Bilkent University

Department of Computer Engineering

CS 464 Term Project

Fall 2023-2024

Section 01

Group 04

Project Progress Report

Flight Price Prediction in India

Group members:

Emre Karataş 22001641

Gökay Özbay 21901888

Mustafa Kütükcü 21902972

Ege Berkay Karagenç 22002799

Erim Berke Salman 22001964

Instructor:

Ayşegül Dündar Boral

Table of Contents

Data Analysis & Preprocessing

The dataset used in the project, found on Kaggle, is called *"Flight Price Prediction"* [1]. This dataset contains information about flight booking options from the website Easemytrip for flight travel between India's five metro cities. We used a file named 'Clean_Dataset.csv' among the three CSV files. It consists of 300261 flight entries, and the data includes airline, flight, source and destination city, departure and arrival time, stops, class, duration, days left, and price [1]. The dataset is split into train, validation, and test as 80%, 10%, and 10%.

"Clean_Dataset.csv" contains the following features [1]:

- **Airline:** Name of the airline (6 categories).
- **Flight:** Flight code (categorical).
- **Source City:** Departure city (6 unique cities).
- **Departure Time:** Time of departure, categorized into six time bins.
- **Stops:** Number of stops (3 distinct values).
- **Arrival Time:** The time of arrival is categorized into six-time bins.
- **Destination City:** Arrival city (6 unique cities).
- **Class:** Seat class - Business or Economy.
- **Duration:** Flight duration in hours (continuous).
- **Days Left:** Days remaining until departure (derived).
- **Price:** Ticket price (target variable).

Before starting the data analysis, we ensured that the dataset contained no missing or inconsistent values.

```
Missing Values Count:
 Unnamed: 0          0
airline              0
flight               0
source_city          0
departure_time       0
stops                0
arrival_time         0
destination_city     0
class                0
duration             0
days_left            0
price                0
dtype: int64
```

Figure 1: Missing Values Checking

```
[ ]  # Checking for negative values in columns where it is not meaningful
     for column in ['duration', 'days_left','price']:
         if (df[column] < 0).any():
             print(f"Negative values found in {column}")
```

```
No negative values found for real valued columns.
```

Figure 2: Data Inconsistency Checking

We also removed a column named “unnamed:0”, the row iteration keeper for the dataset, which does not influence the developed models.

After ensuring that the dataset does not contain missing or inconsistent values, we decided to explore the categorical features of the dataset. We received the following plottings using Pandas, Matplotlib, and Seaborn.

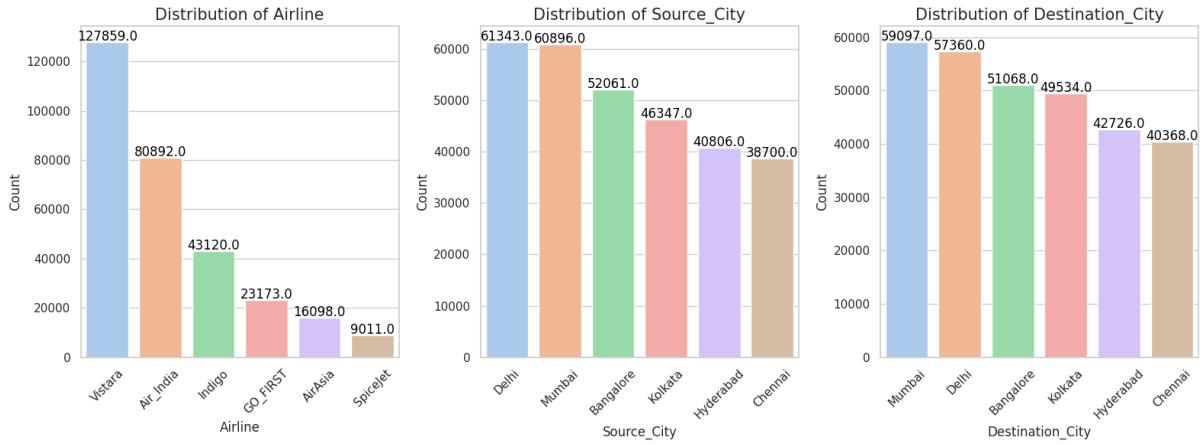

Figure 3: Data Visualization of Categorical Columns - 1

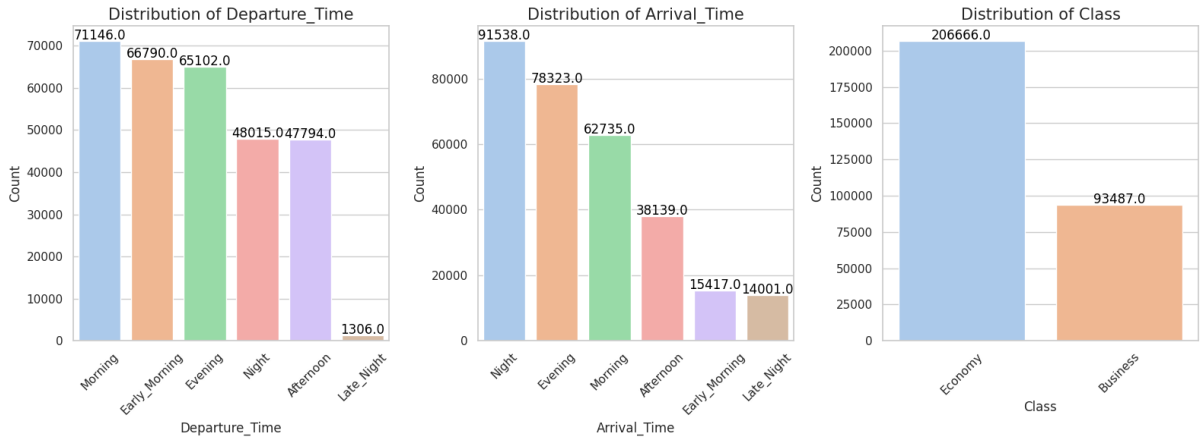

Figure 4: Data Visualization of Categorical Columns - 2

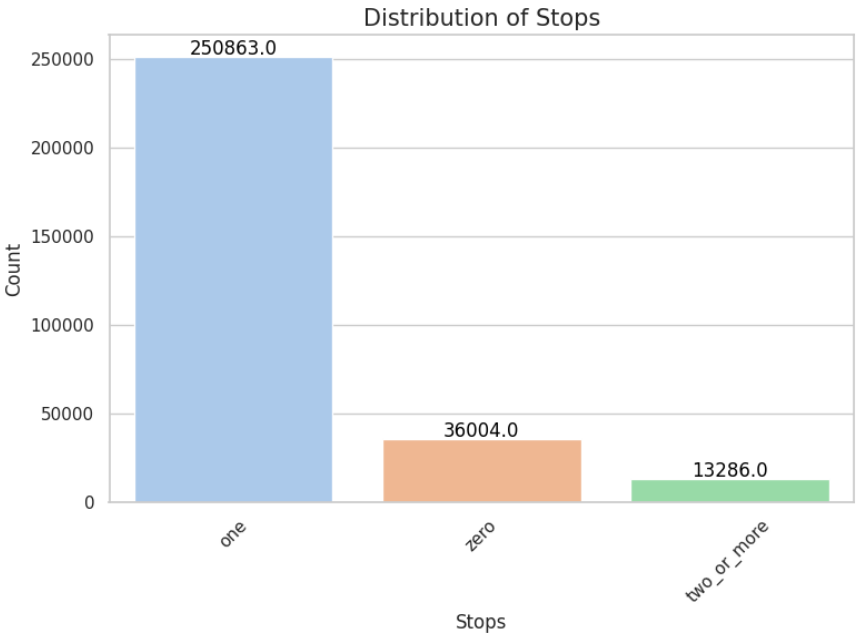

Figure 5: Data Visualization of Categorical Columns - 3

From the distribution of airlines, it can be seen that this data has many airlines, and since the prices of the flights are affected by this, the airlines have to be considered in the model. Distribution of the departure and arrival cities and the number of different cities show that this data is very diverse and represents more of a general case rather than a specific city. Distribution of departure and arrival times are realistic, which further validates the data. Similar to the distribution of airlines, both the class type (business or economy) data and the number of stops further complicate the data and add more features to consider in the model implementation process.

For numerical features, we received the following plottings:

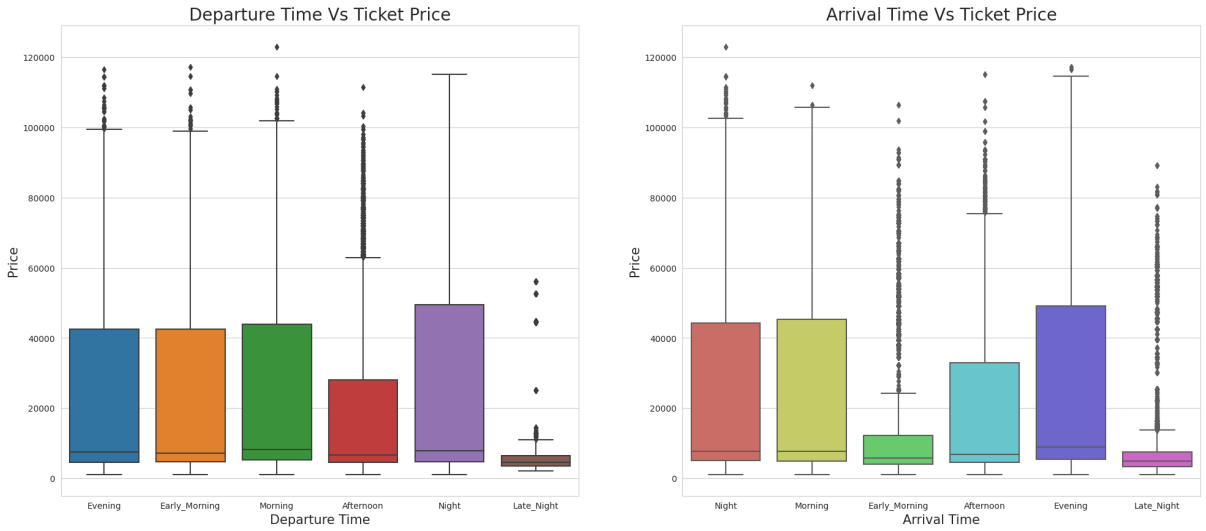

Figure 6: Arrival & Departure Time Relations to Price

The departure time versus ticket price graph is very sensible as the higher demand for a timeframe results in a higher price in real life. Late night is the least wanted time frame for a flight, resulting in the lowest price. Similarly, the afternoon time frame has lower demand as people get into flights either at the beginning of the day or the end. Night time frame has the highest price as that time frame is usually the time that the businessmen get into flights. These prices have similarly been reflected in the arrival time versus ticket price graph, as a flight usually takes a one-time frame.

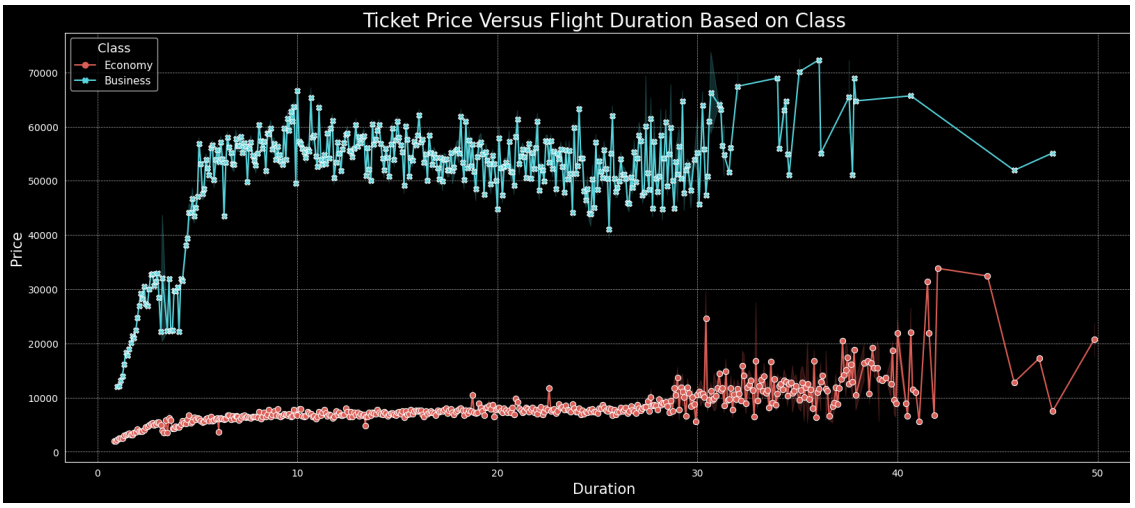

Figure 7: Ticket price versus flight duration for each class

Business class flights have higher prices compared to economy class, which can be seen from this graph. In addition to that, longer flights have higher ticket prices, which can be seen from this graph as well. The reason that the graph does not have a monotonically increasing trend comes from the fact that for some flight durations, there is not that much data, resulting in fringe cases changing the average a lot. In addition to that, there are many other features that affect plane ticket prices.

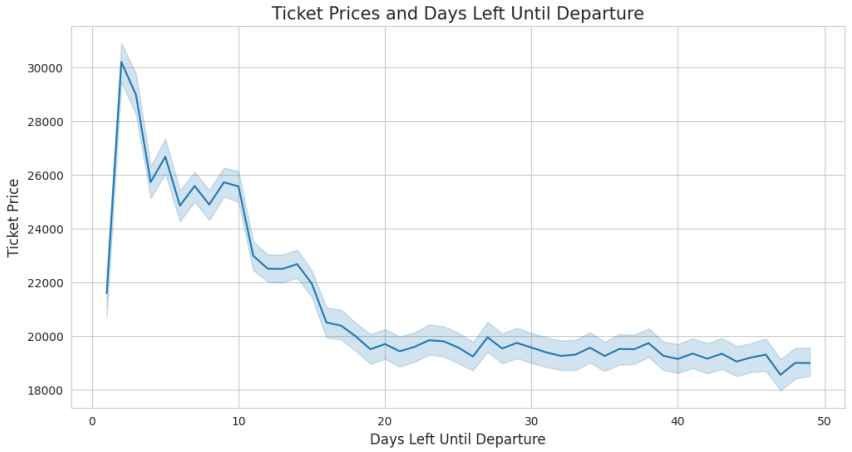

Figure 8: Ticket prices versus days until departure

Ticket prices decrease if they are bought early unless there is a single day or less remaining. If the plane is not fully booked when there is little time remaining for the flight, it indicates less demand for these flights.

After understanding the data and coming up with these plots, a correlation matrix of each feature was made to understand whether features are needed. At first, preprocessing had to be done where “one-hot encoding” was used. This method basically gives numbers to the feature attributes for non-numerical features by assigning each attribute with nonnegative integers. This was done in order to deal with nonnumerical features easily. The correlation matrix can be seen below.

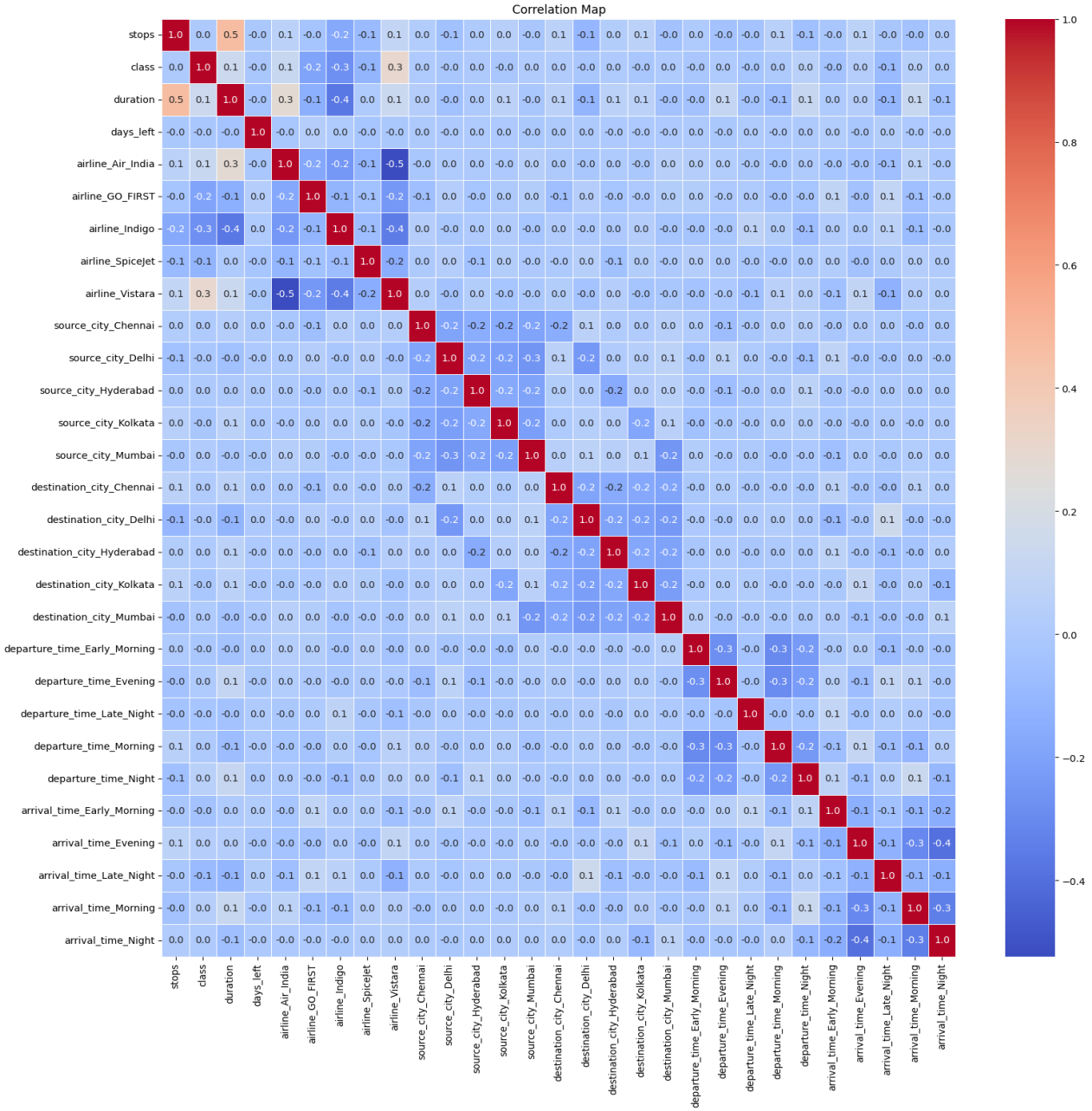

Figure 9: Correlation Matrix

The correlations between any features are quite weak, suggesting that none of these variables strongly predict the others. This could mean that a combination of factors influences the pricing strategy for flights and cannot be accurately predicted by flight duration or booking time alone. This means that non-linear relationships or interactions between these and other variables could better explain the variance in flight prices.

Model Implementation

1. Linear Regression

We first tried the linear regression method, as it is easy to understand and it clearly shows the relationship between features and output. Linear regression is a statistical method for predicting the connection between two variables. The goal is to identify the line that best fits the relationship between the independent and dependent variables, assuming that there is a linear relationship between them. The line is found by minimizing the total squared errors between the expected and actual numbers [2].

Linear Regression formula is given as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$$

Where 'y' denotes the dependent variable we aim to predict, 'β_0' is the y-intercept representing the starting value of 'y' when all independent variables are zero, and the coefficients indicate the weight of each independent variable, and 'ε' is the error term.

The Linear regression model was trained with our train dataset and tested on the test dataset, and prediction results were obtained. Predictions and true values were compared using different methods, and the success of the model was evaluated. We used several metrics to gauge the performance of our Linear Regression model:

- **Mean Absolute Error (MAE):** Reflecting the average absolute difference between the predicted and actual values, thus providing a straightforward measure of prediction accuracy.
- **Mean Squared Error (MSE):** This represents the average of the squared differences between predicted and actual values, highlighting the model's accuracy in terms of the magnitude of errors.
- **Root Mean Squared Error (RMSE):** Offering a more interpretable version of MSE by scaling the errors back to their original units.
- **Root Mean Squared Log Error (RMSLE):** An alternative to RMSE that is particularly useful when dealing with exponential or log-normal distributions of errors.
- **R^2 Score:** Signifying the proportion of the variance in the dependent variable that is predictable from the independent variables, thus measuring the goodness of fit.
- **Adjusted R Square:** This adjusts the R^2 to account for the number of predictors in the model, providing a more accurate assessment of the model's explanatory power.

The results for linear regression are given below.

Model name: Linear Regression

Mean Absolute Error (MAE): 4618.407

Mean Squared Error (MSE): 48705986.75

Root Mean Squared Error (RMSE): 6978.967

Root Mean Squared Log Error (RMSLE): 8.851

R2_score: 0.905576

Adj R Square: 0.905573

2. Lasso Regularized Linear Regression

Although we have gotten good results from Linear Regression, we have also decided to try lasso regularized linear regression to get better results. Lasso regression is a shrinkage-based kind of linear regression. Data values that shrink toward a mean or other center point are referred to as shrinkage. Simple, sparse models, models with fewer parameters, are encouraged by the lasso process [3].

L1 regularization, which involves adding a penalty equal to the absolute value of the magnitude of the coefficients, is carried out by Lasso regression. This kind of regularization may produce a sparse model with few coefficients; some coefficients may become zero and be removed from the model. Larger penalties result in coefficient values closer to zero, which is good for developing simpler models[3].

The formula for the Lasso Regularized Linear Regression is given as follows:

$$\sum_{i=1}^{n} \left(y_i - \sum_j x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \left|\beta_j\right|$$

Where y denotes the real values, x denotes the features, and β denotes the coefficients of features in prediction. In this equation, λ is a tuning parameter that controls the strength of the penalty and, as a result, shrinkage. Therefore, before doing lasso regularized linear regression, it was necessary to determine what the λ value would be. A validation dataset was used for this. The λ that gave the best result on the validation dataset was selected.

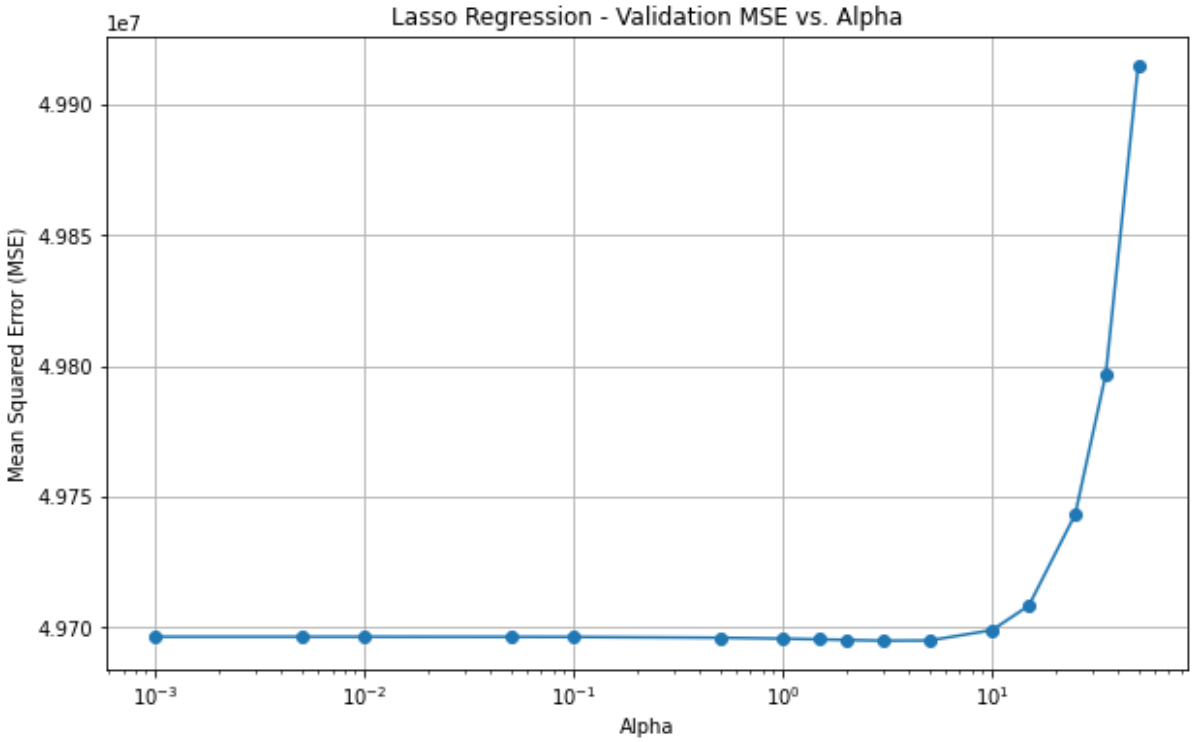

Figure 10: MSE vs Alpha for Lasso Regression - Validation 1

After the first validation, it was noticed that the best results obtained when λ were between 3 and 5. So, the validation has been done a second time in order to get more precise results.

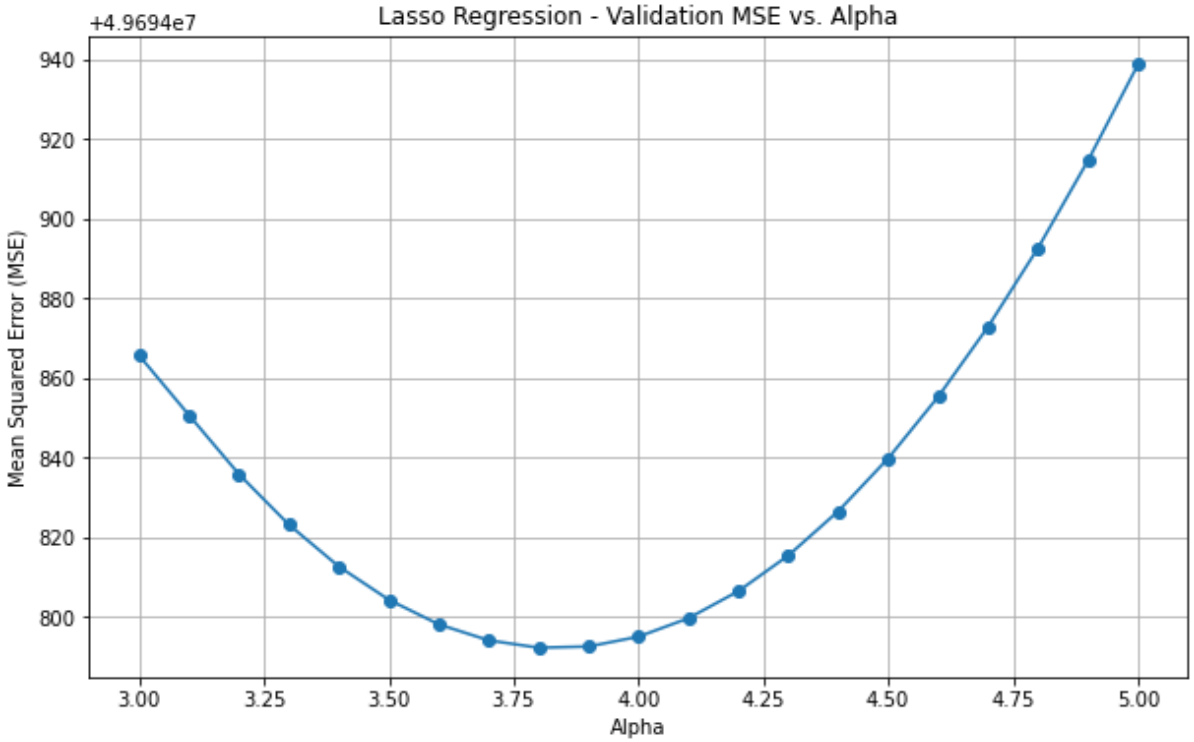

Figure 11: MSE vs Alpha for Lasso Regression - Validation 2

As can be seen from the graph, the best results were obtained when λ was equal to 3.8. It is decided to make lasso regularized linear regression with λ =3.8.

After selecting λ , it was tested on the test dataset. Predictions and true values were compared using the same methods as Linear Regression, and the success of the model was evaluated. We used the same metrics to gauge the performance of the model.

The results for Lasso regression are given below.

Model name: Lasso

Mean Absolute Error (MAE): 4614.248

Mean Squared Error (MSE): 48703631.263

Root Mean Squared Error (RMSE): 6978.799

Root Mean Squared Log Error (RMSLE): 8.851

R2_score: 0.905581

Adj R Square: 0.905578

The results were very similar to linear regression since the dataset's features are not correlated. There is no need to make sparse solutions. However, the prediction is more efficient in lasso regression.

3. Random Forest Regression

Since Linear regression and Lasso regularized linear regression gave very similar results, we used the random forest method, which we thought could achieve better results. With the use of several decision trees and a method known as Bootstrap and Aggregation, or bagging, Random Forest is an ensemble methodology that can handle both regression and classification tasks. The fundamental idea here is to use a combination of decision trees instead of depending only on one to determine the final result [4].

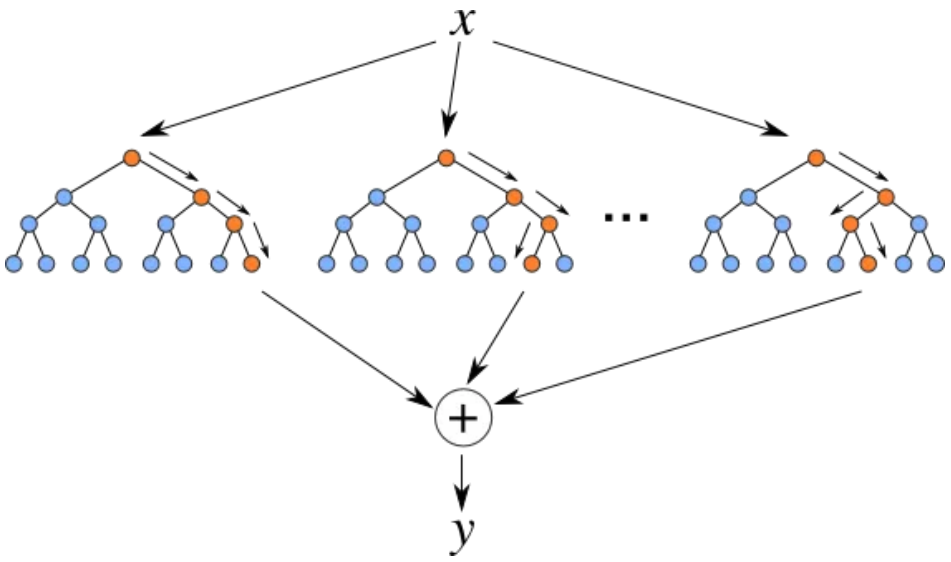

Figure 12: Decision Tree for Random Forest Regression [5]

In order to get better results in the most efficient way in the random forest method, the validation dataset was used to determine the number of trees.

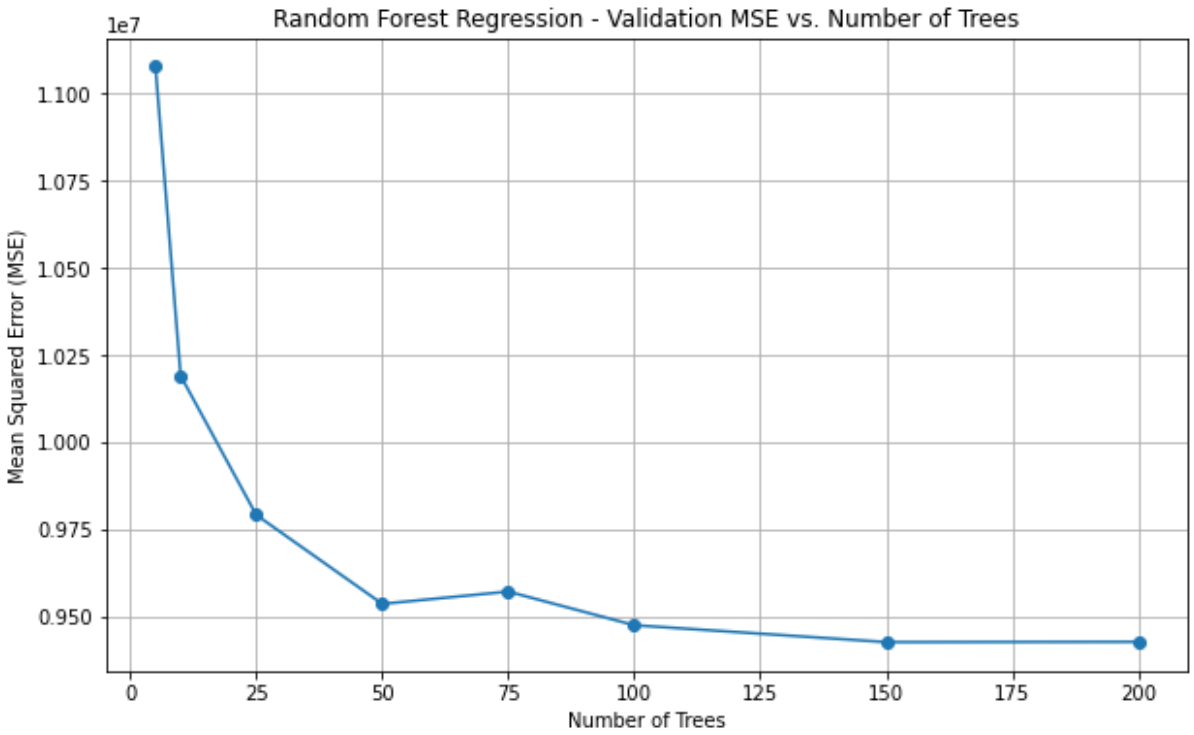

Figure 13: RFR Validation MSE vs Number of Trees

As can be seen from the plot, the error did not change that much after the 150 trees in the forest, so it was decided that the number of trees should be 150.

After selecting the number of trees, the trained model was tested on the test dataset. Predictions and true values were compared using the same methods as before, and the success of the model was evaluated. We used the same metrics to gauge the performance of the model.

The results for the random forest are given below.

Model name: Random Forest

Mean Absolute Error (MAE): 1137.558

Mean Squared Error (MSE): 7837179.929

Root Mean Squared Error (RMSE): 2799.496

Root Mean Squared Log Error (RMSLE): 7.937

R2_score: 0.984806

Adj R Square: 0.984805

As one can see from the results, random forest outperforms the other models, and this could have happened for 3 different reasons:

1. There can be non-linear relationships between features and price, and linear regression or lasso regression are not able to capture this non-linearity. However, Random Forest can also capture complex, nonlinear relationships.
2. Linear regression and Lasso regression assume that features are independent of each other. Although our dataset's correlation between the features is low, it may create small differences. Random Forest considers these correlations and can learn according to that.
3. Data must be numerical for linear and lasso regression, so categorical data must be preprocessed. On the other hand, Random Forest can handle categorical data without needing one-hot encoding.

Model	MAE	MSE	RMSE	R2 Score	RMSLE	MAPE	Adj R Square
LinearRegression()	4618.41	4.8706e+07	6978.97	0.905576	8.851	43.68	0.905573
Lasso(alpha=3.8)	4614.25	4.87036e+07	6978.8	0.905581	8.851	43.58	0.905578
RandomForestRegressor(n_estimators=150)	1137.56	7.83718e+06	2799.5	0.984806	7.937	7.66	0.984805

Table 1: General Models and Evaluation Results

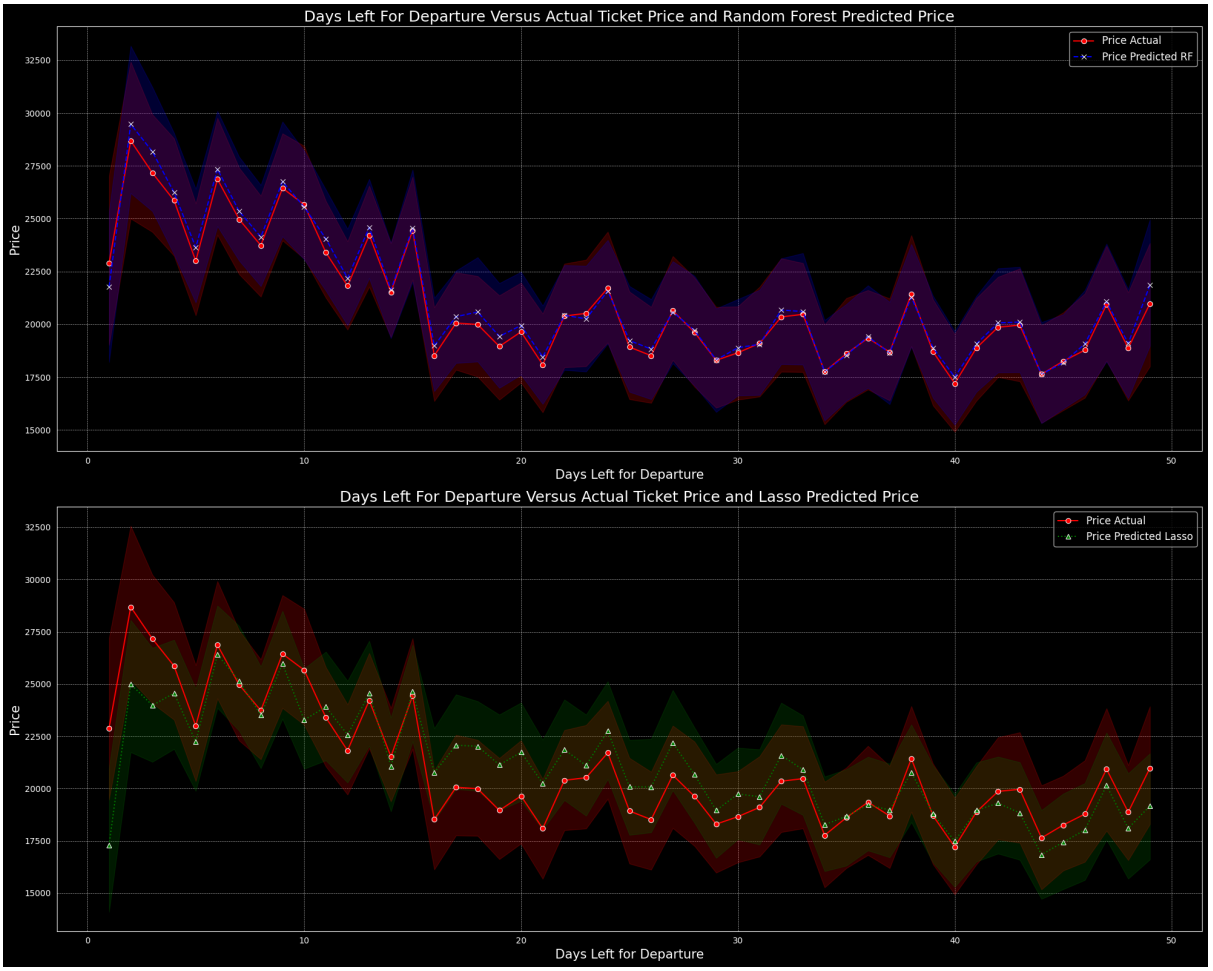

Figure 14: Actual vs Predicted Price with Days Left Features for Lasso and Random Forest

The Random Forest model closely tracks the actual price fluctuations, as indicated by the overlapping red and purple lines, though it shows periods of significant deviation and presents wider prediction intervals suggesting higher uncertainty. The Lasso regression model, represented by the green line, appears to be more conservative, failing to capture the peaks and troughs of the actual prices with smaller prediction intervals, indicating a more stable but less sensitive model. As a result, the Random Forest model demonstrates a better fit to the actual data, albeit with greater variability in its predictions, while the Lasso model underperforms in tracking price movements but maintains a steadier predictive range.

Coding Environment

We used Python 3 for this project's detailed data analysis and model implementation. We developed our Python codes in the Google Colab service with the default Python 3 CPU running environment.

We used the Pandas framework for dataset manipulations, which are needed in the detailed data analysis section. Pandas is a well-known library in the Data Science domain due to its powerful and easy-to-use applications in Python [6].

For extensive visualization of the features in our project, we have used Matplotlib and Seaborn libraries. Matplotlib is a widely acknowledged library for various plottings, mostly used for bar charts, line charts, histograms, heatmaps, and whisker plots in our project. To show our plottings aesthetically, we have used the Seaborn library, built on top of Matplotlib, to ensure visualizations are aesthetic and statistically stable [7].

For preprocessing the data, which is mentioned in the “Data Analysis & Preprocessing” section in detail, we used the Pandas framework for its powerful applications in the matrix [6].

We used the sci-kit-learn (sklearn) library for model development, a popular and efficient open-source machine-learning library [8]. To partition our dataset, we employed sklearn.model_selection’s train_test_split class to ensure a distribution of 80% for training and 10% each for test and validation sets through a two-step split process. We utilized the MinMaxScaler class from sklearn.preprocessing to normalize our numerical data, particularly integer values. Our modeling approach included the implementation of a Linear Regression model and Lasso Normalization using sklearn.linear_model, as well as a Random Forest model with sklearn.ensemble. Lastly, for the evaluation of our models, we have used various metrics from sklearn.metrics, complementing our analysis with graphical representations of the results using Seaborn and Matplotlib for data visualization.

Our models mentioned in this report are relatively easier to run in the Google Colab free tier, and the longest cell run took 10 minutes. However, in the future work that we will mention in the “Scheduled Tasks” section, we will use Neural Networks architectures to develop the models. For these situations, we will need more GPU. To prevent this obstacle, one of our group members (Emre) has a Colab Pro account, and we will use a Tesla T4 GPU server with extensive RAM (+50 GB) if needed.

Accomplished Tasks

1. Exploratory Data Analysis (EDA)
    a) Overview of dataset
    b) Checking missing values and data consistency
    c) Data visualization
    d) Preprocessing
    e) Correlation matrix
2. Model Validation
    a) Lasso Regression
    b) Random Forest Regressor
3. Model Implementation
    a) Linear Regression
    b) Lasso Regression
    c) Random Forest Regressor
4. Model Evaluation
5. Interpretation of current results

Scheduled Tasks

1) More regularization techniques can be implemented to improve the results.

2) Feed Forward Neural Network deep learning method will be implemented.

Workload Distribution

Data Analysis	All of the team members
Preprocessing	Mustafa Kütükcü
Linear Regression Implementation	Ege Berkay Karagenç
Lasso Regression Validation	Emre Karataş, Ege Berkay Karagenç
Lasso Regression Implementation	Erim Berke Salman
Random Forest Validation	Erim Berke Salman, Gökay Özbay

Random Forest Implementation	Mustafa Kütükcü
Model Evaluation	Gökay Özbay, Emre Karataş
Error Analysis	All of the team members

Table 2: Workload Distribution Between Group Members

References

[1] S. Bathwal, “Flight price prediction,” Kaggle, https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction/data (accessed Oct. 16, 2023).

[2] K. Mali, “Everything You Need to Know About Linear Regression!,” Analytics Vidhya,https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/ (accessed Nov. 25, 2023).

[3] S. Glen, “Lasso Regression: Simple Definition,” Statistics How To, https://www.statisticshowto.com/lasso-regression/ (accessed Nov. 25, 2023).

[4] A. Dutta, “Random Forest Regression in Python,”

GeeksforGeeks,https://www.geeksforgeeks.org/random-forest-regression-in-python/

(accessed Nov. 25, 2023).

[5] Chaya, “Random Forest Regression,” Medium,

https://levelup.gitconnected.com/random-forest-regression-209c0f354c84 (accessed Nov.

25, 2023).

[6] pandas, “About pandas,” pandas, https://pandas.pydata.org/about/ (accessed Nov. 25, 2023).

[7] S. Pierre, “Python data visualization with Seaborn and Matplotlib,” Built In, https://builtin.com/data-science/data-visualization-tutorial (accessed Nov. 25, 2023).

[8] scikit, “scikit-learn,” scikit, https://scikit-learn.org/stable/ (accessed Nov. 25, 2023).