Project on optimizing flight price prediction using machine learning

CONTENTS

1.Introduction

Nowadays, the airline corporations are using complex strategies for the flight ticket fare calculations. This highly complicated methods makes the flight ticket fare difficult to guess for the customers, since the fare changes dynamically.

- Our project "Optimizing Flight Booking Decisions Through Machine Learning Price Prediction" which resolve this problem and provide a facility where people will be able to predict the flight-ticket price before purchasing the ticket.

People who work frequently travel through flight will have better knowledge on best discount and right time to buy the ticket. For the business purpose many airline companies change prices according to the seasons or time duration. They will increase the price when people travel more. Estimating the highest prices of the airlines data for the route is collected with features such as Duration, Source, Destination, Arrival and Departure. Features are taken from chosen dataset and in the price wherein the airline price ticket costs vary overtime. we have implemented flight price prediction for users by using KNN, decision tree and random forest algorithms. Random Forest shows the best accuracy of 80% for predicting the flight price. also, we have done correlation tests and metrics for the statistical analysis.

1.1 overview

Optimizing flight price prediction using machine learning involves utilizing various techniques and algorithms to accurately forecast flight ticket prices. The goal is to provide travelers with reliable and timely information about when to book flights to get the best deals, while also assisting airlines and travel agencies in pricing strategies. Here's an overview of the process:

Data collection: This step involves gathering the necessary data for your project or analysis. It can be done by scraping websites, using APIs, or obtaining datasets from various sources.

Visualizing and analyzing data: In this step, you explore the dataset to gain insights and understand its structure. Data visualization techniques like plots, charts, and graphs are used to represent the data visually.

- **Univariate analysis:** Univariate analysis focuses on examining one variable at a time. It helps to understand the distribution, central tendency, and variability of individual variables.
- **Multivariate analysis:** In contrast to univariate analysis, multivariate analysis involves the simultaneous study of multiple variables to uncover patterns, relationships, and dependencies among them.
- **Descriptive analysis:** Descriptive analysis provides a summary of the main characteristics of the data, such as mean, median, standard deviation, etc.

Data pre-processing: This step involves preparing the data for modeling. It includes tasks like handling missing values, dealing with outliers, and converting categorical variables into numerical representations.

- **Drop unwanted features:** Remove any irrelevant or redundant features that may not contribute to the model's performance.
- **Checking for null values:** Identify and handle missing values in the dataset.
- **Remove negative data:** Depending on the context of the problem, you may need to remove or handle negative data points appropriately.
- **Handling outliers:** Outliers are extreme values that can significantly impact the model. They need to be treated carefully, either by removing them or using transformation techniques.
- **Handling categorical data:** Categorical variables need to be encoded into numerical form before feeding them into the model. Common techniques include one-hot encoding and label encoding.
- **Handling Imbalanced data:** If the dataset has imbalanced class distributions, techniques like oversampling, under sampling, or using class weights can help balance the data.

- **Splitting data into train and test:** The dataset is split into two parts - one for training the model and the other for testing its performance. This helps in assessing the model's generalization ability.
- **Model building**: Select a suitable machine learning or deep learning model based on the problem type and dataset characteristics.

Model building: It is a crucial step in the data science and machine learning workflow. It involves creating a mathematical representation or algorithm that can learn patterns and relationships from the input data and make predictions or decisions on new, unseen data.

- **Import the model building libraries:** Import the required libraries and frameworks in your programming environment for building the chosen model.
- **Initializing the model:** Set up the model architecture with the appropriate hyperparameters.
- **Training and testing the model:** Feed the training data into the model, adjust the model parameters during training, and then test the model's performance using the test data.
- **Evaluating performance of the model:** Use various evaluation metrics to assess the model's performance, such as accuracy, precision, recall, F1-score, etc.
- **Save the model:** Save the trained model's weights and configuration to be used later for predictions without retraining.

Application Building: Create a user-friendly interface for your model, such as an HTML file, where users can interact with the model and provide input data.

- **Create an HTML file**
- **Build Python code:** Develop the Python code that integrates the model and the user interface to make predictions.
- **Run the Application:** Deploy the application and run it to interact with the model, providing input data and obtaining predictions or results.

By following these steps, optimizing flight price prediction using machine learning can lead to more accurate and timely predictions, benefiting both travelers and the travel industry as a whole.

1.2 purpose

The purpose of optimizing flight price prediction using machine learning is to develop accurate and reliable models that can forecast the prices of airline tickets. This can offer several benefits to both consumers and the airline industry:

- **Cost Savings for Consumers:** Accurate flight price predictions allow travelers to make more informed decisions about when to book their flights. This can help them find the best deals and avoid overpaying for tickets, leading to cost savings for individuals and families.
- **Revenue Generation for Airlines:** Airlines can use accurate price prediction models to adjust their pricing strategies dynamically. By understanding demand patterns and market trends, airlines can optimize their revenue by offering competitive prices while maximizing seat occupancy.
- **Demand Forecasting:** Machine learning models can analyze historical data and current market conditions to predict demand for specific flights. This enables airlines to allocate resources more efficiently and plan their operations better.
- **Inventory Management:** Airlines can optimize their inventory and seat allocation based on predicted demand. This can help prevent under booking or overbooking, which can lead to revenue loss or customer dissatisfaction.
- **Personalized Pricing and Offers:** Machine learning can help airlines tailor pricing and offers to individual customers based on their preferences, purchase history, and browsing behavior. This can enhance customer satisfaction and loyalty.
- **Competitive Advantage:** Airlines that successfully implement accurate price prediction models can gain a competitive advantage by attracting cost-conscious travelers and maintaining a strong market position.
- **Data-Driven Decision Making:** Machine learning models provide data-driven insights that can guide strategic decisions related to route planning, marketing campaigns, and revenue management.

- **Enhanced Customer Experience:** By helping customers find the best deals and options, accurate flight price prediction contributes to an improved overall travel experience.

2.LITERATURE SURVEY

2.1 Existing problem:

Flight price prediction is a challenging problem that involves many factors, such as demand, supply, seasonality, competition, and so on. Many researchers and practitioners have tried to develop models and algorithms to forecast flight prices accurately and efficiently. Some of the existing problems of flight price prediction are:

- **Dynamic Nature of Pricing:** Flight prices are highly dynamic and can change rapidly based on various factors such as demand, seasonality, fuel prices, airline policies, and external events. Predicting these changes accurately can be challenging.
- **Limited Data Availability:** Historical flight price data might not always be readily available or easily accessible for all routes and airlines. Accurate predictions require a vast and comprehensive dataset spanning an extended period.
- **Feature Engineering:** The success of machine learning algorithms for flight price prediction relies on relevant and meaningful features. Identifying and selecting the right features that contribute significantly to price fluctuations is not always straightforward.
- **Uncertainty in External Factors:** Many external factors, such as geopolitical events, natural disasters, and regulatory changes, can significantly impact flight prices. Predicting these unpredictable events accurately is challenging.
- **Competing Factors:** Multiple factors influence flight prices, and some may even contradict each other. For instance, an airline may offer discounts to attract more customers on a particular route while simultaneously raising prices due to increased demand.

- **Seasonal and Temporal Variations:** Flight prices can exhibit seasonal patterns and temporal variations, making it challenging to develop models that generalize well over time.
- **User-Specific Behavior:** User-specific preferences and booking behavior can be highly variable, making it challenging to predict individual flight prices accurately.
- **Real-time Updates:** As flight prices change frequently, real-time updates become crucial for providing users with up-to-date and reliable predictions.
- **Model Complexity:** Developing an accurate flight price prediction model often requires sophisticated machine learning techniques and computational resources.
- **Lack of Standardization:** The lack of standardization in how airlines price their flights and present their data can create inconsistencies in the prediction process.

2.2 Proposed Solution

- The proposed approach is using machine learning algorithm and we are using supervised learning.
- We are gathering our data from a site. The data is containing some of the details of Indian flight of a short duration.
- This project involves the feature engineering for processing the dataset(data) to convert it into data frame. When we have the processed data frame, we move to normalizing the data frame.
- The regression and ensemble models which we have selected for our prediction is
    - Random Forest Regressor
    - Gradient Boosting Regressor
    - AdaBoost Regressor
    - Randomized Search CV
    - Decision Tree Regressor
    - Linear Regression
    - K-Neighbors Regressor

- Training of model followed as:

  Ensemble Learning Algorithms (RandomForestRegressor, GradientBoostingRegressor, and AdaBoostRegressor):

  - Each algorithm is initialized and stored in a list [rfr, gb, ad].
  - The script iterates through each algorithm using a for loop.
  - For each algorithm, it fits the training data (x_train, y_train) to the model using the .fit() method.
  - After fitting, the algorithm predicts the target variable on the test data (x_test) and stores the predictions in y_pred.
  - The code calculates the R-squared (R2) score for the test data and the train data using the r2_score function from the sklearn.metrics module. The R2 score measures the goodness of fit of the model to the data.
  - If the absolute difference between the R2 score for the train data and the R2 score for the test data is less than or equal to 0.2, it prints the following information for that algorithm:
  - The name of the algorithm (e.g., RandomForestRegressor).
  - The R2 score for the test data.
  - The R2 score for the train data.
  - The Mean Absolute Error (MAE) between the predicted flight prices (y_pred) and the actual flight prices (y_test).
  - The Mean Squared Error (MSE) between the predicted flight prices and the actual flight prices.
  - The Root Mean Squared Error (RMSE), which is the square root of the MSE.

  Other Regression Algorithms (KNeighborsRegressor, LinearRegression, and SVR):

  - Similar to the ensemble learning algorithms, each of these algorithms is initialized and stored in a list [knn, svr, dt].
  - The script iterates through each algorithm using a for loop.
  - For each algorithm, it fits the training data (x_train, y_train) to the model using the .fit() method.

- After fitting, the algorithm predicts the target variable on the test data (x_test) and stores the predictions in y_pred.
- The code calculates the R-squared (R2) score for the test data and the train data using the r2_score function from the sklearn.metrics module.
- If the absolute difference between the R2 score for the train data and the R2 score for the test data is less than or equal to 0.1, it prints the following information for that algorithm:
- The name of the algorithm (e.g., KNeighborsRegressor).
- The R2 score for the test data.
- The R2 score for the train data.
- The Mean Absolute Error (MAE) between the predicted flight prices (y_pred) and the actual flight prices (y_test).
- The Mean Squared Error (MSE) between the predicted flight prices and the actual flight prices.
- The Root Mean Squared Error (RMSE), which is the square root of the MSE.

- Checking cross validation for Random Forest Regressor:

  Linear Regression model (lr) and a Random Forest Regressor model (rfr). Hyperparameter tuning helps find the best combination of hyperparameters for the models that optimizes their performance.

  Hyperparameter Lists:

  - n_estimators: A list of integers representing the number of trees in the Random Forest Regressor. It contains 12 values linearly spaced between 100 and 1200 .
  - max_features: A list of strings with values 'auto' and 'sqrt'. It represents the number of features to consider for every split during the tree-building process in the Random Forest Regressor.
  - max_depth: A list of integers with values linearly spaced between 5 and 30. It represents the maximum depth of the decision trees in the Random Forest Regressor.
  - min_samples_split: A list of integers with values 2, 5, 10, 15, and 100. It represents the minimum number of samples required to split an internal node in the Random Forest Regressor.

- min_samples_leaf: A list of integers with values 1, 2, 5, and 10. It represents the minimum number of samples required to be at a leaf node in the Random Forest Regressor.

Random Grid:

- A random_grid dictionary is created by combining all the hyperparameter lists, forming a grid of different hyperparameter combinations that the RandomizedSearchCV will explore.
- RandomizedSearchCV for Linear Regression (lr):
- The RandomizedSearchCV is initialized with the Linear Regression model (lr), the random_grid dictionary of hyperparameters, and the number of iterations set to 10 (n_iter=10).
- The cv parameter is set to 3, indicating 3-fold cross-validation will be used during the hyperparameter search.
- The RandomizedSearchCV performs a randomized search over the specified hyperparameter grid for the Linear Regression model.

RandomizedSearchCV for Random Forest Regressor (rfr):

- The RandomizedSearchCV is similarly initialized, but the target model is set to the Random Forest Regressor (rfr).
- The scoring metric is set to 'neg_mean_squared_error', which means the search will aim to minimize the negative mean squared error (MSE) during hyperparameter tuning.
- The n_iter is again set to 10, and 3-fold cross-validation is used (cv=3).
- The RandomizedSearchCV performs a randomized search over the specified hyperparameter grid for the Random Forest Regressor model.

Hyperparameter Tuning:

- Both models (lr and rfr) are fitted with the training data (x_train and y_train).
- The RandomizedSearchCV performs hyperparameter tuning by trying different combinations of hyperparameters from the random grid and selecting the best combination based on the specified scoring metric (negative mean squared error for rfr and default score for lr).

- The tuning process uses cross-validation to estimate the model's performance on unseen data and avoid overfitting.

After the hyperparameter tuning is complete, the best hyperparameters and the corresponding best models for both Linear Regression and Random Forest Regressor are stored in the random_search and random_forest_cv objects, respectively. These models can then be used for prediction on new data or further analysis.

- Testing the model is followed as:

RandomForestRegressor (rfr) and a GradientBoostingRegressor (gb). The hyperparameter tuning is done to find the best combination of hyperparameters that optimize the performance of these regression models.

Hyperparameter Grid:

- param_grid: A dictionary containing the hyperparameters and their corresponding values that will be tuned during the RandomizedSearchCV process.
- For RandomForestRegressor (rfr), the hyperparameters being tuned are:
- n_estimators: Number of trees in the random forest. It takes values 10, 30, 50, 70, and 100.
- max_depth: Maximum depth of the decision trees. It takes values None (unlimited depth), 1, 2, and 3.
- max_features: The number of features to consider for every split. It only takes the value 'sqrt'.
- For GradientBoostingRegressor (gb), the same hyperparameters and their corresponding values are used for tuning.

RandomForestRegressor Hyperparameter Tuning:

- A RandomForestRegressor (rfr) object is created.
- The RandomizedSearchCV is initialized with the rfr model, the param_grid dictionary of hyperparameters, and the number of iterations set to the default value of 10 (n_iter=10).
- The cv parameter is set to 3, indicating 3-fold cross-validation will be used during the hyperparameter search.
- verbose=2 means the search progress will be printed.

- n_jobs=-1 indicates that the computation will be performed using all available CPU cores for faster processing.
- The RandomizedSearchCV performs a randomized search over the specified hyperparameter grid for the RandomForestRegressor model (rfr).
- After hyperparameter tuning, the best hyperparameters and the corresponding best RandomForestRegressor model are stored in the rf_res object.

GradientBoostingRegressor Hyperparameter Tuning:

- A GradientBoostingRegressor (gb) object is created.
- The RandomizedSearchCV is similarly initialized with the gb model, the param_grid dictionary of hyperparameters, and other parameters as set for the RandomForestRegressor (rfr) tuning.
- The RandomizedSearchCV performs a randomized search over the specified hyperparameter grid for the GradientBoostingRegressor model (gb).
- After hyperparameter tuning, the best hyperparameters and the corresponding best GradientBoostingRegressor model are stored in the gb_res object.

The code uses RandomizedSearchCV to perform a random search over the specified hyperparameter grid for both the RandomForestRegressor and GradientBoostingRegressor models. This approach is computationally efficient and helps find a good set of hyperparameters for these regression models, leading to improved performance on unseen data. The resulting best models can then be used for making predictions on new data or further analysis.

- Calculate accuracy of the models

  RandomForestRegressor (rfr) and GradientBoostingRegressor (gb), on both the training and test datasets. It then creates a DataFrame price_list to store the target variable "Price" from the original dataset.

RandomForestRegressor (rfr) Calculation and Printing:

- A RandomForestRegressor model is created with hyperparameters: 10 trees (n_estimators=10), 'sqrt' as the number of features for every split (max_features='sqrt'), and no maximum depth specified (max_depth=None).
- The model is trained on the training data using x_train as the input features and y_train as the target variable.

- Predictions are made on both the training and test data using rfr.predict(), and the predicted values are stored in y_train_pred and y_test_pred, respectively.
- The R-squared (R2) scores are calculated between the predicted values and the actual target values for both the training and test data.
- The R2 scores are then printed as "rfr train accuracy" and "rfr test accuracy," indicating the accuracy of the RandomForestRegressor model on the training and test datasets, respectively.

GradientBoostingRegressor (gb) Calculation and Printing:

- A GradientBoostingRegressor model is created with the same hyperparameters as the RandomForestRegressor model (10 trees, 'sqrt' as the number of features for every split, and no maximum depth specified).
- The model is trained on the training data using x_train as the input features and y_train as the target variable.
- Predictions are made on both the training and test data using gb.predict(), and the predicted values are stored in y_train_pred and y_test_pred, respectively.
- The R-squared (R2) scores are calculated between the predicted values and the actual target values for both the training and test data.
- The R2 scores are then printed as "gb train accuracy" and "gb test accuracy," indicating the accuracy of the GradientBoostingRegressor model on the training and test datasets, respectively.

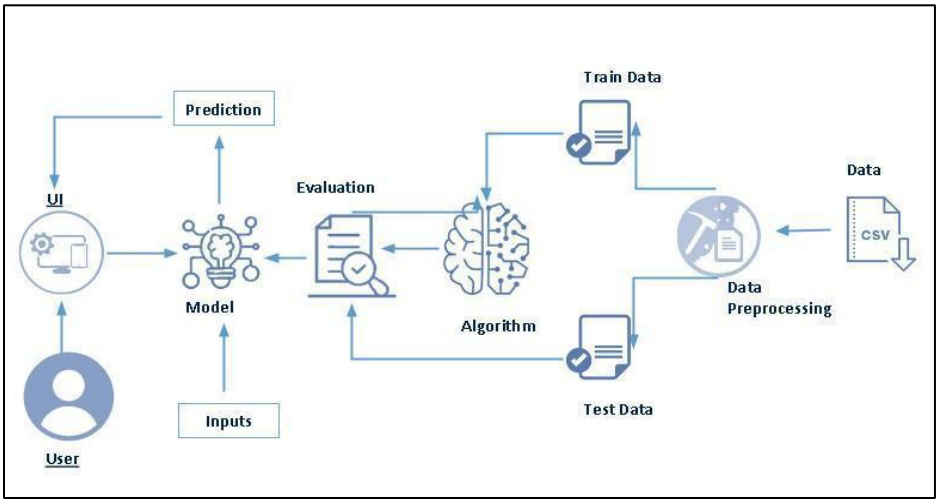
DataFrame Creation:

- The code extracts the "Price" column from the original dataset and creates a new DataFrame called price_list.
- The DataFrame price_list contains one column named "price" with the target variable values ("Price") from the original dataset.

It's worth noting that R-squared (R2) is a measure of how well the predicted values fit the actual values. R2 score can range from -∞ to 1, where 1 indicates a perfect fit, 0 indicates the model performs as well as the mean of the target variable, and negative values indicate poor performance. The higher the R2 score, the better the model's predictions align with the actual target values.

3.THEORITICAL ANALYSIS:

3.1 technical Architecture



Basic proposed system:

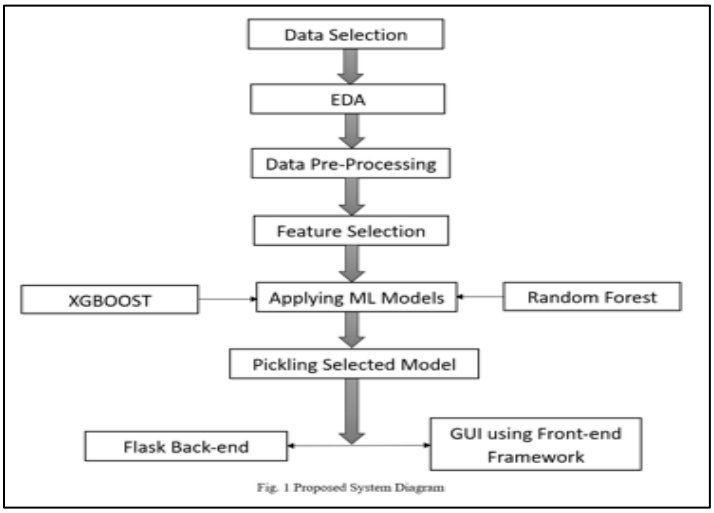


Fig. 1 Proposed System Diagram

Use case diagram:

3.2Hardware / Software designing

Technology:

- Machine Learning (Supervisedlearning).
- Python.
- HTML.
- CSS.
- Flask.

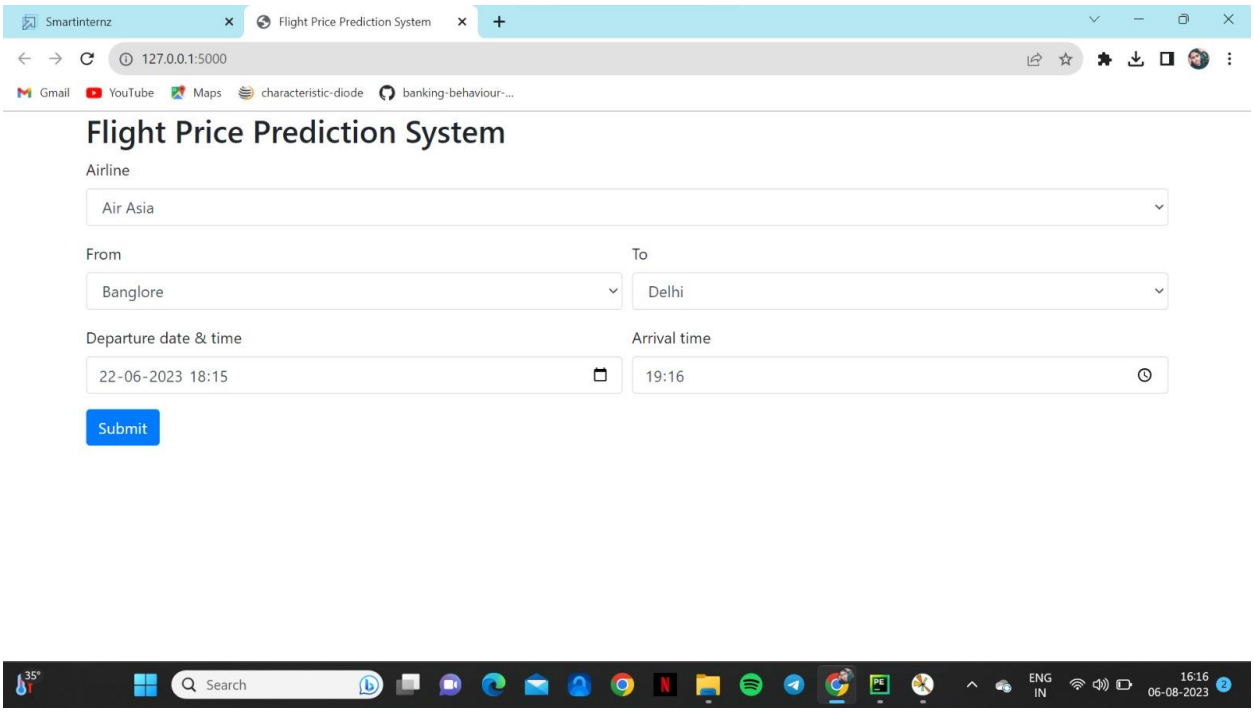
Software:

- Jupyter Notebook.
- Pycharm

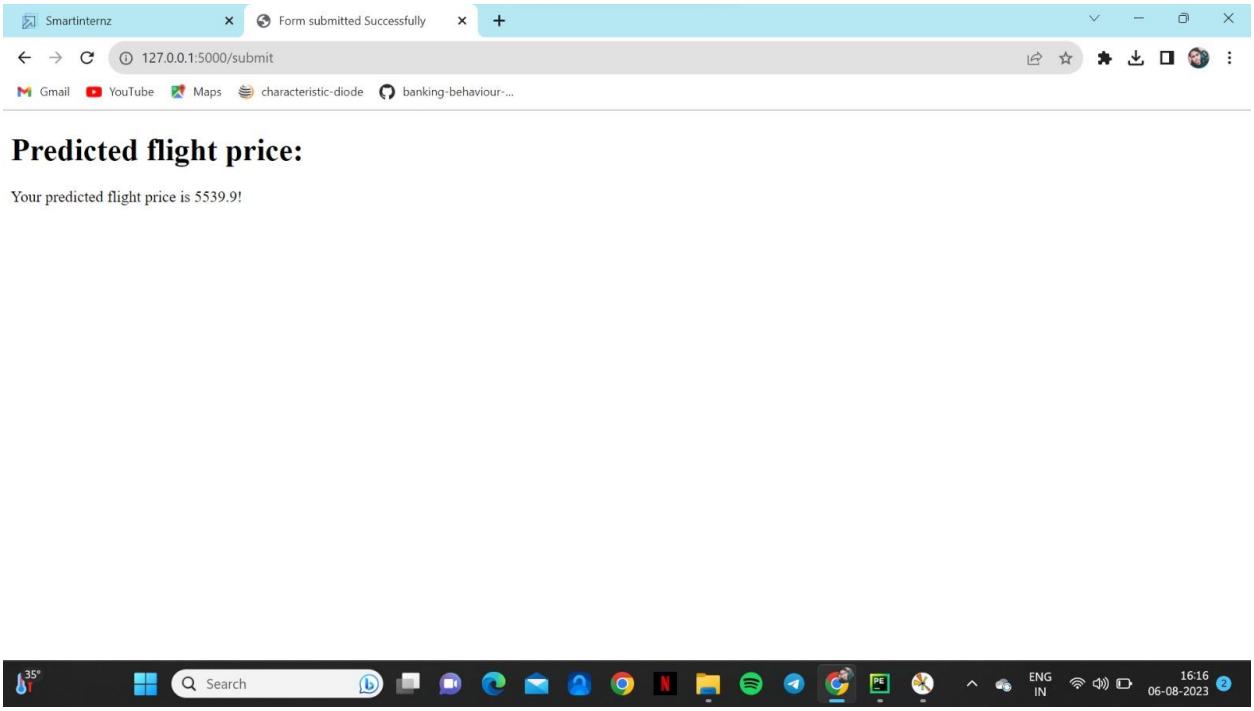
Hard ware:

- Processing power
- Memory (RAM)
- Storage

4.Result


Smartinternz
Flight Price Prediction System
127.0.0.1:5000
Gmail
YouTube
Maps
characteristic-diode
banking-behaviour-...
Flight Price Prediction System
Airline
Air Asia
From
Banglore
To
Delhi
Departure date & time
22-06-2023 18:15
Arrival time
19:16
Submit

Search
ENG
IN
16:16
06-08-2023


Smartinternz
Form submitted Successfully
127.0.0.1:5000/submit
Gmail
YouTube
Maps
characteristic-diode
banking-behaviour-...
Predicted flight price:
Your predicted flight price is 5539.9!

Search
ENG
IN
16:16
06-08-2023

5.Advantages:

1. **Improved Accuracy:** An optimized prediction system can provide more accurate forecasts of flight prices, helping travelers make better informed decisions.
2. **Cost Savings:** Travelers can potentially save money by booking flights at optimal times when prices are lower, leading to cost savings.
3. **Better Planning**: Accurate price predictions enable travelers to plan their trips well in advance, taking advantage of lower prices and availability.
4. **Competitive Edge:** Airlines and travel agencies can gain a competitive edge by offering customers accurate and timely price predictions.
5. **Enhanced Customer Experience:** Travelers appreciate transparency and accurate information, leading to a better overall experience.
6. **Data-Driven Insights:** The prediction system can provide valuable insights into pricing trends, seasonality, and demand patterns, helping airlines adjust their strategies.
7. **Automation**: Once set up, the system can automate the process of monitoring and predicting flight prices, saving time and resources.
8. **Integration Possibilities:** The system can be integrated into travel apps, websites, or platforms, enriching their offerings.

Disadvantages:

1. **Data Quality Dependence**: Accurate predictions rely heavily on high-quality and up-to-date data. Inaccurate or incomplete data can lead to flawed predictions.
2. **Complexity:** Designing and implementing an optimized prediction system can be complex, requiring advanced data science and technical expertise.
3. **Algorithm Performance:** The chosen prediction algorithm(s) might not always perform well, leading to less accurate forecasts.
4. **Variable Factors:** Flight prices can be influenced by a multitude of ever-changing factors, making accurate predictions challenging.
5. **External Shocks:** Unforeseen events like geopolitical issues, natural disasters, or economic crises can disrupt price patterns, affecting prediction accuracy.

6. **Ethical Concerns:** Privacy and ethical concerns may arise if the system relies on user data to make predictions.
7. **User Behavior Changes:** If users change their behavior based on predictions (e.g., booking more during predicted low-price periods), it could alter the accuracy of the model.
8. **Maintenance and Updates:** The prediction model requires ongoing maintenance, updates, and monitoring to adapt to changing market conditions.
9. **Overfitting:** Depending on the complexity of the model, there's a risk of overfitting to historical data and losing generalization capability.

6.Applications

An optimized flight price prediction system can have numerous applications across the travel and aviation industry, as well as for travelers and businesses. Here are some key areas where such a system can be applied:

1. **Travel Booking Platforms:** Online travel agencies (OTAs) and booking websites can integrate the prediction system to provide users with real-time insights into flight price trends, helping them make informed booking decisions.
2. **Airlines:** Airlines can use optimized price prediction to dynamically adjust their pricing strategies based on demand and market conditions, optimizing revenue management.
3. **Travel Apps:** Mobile apps focused on travel planning can incorporate the prediction system to offer users personalized travel itineraries and options for cost-effective flight bookings.
4. **Corporate Travel Management**: Businesses that manage corporate travel arrangements can benefit from accurate price predictions to optimize travel budgets and reduce expenses.
5. **Price Tracking Tools:** Price tracking tools can alert travelers when flight prices drop or reach a certain threshold, enabling them to book flights at optimal times.
6. **Frequent Flyers:** Frequent flyers can use the prediction system to plan their travel schedule and redeem miles or loyalty points when flight prices are projected to be lower.
7. **Travel Deal Websites:** Websites that aggregate travel deals and discounts can incorporate price prediction to showcase the best times to book flights for maximum savings.

8. **Vacation Planners:** Travel planners can optimize itineraries by suggesting alternative travel dates based on predicted price fluctuations.
9. **Market Research:** Researchers and analysts in the travel industry can use the prediction system to study market trends, pricing dynamics, and consumer behavior.
10. **Airfare Alerts:** Services that send airfare alerts and notifications to subscribers can utilize price prediction to enhance the accuracy of their notifications.
11. **Travel Insurance Providers**: Travel insurance companies can factor in predicted flight prices when calculating insurance coverage.

7.Conclusion

This System customer can predict the flight price of a particular seat on a day before booking the flight tickets. It will provide an ease to the customer for the flight ticket booking. Travelers can save money if they choose to buy a ticket when its price is the lowest. The problem is how to determine when is the best time to buy flight ticket for the desired destination and period.

8.Scope

- For purchasing an airplane ticket, the traditional purchase strategy is to buy a ticket far in advance of the flight's departure date to avoid the risk that the price may increase rapidly before the departure date. However, this is usually not always true, airplane companies can decrease the prices if they want to increase the sales.
- Airline companies use many different variables to determine the flight ticket prices that indicates whether the travel is during the holidays, the number of free seats in the plane etc., or even in which month it is, some of the variables are observed, but some of them are hidden.
- In this context, buyers are trying to find the right day to buy the ticket, and on the contrary, the airplane companies are trying to keep the overall revenue as high as possible. Airline companies have the freedom to change the flight ticket prices at any moment. Travelers can save money if they choose to buy a ticket when its price is the lowest.