

**BATCH – 20**

**RegressionAI- A Flask Application for  
Regression Model Training and Data  
Analysis**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**(Artificial Intelligence & Machine Learning)**

Submitted by

<b>N. Sanjeev Reddy</b>	<b>(322103382042)</b>
<b>P. Lalith</b>	<b>(322103382049)</b>
<b>Tarakanta Acharya</b>	<b>(32310338258)</b>



**COLLEGE OF ENGINEERING**  
**(AUTONOMOUS)**

Department of Computer Science and Engineering (AI & ML)

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)**

Affiliated to Andhra University, A.P, Accredited by NBA & NAAC

**MADHURADWADA, VISAKHAPATNAM-530048**

# Abstract

This project delves into the exploration and application of machine learning techniques for predictive modeling, with a focus on regression algorithms. The primary goal is to analyze a given dataset, understand its underlying trends, and build models to accurately predict the target variable. The process begins with comprehensive data exploration, including visualization and statistical analysis, to uncover patterns, relationships, and anomalies. This step ensures that the data is well-prepared for subsequent machine learning tasks, with preprocessing steps such as handling missing values, feature scaling, and encoding categorical variables.

A variety of regression models are implemented to cater to diverse dataset characteristics. These models include Linear Regression, Polynomial Regression, Lasso, Ridge, k-Nearest Neighbors (kNN), Support Vector Regression (SVR), Random Forest, Decision Tree, Quantile Regression, and Isotonic Regression. Each algorithm is rigorously trained and tested, with performance evaluation based on metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE),  $R^2$  Score, and Mean Absolute Percentage Error (MAPE). The models are further analyzed to determine their efficacy across datasets of varying sizes and complexities, providing insights into their suitability for different use cases.

To enhance prediction accuracy and robustness, ensembling techniques are employed, combining the strengths of individual models to create a more reliable predictive framework. The project emphasizes the importance of justifying model selection based on empirical results, ensuring that the chosen approach aligns with the dataset's characteristics and prediction goals.

By integrating data exploration, regression modeling, and ensembling methods into a cohesive workflow, this project serves as a comprehensive platform for understanding and applying machine learning techniques. It not only highlights the strengths and limitations of various algorithms but also provides actionable insights for selecting the most effective approach for predictive tasks, making it a valuable tool for data scientists and machine learning practitioners.

# Index

<b>S.No.</b>	<b>Title</b>	<b>Page No.</b>
1.	Introduction	4
2.	Problem Statement	5
3.	Objectives	6
4.	Libraries Used	7-8
5.	Data Description	9-10
6.	Algorithms Used	11
7.	Algorithm Description	12-16
8.	Ensembling	17-20
10.	Implementation	21-22
11.	Metrics	23-26
12.	Future Scope	27-28
13.	Web References	29

# Introduction

The aviation industry is a dynamic and highly competitive sector, where flight prices fluctuate based on a multitude of factors such as demand-supply dynamics, seasonal trends, fuel costs, airline competition, and external economic conditions. Accurately predicting flight prices is a critical challenge for travelers, airlines, and businesses, as it directly impacts decision-making, budgeting, and revenue optimization. This project aims to address the challenge of modeling and forecasting flight prices using advanced machine learning techniques.

Flight price data is inherently complex, with patterns influenced by a wide range of variables, including booking time, travel dates, route popularity, airline pricing strategies, and external events. Developing a reliable predictive model requires not only the ability to handle large and diverse datasets but also the capacity to capture the intricate relationships and trends within the data. Machine learning offers a powerful set of tools for this purpose, enabling the development of sophisticated models that adapt to the complexity of real-world flight price dynamics.

The flight price dataset considered in this project includes temporal, contextual, and transactional information that reflects pricing trends across various routes, airlines, and timeframes. By analyzing this data, the project aims to explore patterns, detect anomalies, and build predictive models that can forecast flight prices with high accuracy. The insights gained from this analysis are vital for stakeholders, including travelers, airlines, and businesses, to make informed decisions and optimize their strategies.

This project goes beyond simple prediction by providing a comprehensive exploration of the dataset, evaluating multiple regression algorithms, and employing ensembling techniques to enhance model performance. The approach ensures that the solution is robust, scalable, and capable of addressing the complexities of flight price prediction effectively. By leveraging advanced machine learning techniques, this project aims to revolutionize the way flight prices are predicted, benefiting all stakeholders in the aviation industry.

# Problem Statement

The aviation industry is characterized by dynamic and fluctuating flight prices, driven by factors such as demand-supply imbalances, seasonal trends, fuel costs, and competitive pricing strategies. Predicting flight prices accurately is a critical challenge for travelers, airlines, and businesses alike. For travelers, it enables cost-effective planning and booking decisions, while for airlines, it supports optimized pricing strategies and revenue management. However, the complexity and variability of flight price data, influenced by numerous interdependent factors, make traditional forecasting methods inadequate.

To address this challenge, machine learning techniques offer a powerful solution by leveraging data-driven approaches to uncover hidden patterns and trends. This project focuses on developing a predictive model for flight prices using various machine learning algorithms. The goal is to explore and understand the trends in flight price datasets, evaluate multiple regression models, and identify the most suitable algorithms for accurate prediction. Additionally, ensemble methods are employed to combine the strengths of individual models and achieve superior performance.

By providing a robust framework for flight price prediction, this project aims to assist travelers in making informed booking decisions, enable airlines to optimize pricing strategies, and help businesses manage travel budgets effectively. This contributes to a more efficient and transparent aviation industry, benefiting all stakeholders involved.

# Objectives

The primary objective of this project is to develop a data-driven and innovative approach to predicting flight prices, addressing the dynamic and complex nature of the aviation industry. Flight price forecasting plays a crucial role in helping travelers make informed decisions, enabling airlines to optimize pricing strategies, and assisting businesses in managing travel budgets effectively. Traditional methods often struggle to capture the intricate dependencies and nonlinear relationships inherent in real-world flight data. By leveraging machine learning techniques, this project seeks to overcome these limitations, offering a more accurate and adaptive solution. Through thorough data exploration, we aim to uncover critical trends and patterns, understand seasonal variations, and identify key predictors influencing flight prices.

The project emphasizes the development and evaluation of multiple regression models, including linear regression, polynomial regression, lasso regression, ridge regression, k-nearest neighbors (KNN), support vector regression (SVR), random forest, decision tree, gradient boosting, and XGBoost. Each model will be carefully trained and tested to assess its performance under varying dataset sizes, allowing us to identify the best-performing model for different contexts. Beyond individual model analysis, ensemble techniques will be employed to combine the strengths of multiple algorithms. This approach aims to enhance the accuracy, reliability, and generalizability of the predictions, ensuring robust results that adapt to the unique characteristics of the flight price dataset.

Ultimately, this project strives to address real-world challenges in flight price prediction while demonstrating the potential of advanced machine learning techniques to revolutionize predictive analytics in the aviation sector. By optimizing forecasting accuracy, this work has the potential to benefit travelers, airlines, and businesses. It can enable smarter decision-making, better budget planning, and a more efficient approach to handling the dynamic nature of flight pricing. The fusion of data exploration, model evaluation, and ensemble techniques showcases the transformative impact of machine learning in tackling critical challenges in flight price prediction, paving the way for more informed and cost-effective travel strategies.

# Libraries Used

## 1)Numpy :

NumPy is a foundational Python library for numerical computing. It is used extensively for performing mathematical operations on arrays and matrices. In this project, NumPy plays a critical role in handling numerical data, such as calculating statistical measures, generating arrays, and performing matrix operations required during data preprocessing and analysis.



**NumPy**

## 2) Pandas:

**Pandas**



Pandas is a powerful library for data manipulation and analysis. It is used to load, clean, and organize the power consumption dataset. The DataFrame structure provided by Pandas simplifies operations such as handling missing values, filtering data, merging datasets, and generating descriptive statistics. Pandas also allows seamless exploration of trends and dependencies in the dataset.

## 3)Matplotlib:

Matplotlib is used to create basic plots, such as line graphs, bar charts, and scatter plots, to visually represent trends and relationships in the dataset.

**matplotlib**

## 5) Sklearn:

A powerful machine learning library in Python that provides tools for data preprocessing, model training, and evaluation. In this project, it's used to implement and train various regression models (like linear regression, decision trees, etc.) for predicting power consumption and evaluating model performance.



## 6) Flask:



**Flask** is a lightweight and flexible **Python web framework** designed for building web applications and APIs. It is known for its simplicity, minimalism, and extensibility, making it ideal for small to medium-sized projects and rapid prototyping. Flask provides core functionality like routing, templating (with Jinja2), and a built-in development server, while allowing developers to add extensions for additional features like database integration, authentication, and form handling. Its flexibility and ease of use make it a popular choice for building RESTful APIs, web applications, and integrating with data science or machine learning models. Flask is beginner-friendly and highly customizable, enabling developers to choose their tools and libraries.



# Dataset description

## About:

The “ **FlightPricePrediction** ” contains detailed information about flights Around Delhi, Mumbai, Kolkata, Hyderabad, Chennai, Bangalore focusing on **Economy class And Business Class** tickets. It includes features such as the airline, flight number, departure and arrival times, number of stops, flight duration, days left until departure, and ticket price. The goal of this dataset is to analyze and predict flight prices based on factors such as the airline, flight duration, number of stops, and days left until departure. This dataset can be used to explore pricing trends, optimize booking strategies, and improve customer decision-making in the aviation industry.

## Kaggle Link:

To know more about the dataset ,click here

<https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction/data>

## Columns:

- Airline
- Flight Number
- Source City
- Departure Time
- Stops
- Arrival Time
- Destination City
- Class(Economy/Business Class)

## Information of data:

The **Clean\_Dataset.csv** provides a detailed snapshot of flight options between Delhi, Mumbai, Kolkata, Hyderabad, Chennai, Bangalore, focusing on Economy class and Business Class tickets. It is a valuable resource for analyzing flight pricing, duration, and customer preferences. However, its limited scope and timeframe may restrict its applicability for broader analyses. This dataset is

particularly useful for building predictive models or conducting market research in the aviation industry.

```
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Unnamed: 0          300153 non-null int64
 1   airline              300153 non-null object
 2   flight               300153 non-null object
 3   source_city          300153 non-null object
 4   departure_time       300153 non-null object
 5   stops                300153 non-null object
 6   arrival_time         300153 non-null object
 7   destination_city     300153 non-null object
 8   class                300153 non-null object
 9   duration              300153 non-null float64
10   days_left            300153 non-null int64
11   price                300153 non-null int64
dtypes: float64(1), int64(3), object(8)
```

## Description of data:

```
Summary Statistics:
      Unnamed: 0      duration      days_left      price
count  300153.000000  300153.000000  300153.000000  300153.000000
mean    150076.000000    12.221021    26.004751   20889.660523
std      86646.852011     7.191997    13.561004   22697.767366
min         0.000000     0.830000     1.000000    1105.000000
25%      75038.000000     6.830000    15.000000    4783.000000
50%      150076.000000    11.250000    26.000000    7425.000000
75%      225114.000000    16.170000    38.000000   42521.000000
max      300152.000000    49.830000    49.000000  123071.000000
```

# Algorithms Used

1. Linear Regression
2. Lasso Regression
3. Decision Tree Regression
4. Random Forest Regression
5. Gradient Boosting Regressor
6. AdaBoost Regression
7. Extra Trees Regression
8. XGB Regression
9. K-Nearest Neighbors (KNN) Regression

# Algorithms Description

## 1. Linear Regression:

Linear Regression models the relationship between the dependent variable Y and one or more independent variables X using a linear equation. The goal is to find the best-fitting line that minimizes the difference between the observed and predicted values (error). The formula for simple linear regression (one independent variable) is:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Where:

- Y is the dependent variable (target).
- X is the independent variable (feature).
- $\beta_0$  is the intercept (constant term).
- $\beta_1$  is the slope (coefficient) of the line.
- $\epsilon$  is the error term.

In multiple linear regression (more than one independent variable), the formula becomes:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$

The algorithm aims to minimize the residual sum of squares to determine the best-fitting line.

## 2. Lasso Regression :

Lasso (Least Absolute Shrinkage and Selection Operator) Regression modifies linear regression by adding an L1 penalty to the loss function. This penalty forces some coefficients to become exactly zero, effectively performing feature selection. The formula for Lasso regression is:

$$\text{Loss Function} = \sum (Y_i - \hat{Y}_i)^2 + \lambda \sum |\beta_j|$$

Where:

- $y_i$  is the actual value,
- $\hat{y}_i$  is the predicted value,
- $w_j$  are the regression coefficients,
- $\lambda$  (alpha in Python) is the **regularization parameter** that controls the penalty,
- The second term  $\lambda \sum |w_j|$  is the **L1 penalty**, which forces some  $w_j$  to become Zero.

### 3. Decision Tree Regression:

A Decision Tree for regression makes predictions by splitting the data based on feature values. At each decision node, the algorithm chooses the feature that best divides the data into subsets, minimizing the variance within each subset. The tree is built recursively, where each split seeks to reduce the prediction error. The prediction for each leaf node is the average value of the target variable in that subset of data.

There is no explicit formula for Decision Trees, but the process follows these steps:

1. **Choose a feature:** Select the feature that results in the best split (usually by minimizing variance or maximizing information gain).
2. **Split the data:** Divide the dataset into subsets based on the feature's value.
3. **Repeat:** Continue splitting each subset recursively until the tree meets a stopping criterion (e.g., maximum depth or minimum sample size).
4. **Predict:** For a new input, traverse the tree based on the feature values until reaching a leaf node. The prediction is the mean target value of that leaf.

### 4. Random Forest Regression :

Random Forest Regression is an ensemble method that builds multiple decision trees, each trained on a random subset of the data. The trees are trained independently, and the final prediction is made by averaging the predictions of all individual trees. The random selection of features and data for each tree helps reduce overfitting.

While Random Forest doesn't have a single formula, it involves the following steps:

1. **Bootstrap sampling:** Randomly sample data points (with replacement) to create different training subsets for each tree.
2. **Random feature selection:** For each tree, randomly select a subset of features to split the data at each node.
3. **Build trees:** Construct a decision tree for each training subset, ensuring the trees are diverse by using random sampling and feature selection.
4. **Predict:** For a new input, each tree in the forest makes a prediction, and the final output is the average of all the predictions.

### 5. Gradient Boosting Regressor

Gradient Boosting Regressor is an ensemble learning technique that builds models

sequentially, where each new model attempts to correct the errors of the previous one. It uses gradient descent optimization to minimize the loss function. The final model is a weighted sum of all the individual models (typically decision trees). The formula for the prediction is:

$$F(x) = \sum_{m=1}^M \gamma_m h_m(x)$$

Where:

- $F(x)$  is the final model prediction.
- $M$  is the number of models (trees).
- $\gamma_m$  is the weight assigned to the  $m$ -th model.
- $h_m(x)$  is the prediction of the  $m$ -th model.

Gradient Boosting is effective for capturing complex, nonlinear relationships and is robust to overfitting when properly tuned.

## 6. AdaBoost Regression

AdaBoost (Adaptive Boosting) Regression is another ensemble technique that combines multiple weak learners (typically decision trees) into a strong learner. Unlike Gradient Boosting, AdaBoost focuses on reweighting the data points that are mispredicted by previous models, giving them higher importance in subsequent iterations. The final prediction is a weighted sum of the predictions from all weak learners:

$$F(x) = \sum_{m=1}^M \alpha_m h_m(x)$$

Where:

- $F(x)$  is the final model prediction.
- $M$  is the number of weak learners.
- $\alpha_m$  is the weight assigned to the  $m$ -th weak learner.
- $h_m(x)$  is the prediction of the  $m$ -th weak learner.

AdaBoost is particularly useful for improving the performance of weak models and is less prone to overfitting compared to individual models.

## 7. Extra Trees Regression

Extra Trees (Extremely Randomized Trees) Regression is an ensemble method that builds multiple decision trees and averages their predictions. Unlike Random Forest, Extra Trees introduces additional randomness by selecting split points for features at random, rather than computing the optimal split. This reduces variance and computational cost. The final prediction is the average of all individual tree predictions:

$$F(x) = \frac{1}{M} \sum_{m=1}^M h_m(x)$$

Where:

- $F(x)$  is the final model prediction.
- $M$  is the number of trees.
- $h_m(x)$  is the prediction of the  $m$ -th tree.

Extra Trees is computationally efficient and works well for high-dimensional data.

## 8. XGB Regression

XGBoost (Extreme Gradient Boosting) Regression is an advanced implementation of gradient boosting designed for speed and performance. It uses a more regularized model formalization to control overfitting and includes additional features like handling missing values and built-in cross-validation. The objective function for XGBoost is:

$$\text{Obj}(\theta) = \sum_{i=1}^n L(y_i, \hat{y}_i) + \sum_{m=1}^M \Omega(h_m)$$

Where:

- $L(y_i, \hat{y}_i)$  is the loss function (e.g., mean squared error).
- $\Omega(h_m)$  is the regularization term for the  $m$ -th tree.
- $\hat{y}_i$  is the prediction for the  $i$ -th data point.

- $MM$  is the number of trees.

XGBoost is highly scalable, supports parallel processing, and is widely used in competitions and real-world applications due to its accuracy and efficiency.

## 9. K-Nearest Neighbors Regression :

K-Nearest Neighbors (KNN) Regression is a non-parametric algorithm used for regression tasks, where the output value is predicted based on the average of the values of its  $k$ -nearest neighbors. KNN is simple and intuitive, operating by searching for the  $k$  closest data points to the input and then predicting the target value as the mean of those nearest neighbors' target values.

The algorithm works as follows:

1. For a given test point, calculate the distance between the test point and all other points in the training set.
2. Sort the distances and identify the  $k$  closest neighbors.
3. Compute the average of the target values of these  $k$  neighbors to predict the output for the test point.

The formula for KNN regression is:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^k y_i$$

where  $\hat{y}$  is the predicted output,  $k$  is the number of nearest neighbors, and  $y_i$  are the target values of the  $k$  nearest neighbors. The distance between points is typically measured using metrics like Euclidean distance:

$$d(x, x') = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}$$

where  $x$  and  $x'$  are the feature vectors of two data points.



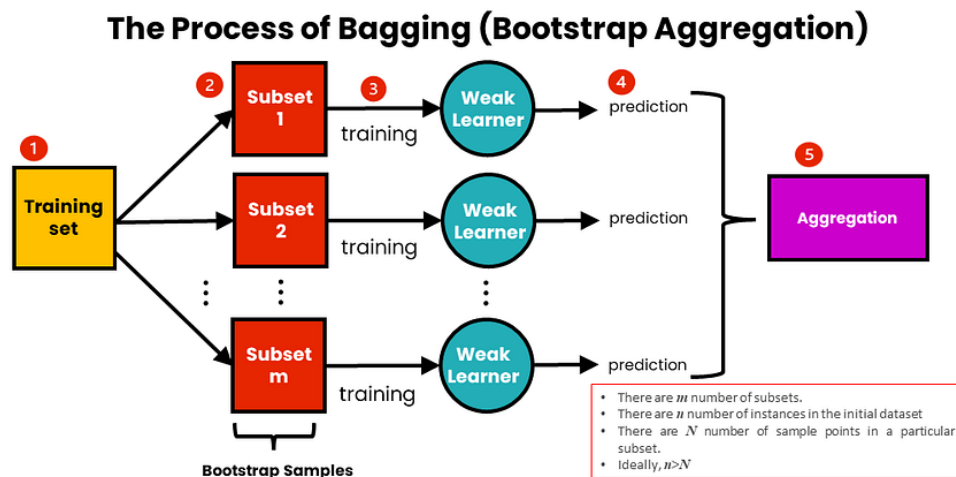
# Ensembling

Ensembling is a machine learning technique that combines the predictions from multiple models to improve the overall performance, typically by reducing variance (overfitting), bias (underfitting), or both. It is based on the idea that combining the strengths of multiple models can lead to better results than relying on a single model. Here are some common ensembling methods:

## Bagging:

Bagging works by training multiple instances of the same algorithm on different random subsets of the training data. Each model is trained on a different bootstrap sample (random sample with replacement). The predictions of these models are averaged (for regression) or voted on (for classification).

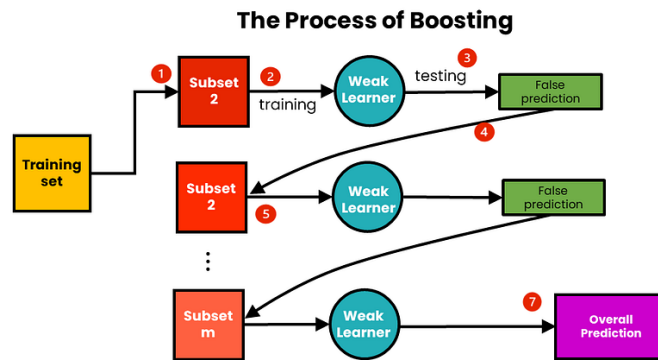
- **Random Forest** is a popular example of bagging, where many decision trees are trained on different subsets of the data, and the final prediction is the majority vote or the average of their predictions.



## Boosting:

Boosting is an ensemble method that builds models sequentially, where each model attempts to correct the errors made by the previous ones. The key idea is to assign higher weights to the misclassified data points, forcing subsequent models to focus more on these difficult cases.

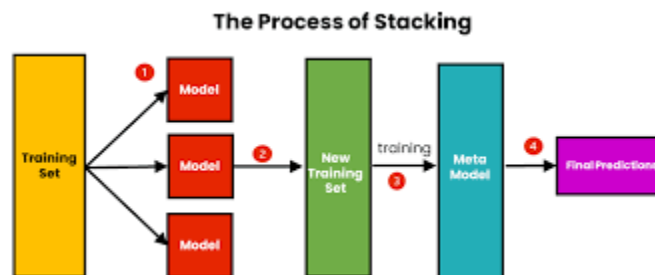
- Examples:
  - **AdaBoost (Adaptive Boosting):** It assigns higher weights to misclassified instances and adds weak learners sequentially to improve predictions.
  - **Gradient Boosting:** It fits new models to the residual errors (differences between predicted and actual values) of the previous models.
  - **XGBoost, LightGBM, and CatBoost** are optimized versions of gradient boosting that provide better performance and speed.



## Stacked Generalization:

Stacking involves training multiple models (often of different types) and then using a meta-model to combine their predictions. The idea is that different models will have different strengths, and a meta-model can learn the best way to combine them.

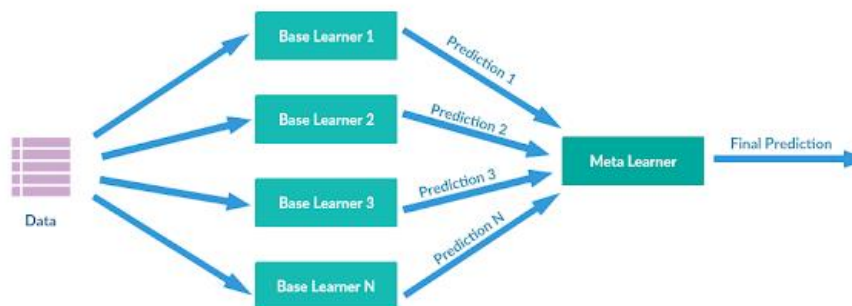
- How it works:
  1. Train multiple base models on the data.
  2. Use the predictions from these base models as features for a meta-model.
  3. The meta-model learns how to best combine the outputs of the base models to make the final prediction.



## Voting:

Voting is a simple ensembling method where predictions from multiple models are combined by majority voting (for classification) or averaging (for regression). There are two types of voting:

- **Hard Voting:** Each model gives a class label, and the class with the majority vote is chosen as the final prediction.
- **Soft Voting:** Each model gives a probability for each class, and the class with the highest average probability is chosen.



## Blending :

Blending is similar to stacking but differs mainly in how the base model predictions are combined. Instead of using the training data to build a meta-model (as in stacking), blending uses a holdout validation set to train the meta-model.

In the project ,we have implemented a variety of models that can be used individually or as part of an ensemble strategy:

- **Linear Models:** Lasso, Ridge, Linear Regression
- **Tree-based Models:** Decision Tree, Random Forest
- **Support Vector Machines:** SVR
- **Instance-based Learning:** KNN
- **Isotonic Regression:** Special handling for regression with non-monotonic relationships
- **Polynomial Regression:** For capturing non-linear relationships

To enhance prediction accuracy, we could combine these models using ensembling techniques. For example:

- Use **Bagging** to combine models like Decision Trees or KNN.

- Use **Boosting** techniques like **Gradient Boosting** to combine weak learners for improving regression performance.
- Use **Stacking** to combine multiple models of different types (e.g., Linear Regression, Random Forest, and KNN) and use a meta-model to improve the final prediction.

Ensembling works well when combining models that perform well but have complementary weaknesses. This way, the errors of one model are corrected by others, leading to a more robust prediction.

We used a **weighted average ensemble** method, where predictions from multiple models (in this case, seven regression models) are combined using weights based on their  $R^2$  scores. Here's how it works:

1. **Model Training:** Each model is trained using the same training data ( $X_{\text{train}}$ ,  $y_{\text{train}}$ ).
2. **Prediction:** Each model makes a prediction on the scaled input data ( $\text{input\_scaled}$ ).
3.  **$R^2$  Calculation:** The  $R^2$  score for each model is computed using the test data ( $X_{\text{test}}$ ,  $y_{\text{test}}$ ).
4. **Weight Calculation:** The weight for each model is calculated based on its  $R^2$  score relative to the total  $R^2$  of all models. This means models with higher  $R^2$  scores contribute more to the final prediction.
5. **Final Prediction:** The final prediction is computed as the weighted sum of the individual model predictions, with each model's prediction scaled by its corresponding weight.

This approach uses the  $R^2$  score as a performance metric to determine the importance of each model in the ensemble. Models that perform better (i.e., have higher  $R^2$  scores) are given more influence in the final prediction.

# Implementation

## Core Functionalities:

### 1. Dataset Handling:

- A dataset (clean\_Dataset.csv) is loaded and preprocessed by dropping unnecessary columns and scaling the features.
- The target variable is Price , and the remaining columns are used as features.

### 2. Model Selection & Training:

- Users can choose various regression models (e.g., Lasso, linear Regression, Decision Tree, etc.) and train them on the dataset.
- The models are trained using scikit-learn and evaluated on multiple regression metrics, including MAE, MSE, R2 score, etc.
- Trained models are saved to disk using joblib.

### 3. Model Metrics:

- Metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R2 score, and others are calculated and displayed after training a model.

### 4. Routes and Pages:

- The app has routes for:
  - Home page
  - Model training and results display
  - Dataset exploration

## Flask Routes Overview:

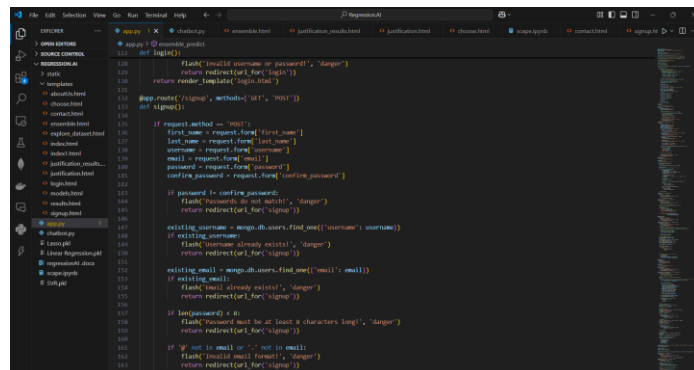
- Home Route( / )
- Prediction Route( /predict)
- Capture User Input
- Dynamic Model Selection
- DataProcessing
- Feature Selection
- Make Prediction
- Prepare results

## Model Training and Evaluation:

- **Model Selection:** Models like Lasso, Linear Regression, Decision Tree, Random Forest, XGB, etc., are supported.
- **Training Process:** The train\_model route handles the training of the selected model, and it includes functionality for calculating and displaying regression metrics.
- **Metrics:** The app calculates several performance metrics for model evaluation:
  - MAE (Mean Absolute Error)
  - MSE (Mean Squared Error)
  - RMSE (Root Mean Squared Error)
  - R2 (R-squared score)

## Areas for Enhancement or Review:

- **train\_justification Route:** The code cuts off while defining this function, but it seems like it's aimed at testing different train-test splits for model validation.
- **Model Saving:** Each trained model is saved with a filename corresponding to its name using Joblib (e.g., Lasso.pkl).
- 



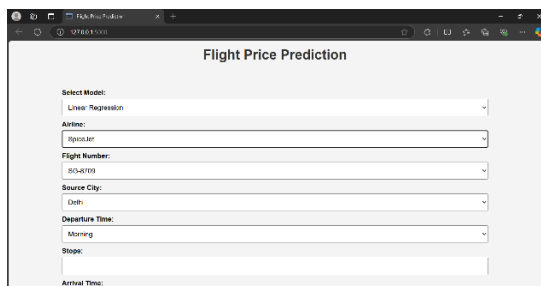
```
def login():
    flash('Invalid username or password!', 'danger')
    return redirect(url_for('login'))

def signup():
    flash('Invalid username or password!', 'danger')
    return redirect(url_for('signup'))

def password_confirm():
    flash('Password does not match!', 'danger')
    return redirect(url_for('password_confirm'))

def password_confirm():
    flash('Password does not match!', 'danger')
    return redirect(url_for('password_confirm'))

def password_confirm():
    flash('Password does not match!', 'danger')
    return redirect(url_for('password_confirm'))
```



Flight Price Prediction

Select Model: Linear Regression

Airline: Spoken

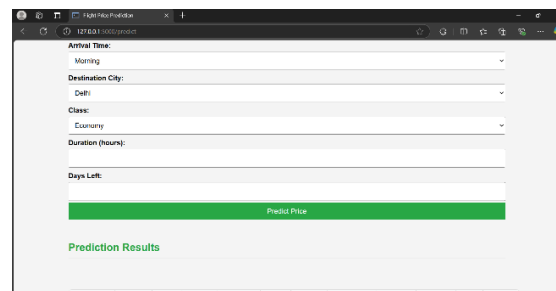
Flight Number: 80-4709

Source City: Delhi

Departure Time: Morning

Stop:

Arrival Time:



Arrival Time: Morning

Destination City: Delhi

Class: Economy

Duration (hours):

Days Left:

Predicted Price:

Prediction Results

# Metrics

In a regression model, various metrics are used to evaluate performance by measuring the difference between predicted and actual values. The following metrics are considered in this project:

## 1 )Mean Absolute Error(MAE):

MAE measures the average absolute difference between the actual values and the predicted values. It provides a linear score, meaning all individual differences are weighted equally.

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

Where:

- $y_i$  = Actual value
- $\hat{y}_i$  = Predicted value
- $n$  = Number of observations

### Interpretation:

Lower MAE indicates a better fit. Since it does not square the errors, it is less sensitive to large deviations.

## 2 )Mean Squared Error(MSE):

MSE calculates the average squared difference between actual and predicted values. The squaring of errors penalizes larger errors more than smaller ones.

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

Where:

- $y_i$  = Actual value
- $\hat{y}_i$  = Predicted value
- $n$  = Number of observations

### Interpretation:

Lower MAE indicates a better fit. Since it does not square the errors, it is less sensitive to large deviations.

### 3. Root Mean Squared Error (RMSE)

RMSE is simply the square root of MSE. It has the same units as the dependent variable, making it easier to interpret.

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

#### Interpretation:

RMSE is useful when larger errors need to be more heavily penalized. Lower RMSE values indicate better performance.

### 4. Mean Absolute Percentage Error (MAPE)

MAPE measures the percentage error by dividing the absolute error by the actual value.

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

#### Interpretation:

Lower MAPE indicates better model performance. However, MAPE can be problematic when actual values are close to zero, leading to high error percentages.

### 5. R-squared (R^2) Score

#### Definition:

The R^2 score, also known as the coefficient of determination, represents the proportion of variance in the dependent variable that is explained by the independent variables.

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

Where:

- $\bar{y}$  = Mean of actual values



**Interpretation:**

- $R^2 = 1 \rightarrow$  Perfect fit
- $R^2 = 0 \rightarrow$  Model explains no variance
- $R^2 < 0 \rightarrow$  Model performs worse than simply using the mean

Higher  $R^2$  indicates better model fit, but it does not account for overfitting.

## 6. Explained Variance Score (EVS)

EVS measures the proportion of variance that the model explains relative to the total variance.

$$EVS = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$$

**Interpretation:**

Similar to  $R^2$ , but more sensitive to outliers. A higher value means the model explains more variance.

## 7. Maximum Error

This metric finds the maximum absolute error between the actual and predicted values.

$$\text{Max Error} = \max(|y_i - \hat{y}_i|)$$

**Interpretation:**

Useful for identifying worst-case errors, but does not provide an overall measure of performance.

## 8. Median Absolute Error (MedAE)

MedAE computes the median of all absolute differences between actual and predicted values.

$$\text{MedAE} = \text{median}(|y_i - \hat{y}_i|)$$

**Interpretation:**

Unlike MAE, MedAE is less sensitive to outliers.

**9. Sum of Squared Errors (SSE)**

SSE measures the total squared differences between actual and predicted values.

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

**Interpretation:**

SSE is useful when comparing models, but is not normalized.

# Future Scope

## 1. Cloud-Based Session Management and Database Handling

- Implement cloud storage (AWS S3, Google Cloud Storage, Firebase) for better scalability and reliability of user sessions and model storage.
- Use serverless databases (MongoDB Atlas, Firebase Firestore, AWS DynamoDB) to manage user data and model metadata efficiently.
- Enable multi-user authentication with OAuth, JWT, or Firebase Auth to support secure access across multiple devices.

## 2. Model Download and Deployment Options

- Allow users to download trained models in multiple formats (.pkl, .h5, .onnx) for offline use.
- Provide API-based access so users can integrate trained models into their own applications.
- Enable one-click deployment of models to cloud-based inference services like Google AI Platform, AWS SageMaker, or Hugging Face Spaces.

## 3. Improved User Experience and UI Enhancements

- Design an interactive and responsive dashboard for better visualization of regression results.
- Implement drag-and-drop dataset upload and interactive feature selection.
- Use dynamic plots with Plotly or D3.js for real-time analytics and comparisons between models.

## 4. Enhanced Dataset Upload and Exploration

- Support bulk dataset uploads in multiple formats (CSV, Excel, JSON, Parquet).
- Implement automated EDA (Exploratory Data Analysis) with summary statistics, feature correlations, and missing data handling.
- Add interactive filtering and transformation tools to clean and preprocess datasets before training.

## **5. Deep Learning Model Integration**

- Extend the platform to support Deep Learning models like:
  - Artificial Neural Networks (ANNs) for complex non-linear regression tasks.
  - LSTMs & GRUs for time-series forecasting.
  - CNNs for spatial regression tasks like image-based predictions.
- Implement TensorFlow and PyTorch support for model training and inference.

## **6. Advanced Feature Engineering and Selection**

- Implement automated feature extraction techniques using Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE).
- Introduce domain-specific feature engineering options for financial, healthcare, and industrial datasets.
- Support feature interaction detection and polynomial feature expansion to improve model performance.

## **7. Model Explainability and Interpretability**

- Add SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations) to interpret model predictions.
- Provide bias detection and fairness analysis to ensure ethical AI practices.

## **8. Real-Time Model Performance Tracking**

- Implement Live Model Monitoring using tools like MLflow or TensorBoard.
- Allow users to compare training vs. validation performance dynamically.
- Provide alerts for model drift when the dataset distribution changes over time.

## **9. Ensemble Learning and AutoML Features**

- Support Stacking, Bagging, and Boosting (XGBoost, LightGBM, CatBoost) for better regression performance.
- Implement AutoML to automatically select the best model and hyperparameters based on dataset characteristics.

# Web References

## **1.Power Consumption Dataset**

<https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction/data>

## **2. Regression Analyses and Their Particularities in Observational Studies**

<https://pmc.ncbi.nlm.nih.gov/articles/PMC11019761/>

## **3. An Introduction to Regression Analysis**

[https://chicagounbound.uchicago.edu/cgi/viewcontent.cgi?article=1050&context=law\\_and\\_economics](https://chicagounbound.uchicago.edu/cgi/viewcontent.cgi?article=1050&context=law_and_economics)

## **4. Model Downloading**

<https://joblib.readthedocs.io/en/stable/>

## **5.Flask Reference**

<https://flask.palletsprojects.com/en/stable/>