# Flight

# Fare

# Prediction

**Group Name: G**

**Group Members:**

| First name | Last Name | Student number |
|---|---|---|
| Dev | Makwana | C0885064 |
| Krina | Patel | C0886861 |
| Mahaveersinh | Chauhan | C0884854 |
| Trushna | Patel | C0886910 |

**Submission date: 15/08/2023**

# Table of Contents

# Abstract

Flight fare prediction is a critical task in the airline industry, offering valuable insights for travelers and airlines alike. The use of machine learning regression techniques, such as Random Forest, Decision Tree, XGBoost, and AdaBoost, to forecast airline prices is examined in this paper. To get the best accuracy, the XGBoost regressor and Random Forest regressor and hyperparameter optimization are the main concerns. As part of the technique, the flight fare dataset is examined and preprocessed to manage missing values, categorical features are encoded, and uniform scaling is ensured. For correct evaluation, the dataset is divided into training and testing sets. For the Random Forest regressor the study focuses on hyperparameter optimization. The number of estimators, the maximum depth, and the minimum samples per leaf are among the parameters that can be optimized using strategies like RandomizedSearchCV. The similar parameters along with learning rate and subsamples. The study attempts to attain the highest level of accuracy for airline fare prediction by fine-tuning these factors. The findings demonstrate that XGBoost produces the most precise predictions of airline fares when its hyperparameters are appropriately calibrated among the algorithms examined. Optimizing model configurations is crucial since hyperparameter adjustment considerably improves model accuracy. This study contributes to travel forecasting by giving travelers and airlines helpful information to use when making decisions.

# Introduction

The ability to predict airline fares is an essential component of the contemporary travel scene, influencing choices made by both travelers and carriers. The research of cutting-edge data science and machine learning approaches has been prompted by the dynamic and ever-changing nature of airline ticket prices to create precise predictive models. These models allow airlines to maximize revenue and pricing strategies and enable travelers to make educated decisions. Analysis of airline costs looks at variables influencing the cost of airline tickets. Booking timelines, seasonality, day/time of travel, route distance, stops, carrier reputation, economic circumstances, and competitiveness are important factors. The data considered in this project is based on the Flight Fare Prediction (Flight Fare Prediction MH, n.d.) . Algorithms for dynamic pricing change prices in response to current conditions. Such research informs bargain-hunting customers and aids airlines in streamlining their revenue-generating plans.

The airline sector is influenced by various factors, including departure and arrival cities, route length, stops, and airline company. Market demand, economic changes, competition, and supply dynamics also impact airfare. Analytical methods are needed to identify trends and predict future costs. To overcome this difficulty, machine learning has emerged as a disruptive force. By simulating the complex interplay between features that influence pricing, regression algorithms can be used to forecast flight prices. These algorithms offer a toolkit for capturing nonlinear relationships, handling noisy data, and adapting to various dataset types. They range from decision trees to ensemble methods like Random Forest, gradient boosting techniques like XGBoost, and adaptive boosting using AdaBoost.

Precise flight fare prediction benefits travelers, airlines, and the economy by enabling informed plans, enhancing revenue management, and offering intelligent pricing choices. This study investigates the complexities of predicting airline fares using data science and machine learning. We aim to uncover the patterns underlying flight fares and contribute to the development of prediction models in the aviation industry by examining historical flight data, using a variety of regression algorithms, and utilizing the power of hyperparameter tuning. The knowledge gleaned from this endeavor has the potential to revolutionize how consumers engage with airfare information and to equip airlines with the tools they need to prosper in a competitive and changing industry.

# Methodology

The goal of this project is to analyze the Airline data to predict the price of the flight fare based on certain parameters such as Airline, Source, Destination, Route, Stops, and so on. The basic steps implemented while developing this project are:

## 1. Dataset details

The process begins with collecting data from Kaggle (*Flight Fare Prediction MH*, n.d.) and understanding the details about data on the significance of each feature in the data and how to utilize them for predicting price.

| **Features** | **Description** |
|---|---|
| Airline | The airline's name. |
| Date_of_Journey | The date of the journey |
| Source | The source from which the service begins. |
| Destination | The destination where the service ends. |
| Route | The route taken by the flight to reach the destination. |
| Dep_Time | The time when the journey starts from the source. |
| Arrival_Time | Time of arrival at the destination. |
| Duration | Total duration of the flight. |
| Total_Stops | Total stops between the source and destination. |
| Additional_Info | Additional information about the flight |
| Price | The price of the ticket |

## 2. Analysis Process

### I. Data Cleaning

The dataset from Kaggle (*Flight Fare Prediction MH*, n.d.) contains a few null values that are required to handle for better results. Thus, there are further cleaning steps were carried out to make the data available for further processing, such as removing missing values. The missing values can be imputed with some meaningful value, but there were only two null values in Total_stops and Route, one in each column. Thus, they were dropped from the further process. In addition, there were approximately 220 duplicate values that were ignored.

## II. Exploratory Data Analysis

There is various plotting implemented to analyze datasets to explore hidden patterns in brief and answer certain questions that do not require modeling.

1) **Average Price by Airline:** This was plotted to check the average price range according to the type of airline. From the bar plot, it was concluded that the average price of Jet Airways Business is extremely high, which might be due to business class data. In addition, Air India, Jet Airways, Multiple Carriers, and Multiple Carriers Premium Economy follows a similar average trend of approximately 11K.


Figure 1 Average Price by Airline

2) **Price Distribution over Airlines:** As per the trend visualized in the above plot, the prices of Jet Airways will be extremely high while others. There are, however, some extreme values in almost every airline that seems to be an outlier but the distribution follows the average price trend.


Figure 2 Price Distribution over Airlines

3) **Average Price of all Airlines by Stops:** This stacked bar plot below depicts that the 1-stop and 2-stops flights are more expensive comparatively than the others. Additionally,

Jet Airways Business can be seen major contribution in them due to which the hype is observed. Also, only Air India flights have the choice of 4-stops and 3-stops have another contributor, Multiple Carriers.

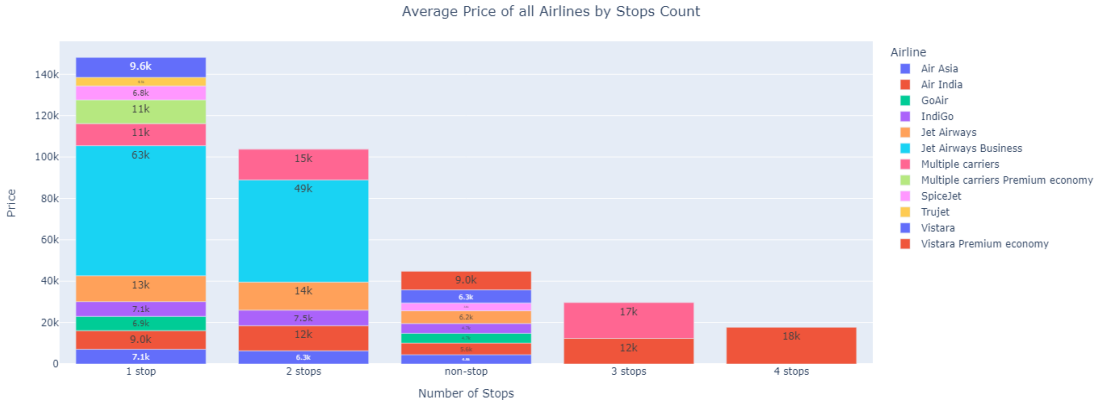


Figure 3 Average Price of all Airlines by Stops

4) **Impact of Source City on Price:** The pie chart below illustrates that if the departure city is Delhi, Kolkata, or Banglore than they have considerably higher prices amongst which, Delhi being source has the maximum mean fare.

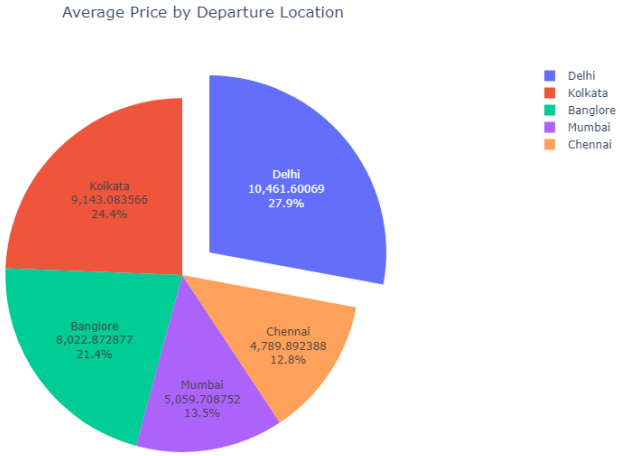


Figure 4 Impact of Source City on Price

# III. Feature Extraction

- The features have some intrinsic special properties that can be used for data analysis, like the day of the journey, whether it was a weekday or not, and the month as winter months are more affected by the fare price.

- By transforming categorical data to dummy variables, the Pandas get_dummies (“Python Pandas - Get_dummies() Method,” 2020) method provides unique columns

for each categorical data type. As the Airline column has 12 unique values containing the airline's name, which are categorical values. Those categorical values must be converted into numerical values using the One-Hot Encoding technique. Two other categorical columns that are source and destination, were also encoded to get equivalent numerical values.

- Use the Total_stops column and convert the categorical values like nonstops, 1-stops, and so on with numerical values.
- **Feature Correlation: It** was concluded that total stops and durations hours are highly correlated to the flight price. The relationship is plotted after cleaning and feature engineering; hence, it contains many features.

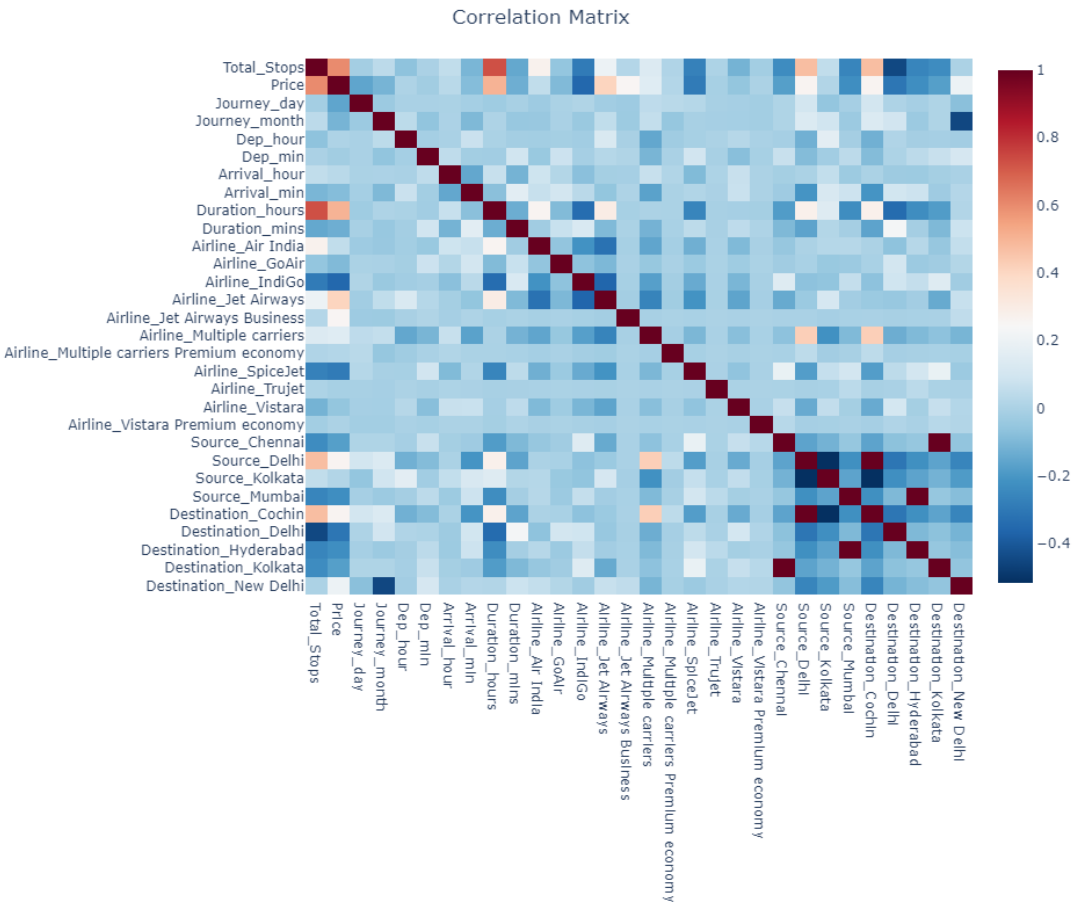


Figure 5 Correlation Matrix

## IV. Feature Selection

**Extra Trees Regressor:** Extremely Randomized Trees Regressor, also known as Extra Trees Regressor (*ML | Extra Tree Classifier for Feature Selection - GeeksforGeeks*, 2019), is a decision tree-based machine learning method that is a member of the ensemble learning

family. Extra Trees Regressor is a tool for feature selection and regression tasks. Extra Trees Regressor calculates the importance of each feature by measuring how much the Mean Squared Error (MSE) or another appropriate loss function decreases when a particular feature is used for a split. Features that contribute more to reducing the variance in predictions across the ensemble are considered more important. We have selected important features according to the result of ExtraTreesRegressor with nlargest=20. The experiment resulted in providing 20 features that were important like total stops, journey day, duration of flight, and type of airline.

## V. Machine Learning Model

**Random Forest:** Random Forest is a Machine Learning approach that combines different decision trees to generate predictions. A subset of features and data points are randomly selected for each decision tree's training, and the average or majority of forecasts are then used ("Random Forest," 2023). This raises the model's accuracy and reduces the likelihood of overfitting. We picked the random forest to predict the Supreme Court rulings since it is a well-liked model that considers the facts of the cases, the legal issues, the decisions made by lower courts, and the style of decision.

**XGBoost:** An enhanced gradient boosting algorithm is XGBoost Regressor (*XGBoost for Regression - GeeksforGeeks*, 2020). It sequentially generates decision trees to reduce the gradient of the loss function. XGBoost includes regularisation terms to limit model complexity and avoid overfitting. It also allows parallel computation for quicker training and uses a complex tree-pruning method.

**Decision Tree Regressor:** A predictive model called a decision tree regression employs a tree-like structure to make predictions. To reduce the variance of the target variable within each subset, it divides the data into subsets based on the feature values. From the root of the tree to a leaf node, predictions are produced, with the predicted value being the meaning of the target values in that leaf.

**K-Nearest Neighbors (KNN) Regressor:** By calculating the average of the target values of its K nearest neighbors in the feature space, the KNN Regressor (Singh, 2018) predicts the target value of a new data point. It determines the separation between data points and chooses the K nearest points to use as prediction points. The value of K and the distance metric can affect how well a KNN Regressor performs.

**AdaBoost Regressor:** AdaBoost Regressor (*Sklearn.Ensemble.AdaBoostRegressor*, n.d.) is an ensemble technique that builds a strong model out of several weak learners (often decision trees). In each iteration, it prioritizes samples that were incorrectly categorized and gives various weights to the data points, thereby enhancing the model's performance. A weighted average of the guesses made by several students makes up the final prediction.

## VI. Hyper-parameter Tuning

The scikit-learn library in Python provides a function called RandomizedSearchCV that is used for cross-validation-based hyperparameter tuning and model selection. It is especially made to quickly browse various hyperparameters and determine the ideal combination for a machine learning model. Before training a model, hyperparameters are established, impacting the model's performance and behavior.

The best model that gives appropriate approximation is the XGBoost regressor, which is why it was decided to tune its parameters to improve the prediction score. The hyper-parameters that can be tried are n_extimators, sub-samples, max_depth, and learning rate, resulting in improved prediction.

Another better-performing model is RandomForestRegressor, for which the hyper-parameters could be things like the number of trees in the forest, the deepest a tree can grow, the bare minimum of samples needed to divide an internal node, etc. After performing hyper-parameter tuning on the Random Forest algorithm, there was an improvement in the score.

# Results

The Machine Learning Models listed in the previous chapter were implemented to analyze the results on the decided dataset using the measure of Training and Testing scores.

| Machine Learning Model | Training Score | Testing Score |
|---|---|---|
| DecisionTreeRegressor | 0.96 | 0.71 |
| RandomForestRegressor | 0.95 | 0.81 |
| KNeighborsRegressor | 0.73 | 0.54 |
| AdaBoostRegressor | 0.44 | 0.35 |
| XGBRegressor | 0.93 | 0.83 |

Based on the observations listed in the above table, some conclusions were made such as while Random Forest, Decision Tree, and XGBoost algorithms were the best performing, SVR seems to be the worst in this case. In this experiment of implementing six different techniques on our dataset, the performance of trees-based algorithms outweighs the performance of the rest of the algorithms. Since there was the least difference between training and testing scores of XGBRegressor that resulted in 0.83 while the Random Forest was also performing well, thus it was tuned the hyper-parameters of Random Forest to improve the score. After tuning hyper-parameters, the score of XGBoost and Random Forest was observed to be 0.84 and 0.82, respectively. The figure on the left shows the predicted values against actual values for the Hyper-parameter tuned XGBoost and the right-side figure shows the same plot for the Hyper-parameter tuned Random Forest Regressor.

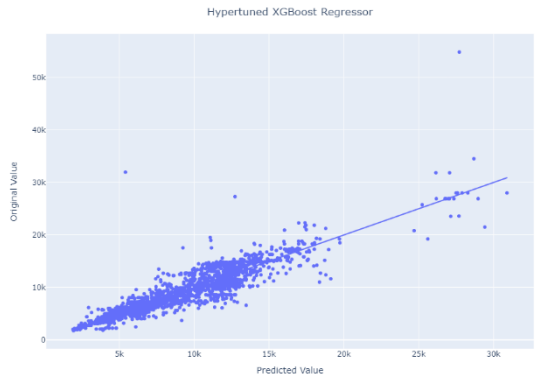


Figure 6 Hyper-parameter tuned Random Forest result

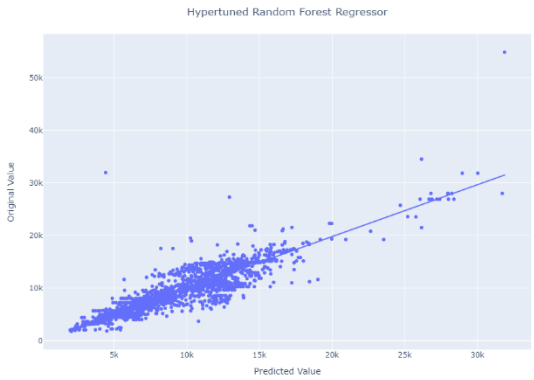


Figure 7 Hyper-parameter tuned XGBoost result

# Conclusion and Future Work

In the world of contemporary air travel, the goal of roughly estimating airline fares in advance has enormous importance. This practice improves financial readiness, flexibility, and the entire travel experience by enabling travelers to anticipate and prepare for fare variations. Airlines stand to gain from greater customer happiness, enhanced revenue management, and the potential to create innovative marketing efforts. The nexus of precise pricing forecasts and proactive decision-making highlights the mutually beneficial relationship between passengers looking for affordable options and airlines aiming for optimal revenue optimization. Approximating flight fares in advance emerges as a crucial step towards a more open, customer-focused, and economically effective travel environment as technology and data-driven insights continue to transform the aviation business.

In future work, one of the things that can be done for the same data can be inter-airline price competition and real-time pricing insights.

# References


*Flight Fare Prediction MH*. (n.d.). Flight Fare Prediction MH | Kaggle. https:///datasets/nikhilmittal/flight-fare-prediction-mh

Python Pandas—Get_dummies() method. (2020, July 28). *GeeksforGeeks*. https://www.geeksforgeeks.org/python-pandas-get_dummies-method/

Random forest. (2023). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=1160091469

*XGBoost for Regression - GeeksforGeeks*. (2020, August 29). GeeksforGeeks. https://www.geeksforgeeks.org/xgboost-for-regression/

Singh, A. (2018, August 22). *KNN algorithm: Introduction to K-Nearest Neighbors Algorithm for Regression*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

*sklearn.ensemble.AdaBoostRegressor*. (n.d.). Scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html

*ML | Extra Tree Classifier for Feature Selection - GeeksforGeeks*. (2019, July 21). GeeksforGeeks. https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/