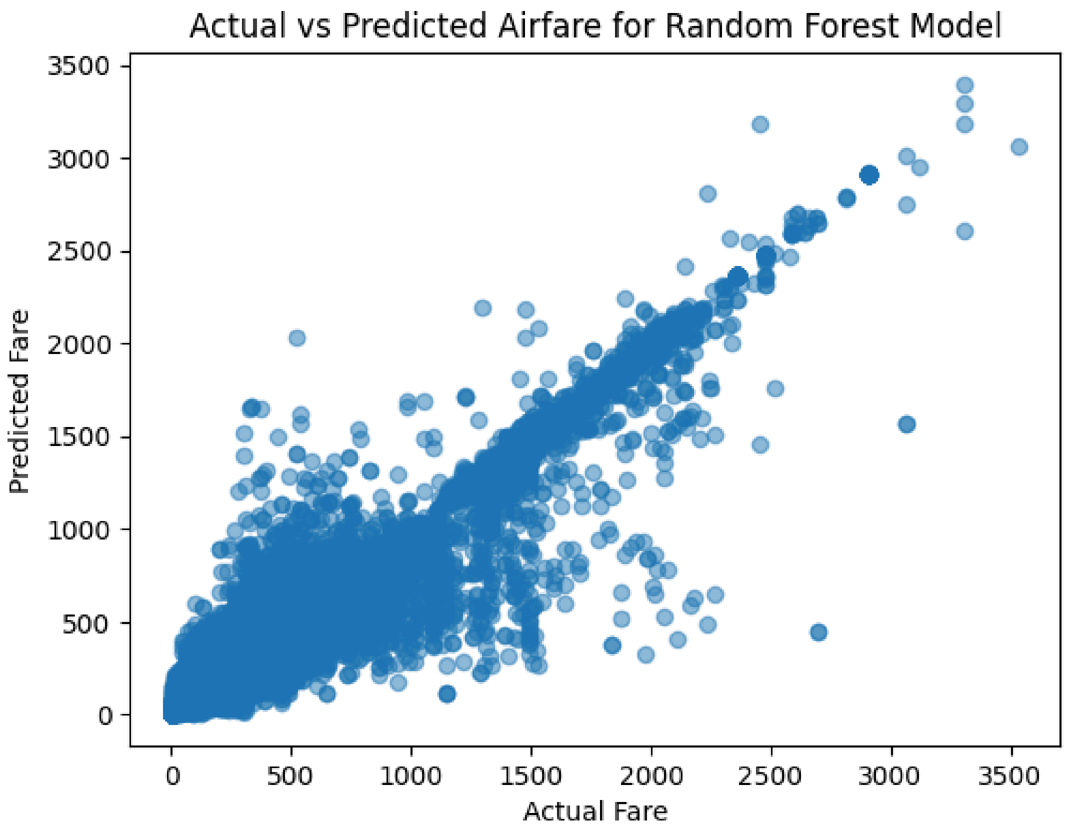
This is what we achieved in terms of metrics and performance for the Random Forest Model (Main Approach):
Mean Absolute Error (MAE): 30.8358
Root Mean Squared Error (RMSE): 57.9759
R^2: 0.9299
Custom Accuracy Based on MAPE: 85.9594 → 85.96%

Here, the R^2 of 0.9299 indicates that approximately 92.99% of the variance in the airfare prices can be explained by the Random Forest Model. This shows that the model is relatively good at predicting the price of flight tickets.

Moreover, the MAE of 30.84 means that, on average, the XGBoost Model's predicted airfare is off by about $30.84 compared to the true price. The RMSE of 57.98 shows that the errors are low overall, and there are few to no large outliers. Both of these metrics are relatively low, indicating that this model achieved a significant reduction in error and is performing better overall.

The custom accuracy calculated for the Random Forest Model also resulted in 85.96%, the highest of the three models trained. This indicates that this particular is really good for this dataset and a better model for predicting flight prices, which enforces the idea that this model is suitable for this circumstance.

Below is the actual vs predicted scatter plot of the Random Forest Model:



This plot shows that the points are tightly clustered around a 45-degree line, indicating that the actual fare is equal to the predicted fare. In other words, the model accurately predicts most flight ticket prices. Additionally, the points are in a tight spread, and not many points stray from one another or have many outliers, suggesting that the model is generalizing over the various price ranges and that most of the predictions are reliable. Also, there aren't any groupings or clusters of groups distinct from one another, which implies that the model can capture the continuous relationship between the features and airfare without flattening or oversimplifying any attributes.

Overall, it is clear that the Random Forest Model was able to better capture the non-linear and complex patterns of the data effectively, as seen from the metrics. As such, this model was the best of the three models we trained and is being collected to be implemented within our web application.

<u>XGBoost Model:</u>

This is what we achieved in terms of metrics and performance for the XGBoost Model (Oracle):
Mean Absolute Error (MAE): 61.6697

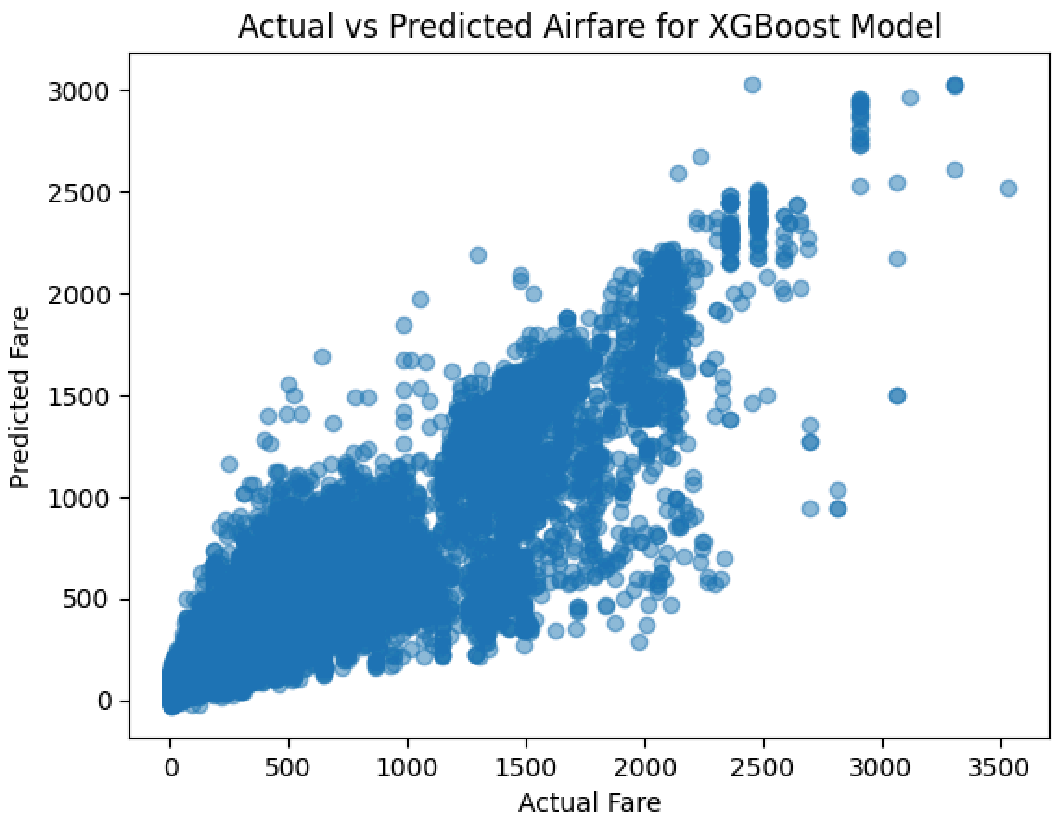
Root Mean Squared Error (RMSE): 95.1599
R^2: 0.8110
Custom Accuracy Based on MAPE: 69.9640 → 69.96%

For this model, the R^2 of 0.8110 indicates that approximately 81.10% of the variance in the airfare prices can be explained by the XGBoost Model. This shows that the model is moderate and decent at predicting the price of flight tickets.

Moreover, the MAE of 61.67 means that, on average, the XGBoost Model's predicted airfare is off by about $61.67 compared to the true price. The RMSE of 95.16 shows that even though most predictions are close, some large errors are still pulling the RMSE higher. These metrics are relatively high, implying that this model had extensive errors and struggled to predict flight prices fully.

Additionally, the custom accuracy that the XGBoost Model achieved was 69.96%, better than the Linear Model but lower than the Random Forest. This suggests that this particular model is decent and predicts flight prices, but it is not the best for our case.

Below is the actual vs predicted scatter plot of the Random Forest Model:



From this plot, the points are clustered fairly closely along a 45-degree line, suggesting that the general trend of the airfares is captured, but not as good as the Random Forest Model. The points are more spread out, so the model struggled with the more expensive prices compared to the Random Forest. Also, the points that exceed $1000 seem to spread out more and have more variance, so the XGBoost model has some difficulty with more expensive prices. Actual prices from around $2000 to $3000 are predicted to be lower than expected.

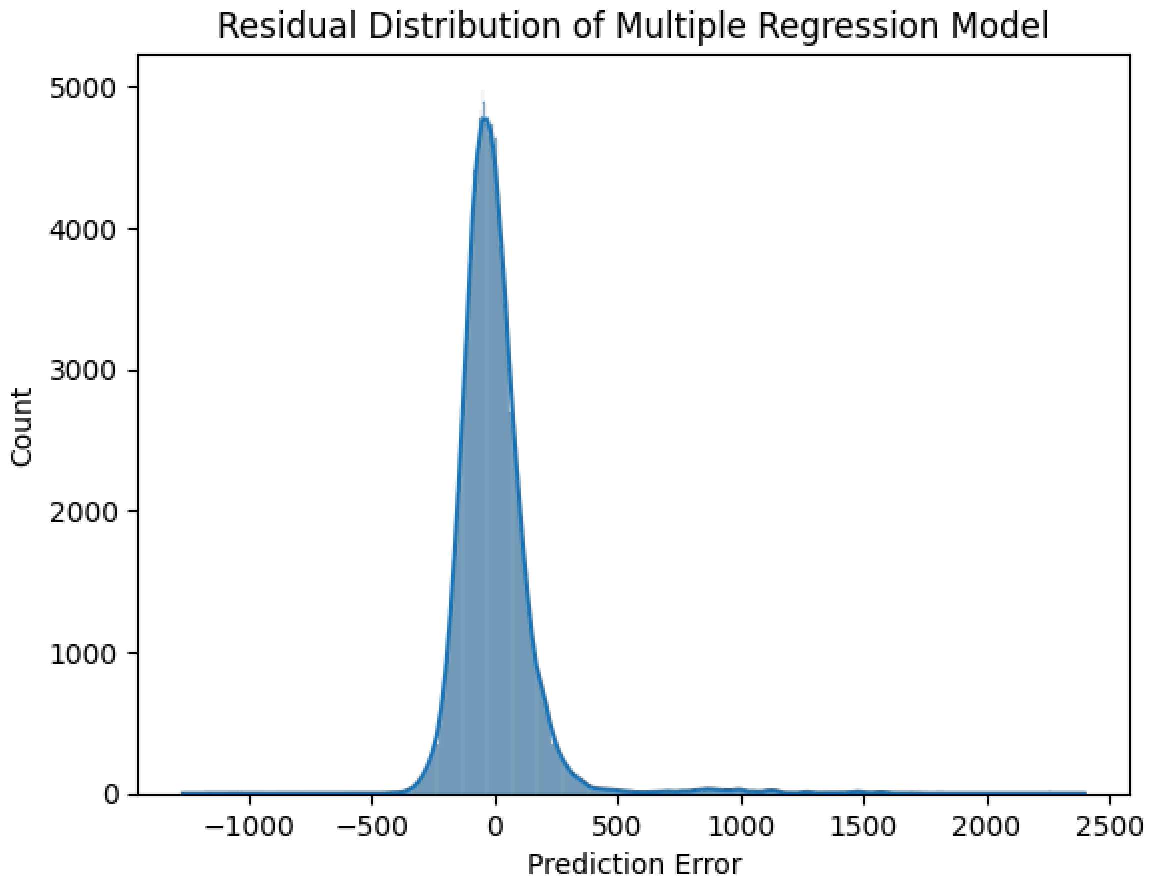
Altogether, we can see that the XGBoost Model is decent in predicting airfare, especially compared to the Multiple Linear Regression Model, and can capture the non-linear and complex relationships between features and the price moderately, but, in the end, the Random Forest Model does a better job in all aspects in comparison to the XGBoost Model, and as such this model is not the best for this scenario.

Ultimately, the Random Forest Model performed the best, followed by the XGBoost Model. The Multiple Linear Regression Model did the worst out of the three. The good performance of the Random Forest Model allows us to implement this model within our web application to create a system that will enable users to predict their flight prices based on their flight details.

Error Analysis:
One experiment we ran was implementing three separate models, Multiple Linear Regression, Random Forest, and XGBoost, where each model was trained on the same features via a 70/30 train-test split to provide a fair performance comparison. The pros of doing such a thing allowed us to benchmark performance and compare the different models. Each model also provided different assumptions and depth to our analysis, such as Linear Regression providing interpretability, Random Forest providing a way to capture nonlinear interactions, and XGBoost providing gradient-boosting. The vast difference in performance between Linear Regression and the tree-based models led us to learn that airfare prediction is not a linear problem, and as such, complex models with access to nonlinear analysis are needed to provide appropriate exploration. However, there were still cons to this, such as increased computation time, as training these specific models, especially on such a large dataset, was computationally expensive. We also had to deal with complex model management. Each model required different preprocessing steps, such as using one-hot encoding for Linear Regression and label encoding for the tree-based models.

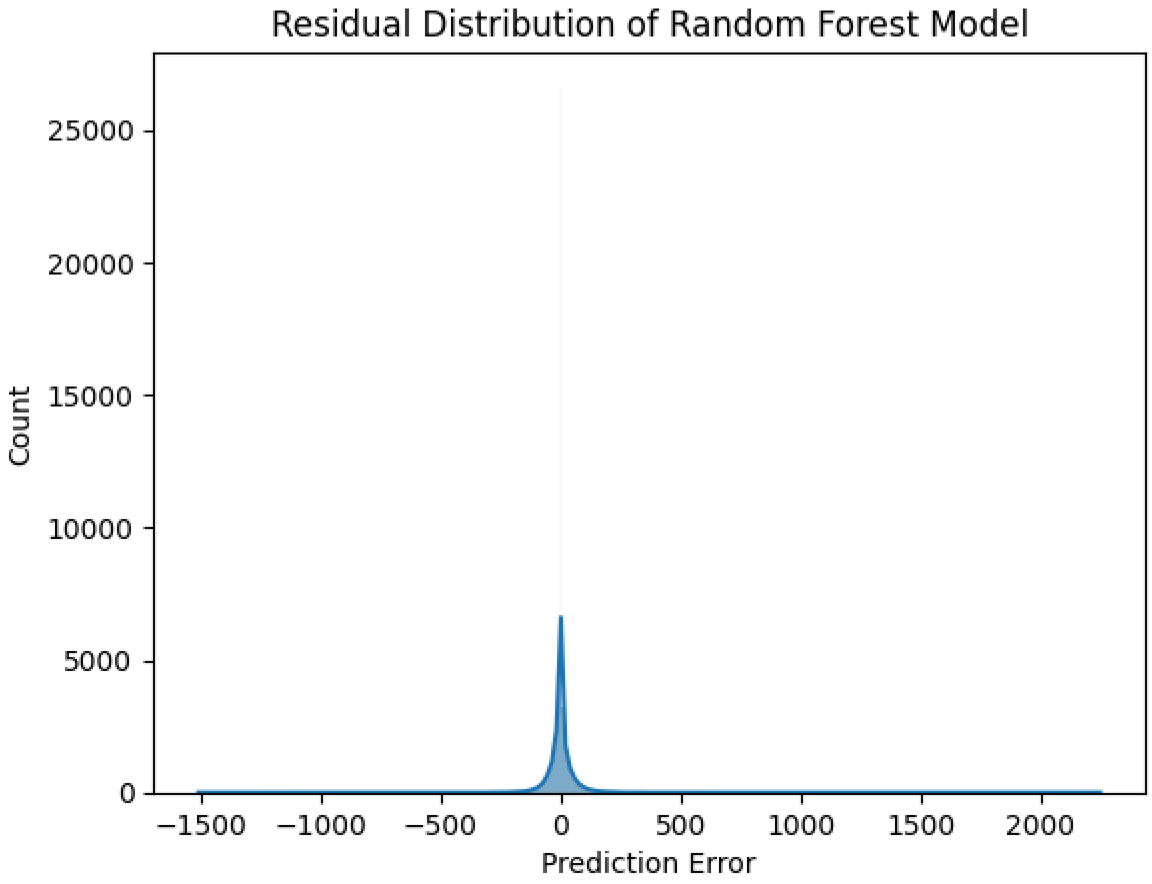
Another experiment led us to generate residual plots for each model. These plots calculated the difference between the actual and predicted airfare for each test observation. Observing the plots aided us in understanding the bias, error spread, and consistency of each model. The pros of this experiment were that they helped us visualize the consistency of the models in another way, were easy to interpret, and showed the bias of each of the models, if any. The cons of this experiment were that this visualization is not fully helpful without also knowing the other evaluation metrics for each model, such as RMSE and MAE, so we needed to combine multiple evaluation metrics to gauge model performance, requiring more computation, time, and code. Also, these plots do not identify which observations or features cause the errors that are shown within the plots, so it becomes challenging to find the faults in our models, dataset, etc., and requires alternate approaches to remedy this, again, needing more computation, time, and code.

Residual Distribution of Multiple Regression Model:



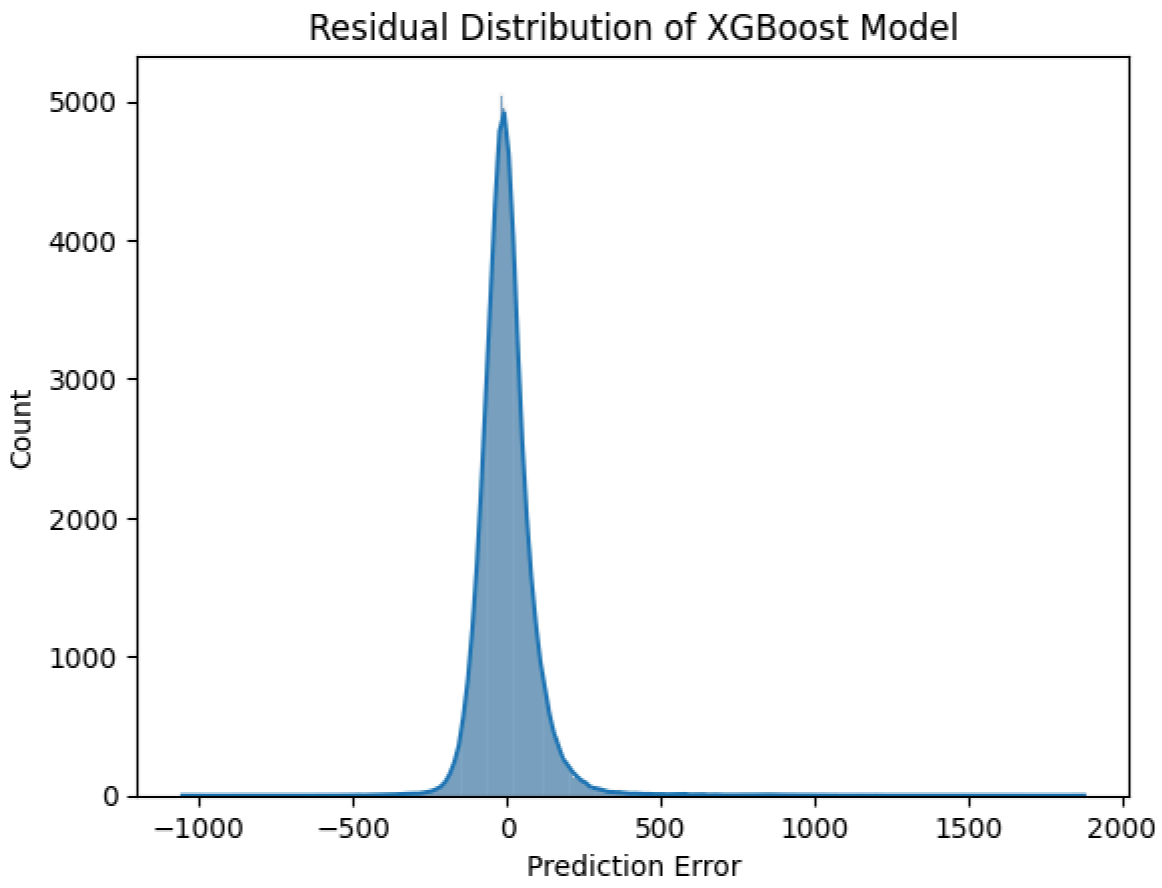
The residuals had a wider and flatter distribution for the Multiple Linear Regression Model compared to the other models. The center is near 0, but the error spread is large, and the tails extend extensively in both directions, which indicates that this model often made large errors and both underpredicted and overpredicted the flight prices. This further reinforces the claim that the Multiple Linear Regression Model is unsuitable for this dataset and performed the worst out of the three models.

Residual Distribution of Random Forest Model:



The residuals of the Random Forest Model formed a compact, tight, and symmetric peak centered at 0. The error spread was very low, indicating that the predictions were very close to the actual values. The height of the plot shows that the model often produced low-error predictions, which reinforces our analysis that the Random Forest Model is the best model in this case and a good model for predicting airfare.

Residual Distribution of XGBoost Model:



The XGBoost Model's residual plot is more compact but marginally larger than the Random Forest's. The plot distribution is relatively symmetric and centered around 0 with a moderate spread, seen more on the positive side of the tail. The residuals suggest that the XGBoost Model can be more tuned, especially for expensive prices. Ultimately, the XGBoost Model performed better than the Multiple Linear Regression Model, but not as well as the Random Forest.

Furthermore, the takeaway message is that the Random Forest model is clearly the better-performing model for its robustness, statistics, and consistency. The XGBoost Model was decent in its performance, but did not compare to the metrics of the Random Forest. The Multiple Linear Regression Model performed the worst in all metrics compared to the other models.

Moreover, we were surprised that the Multiple Linear Regression Model performed worse than we had expected, even with the basic preprocessing. Additionally, the XGBoost model struggled to predict more expensive prices, which surprised us.

In addition, potential errors in the methods were skewed price distributions, unbalanced class distributions, and outliers. The skewed price distribution meant that the prices were concentrated around the lower to mid-range area, and that expensive fares were not frequent in the dataset.

This explains why the residual plot of the XGBoost Model presented itself as it did, where there were more error spreads for high-fare predictions due to the limited observations connected to high prices, leading to a skewed price distribution. Features, such as cabin class and airline, implicitly defined groups for the prices, where certain combinations were more frequent than others when training the data. As such, the less common combination of features resulted in weaker generalization, as seen for the Multiple Linear Regression Model, leading the model to fail to adapt to the price patterns, giving poor model performance because of the unbalanced class distributions. Outliers were present in the dataset, which influenced these combinations. As such, the long tails in some of the residual plots for the models, more specifically for the Multiple Linear Regression and XGBoost Models, pulled the error metrics higher than they should have, giving us mediocre results and performance of the models.

Future Work:
If we had more time, one way we could improve our model(s) would be to do further feature engineering. We could extract additional features such as airline categories, i.e., cheap and premium airlines, or extract seasonal travel statistics to capture seasonal trends and price patterns more precisely and accurately for better predictions. Another way could be to expand our dataset when implementing it within our model. Currently, we have only utilized the first 1 million observations from the dataset, which is close to 6 million. If we had considered more observations, maybe even the entire dataset, we could have improved our models. However, computational and time costs were not in our favor, so we did not do so. Instead, we went with the first 1 million observations. Possibly utilizing GPU-accelerated techniques with a good GPU unit could allow us to accomplish such a thing. Lastly, we could have also trained a couple more models, such as Neural Networks, and used them as a comparison to get the best model again. But, again, we could not do so due to our limited computational power and training time, and GPU acceleration and better parallel processing might have helped here.

References

"CodeCogs Equation Editor." *CodeCogs*, www.codecogs.com/latex/eqneditor.php. Accessed 6 Apr. 2025.

Dhavale, Kunal. "Flight-Fare-Prediction." *GitHub*, 2023, github.com/KunalDhavale/Flight-Fare-Prediction/tree/master. Accessed 11 Mar. 2025.

Wong, Dillion. "Flight Prices." *Kaggle*, 2023, www.kaggle.com/datasets/dilwong/flightprices. Accessed 11 Mar. 2025.