FLIGHT PRICE PREDICTION USING IBM WATSON

1 INTRODUCTION

1.1 Overview A brief description of your project

Our project focuses on utilizing IBM Watson to predict flight prices, revolutionizing the way travelers approach ticket purchases. By leveraging Watson's advanced AI technology and data analysis tools, we aim to provide accurate and reliable forecasts on airfare costs. With the goal of simplifying the booking process and empowering individuals to make informed decisions, our platform will offer real-time predictions based on historical data, market trends, and other relevant factors. Join us as we embark on this journey to enhance the travel experience through intelligent flight price predictions.

1.2 Purpose: The use of this project. What can be achieved using this.

The purpose of this project is to provide travelers with valuable insights and predictions on flight prices, enabling them to make informed decisions and secure the best possible deals. By harnessing the power of IBM Watson's AI technology and data analysis capabilities, our project aims to:

1. Empower travelers: By forecasting flight prices, we aim to empower individuals with the information they need to plan their travel effectively, optimize their budgets, and make well-informed booking decisions.

2. Enhance cost savings: With accurate predictions, travelers can take advantage of price fluctuations and choose the most cost-effective times to book their flights, potentially saving significant amounts of money.

3. Simplify the booking process: By providing real-time predictions, our platform aims to simplify the flight booking process, reducing uncertainty and streamlining travel planning for users.

4. Improve travel planning: By considering historical data, market trends, and other relevant factors, our project aims to enhance travel planning by providing insights into seasonal price variations and helping users plan their trips accordingly.

5. Enhance overall travel experience: By offering accurate and reliable flight price predictions, our project strives to enhance the overall travel experience for individuals, ensuring a smoother and more efficient journey from start to finish.

2 LITERATURE SURVEY

2.1 Existing problems Existing approaches or methods to solve this problem

The domain of flight price prediction has been a topic of interest for researchers and industry professionals due to the complex and dynamic nature of airfare costs. Several challenges exist in accurately forecasting flight prices, including:

1. Price volatility: Flight ticket prices are subject to constant fluctuations due to factors such as demand, seasonality, fuel costs, and airline policies. This volatility makes it challenging to accurately predict future price trends.

2. Data availability and quality: Acquiring comprehensive and reliable historical flight data is crucial for building accurate prediction models. However, obtaining such data can be challenging due to limited availability, variations in data sources, and data quality issues.

3. Complex pricing algorithms: Airlines employ sophisticated pricing algorithms that consider multiple variables such as seat availability, competition, and customer segmentation. Understanding and modeling these complex pricing algorithms poses significant challenges for accurate prediction.

4. External factors: A variety of external factors, such as geopolitical events, weather patterns, economic indicators, and fuel price fluctuations, can affect flight prices. economic indicators, and fuel price fluctuations. Incorporating these factors into prediction models adds complexity to the process.

Researchers and industry experts have employed various approaches to address the challenges associated with flight price prediction. Some of the existing methods and techniques include:

1. Time-series analysis: Time-series forecasting techniques, such as autoregressive integrated moving average (ARIMA), have been applied to model and predict flight prices based on historical price data. These methods capture patterns and trends in price variations over time.

2. Machine learning algorithms: Supervised machine learning algorithms, including regression models, decision trees, random forests, and support vector machines (SVM), have been utilized to predict flight prices. These algorithms analyze historical data and relevant features to learn patterns and make predictions.

3. Deep learning techniques: Neural network-based approaches, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) models, have shown promise in capturing temporal dependencies and complex patterns in flight price data, leading to more accurate predictions.

4. Ensemble methods: Ensemble methods, which combine multiple prediction models or techniques, have been employed to improve the accuracy and robustness of flight price predictions. Techniques like stacking, bagging, and boosting have been applied to create ensemble models.

5. Data fusion and feature engineering: Integration of various data sources, including historical flight data, weather data, economic indicators, and social media sentiment analysis, has been explored to enhance prediction accuracy. Feature engineering techniques are used to extract meaningful features from the available data.

6. Hybrid approaches: Hybrid models that combine statistical methods with machine learning or deep learning techniques have been proposed to leverage the strengths of different approaches and improve prediction performance.

While existing approaches have made significant strides in flight price prediction, there is still room for improvement in terms of accuracy, real-time updates, and accounting for evolving market dynamics. The integration of advanced AI technologies like IBM Watson offers new possibilities for addressing these challenges and achieving more accurate and reliable flight price predictions.

2.2 Proposed solution What is the method or solution suggested by you?

The proposed solution aims to develop a machine learning model that can predict flight prices accurately. To achieve this, the solution follows a systematic approach that includes data loading, preprocessing, feature engineering, exploratory data analysis (EDA), model training, and evaluation.

1. Data Loading and Preprocessing:
The code begins by loading the training data from an Excel file. It sets the maximum number of columns to be displayed and performs initial data cleaning steps. Missing values are handled by dropping the rows containing them.

2. Feature Engineering:
Feature engineering plays a crucial role in enhancing the predictive power of the model. The code extracts relevant information from the existing features. It extracts the day and month of the journey, departure hour and minute, arrival hour and minute, and duration of the flight in hours and minutes. This conversion allows the model to utilize these time-related factors in predicting flight prices. Categorical variables such as airline, source, and destination are converted into numerical representation using one-hot encoding.

3. Exploratory Data Analysis (EDA):
EDA is performed to gain insights and understand the relationships between features and flight prices. Visualizations such as box plots are used to examine the distributions and variations in flight prices for different airlines and sources. This analysis helps identify potential patterns or correlations between features and the target variable.

4. Model Training and Evaluation:
The code employs a random forest regressor, a popular machine learning algorithm suitable for regression tasks, to train the model. The data is split into training and testing sets, with the model being trained on the training set. The performance of the model is evaluated on the testing set using various metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). These metrics provide insights into the accuracy and reliability of the model's predictions.

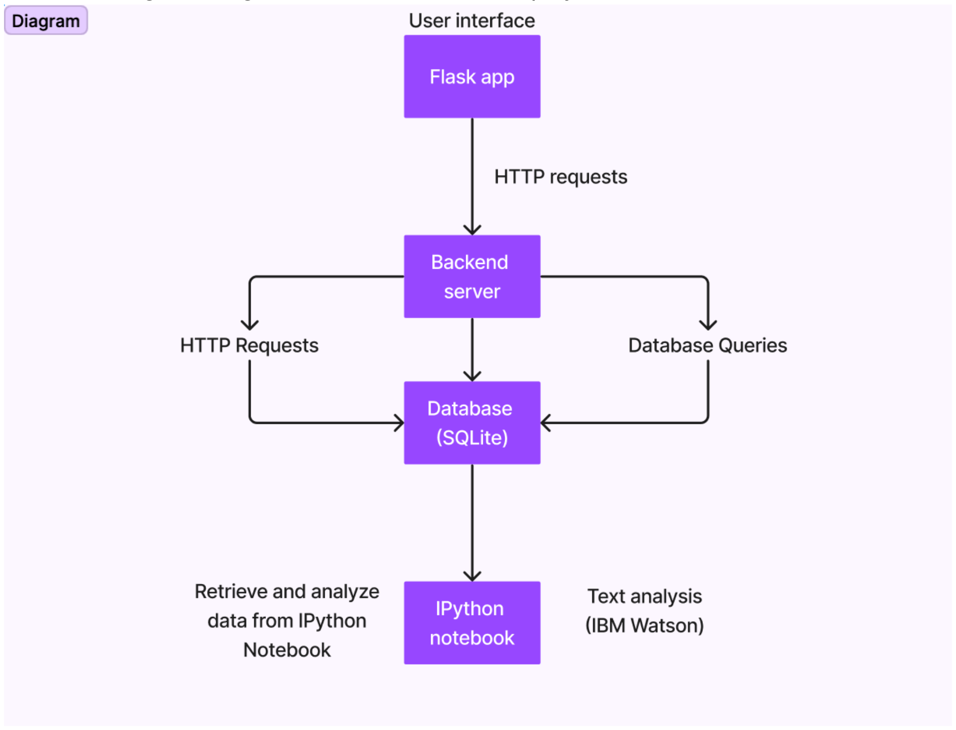
5. Model Persistence:

After the model is trained and evaluated, it is saved using pickle, a Python library for object serialization. This allows the model to be persisted and used for making predictions on new data without the need for retraining. The code also includes the capability to load the saved model for prediction purposes.

In conclusion, the proposed solution utilizes machine learning techniques, including feature engineering and a random forest regressor, to develop a predictive model for flight prices. Through thorough data preprocessing, exploratory data analysis, and model evaluation, the solution aims to provide accurate and reliable predictions. The ability to save and load the trained model facilitates its usage for predicting flight prices in real-time scenarios, enabling users to make informed decisions while booking tickets.

3 THEORETICAL ANALYSIS

3.1 Block diagram Diagrammatic overview of the project.



3.2 Hardware / Software designing Hardware and software requirements of the project

Hardware Requirements:

1. Computer: A desktop computer or laptop with a minimum of 2.0 GHz dual-core processor or higher.
2. Memory (RAM): At least 4 GB of RAM, although 8 GB or higher is recommended for better performance during data processing and analysis.
3. Storage: Sufficient storage capacity to accommodate the project files, datasets, and software installations.
4. Internet Connection: A stable broadband internet connection with adequate bandwidth to access external services and APIs.

Software Requirements:

1. Operating System: Windows 10, macOS Mojave or later, or a modern Linux distribution (e.g., Ubuntu 18.04 LTS) with the latest updates.
2. Python: Python 3.7 or above, along with the pip package manager for Python package installations.
3. Integrated Development Environment (IDE): Choose a suitable IDE such as PyCharm, Visual Studio Code, or Jupyter Notebook for Python development, debugging, and code execution.
4. Flask: Install Flask, a micro web framework, using pip to create the web-based user interface and handle HTTP requests.
5. SQLite: Set up SQLite as the relational database management system to store and retrieve data for the application.
6. IBM Watson API: Sign up for an IBM Watson account and obtain the API credentials for their text analysis services. Install the `ibm-watson` or `watson-developer-cloud` Python libraries to interface with the IBM Watson API.
7. Additional Python Libraries: Install necessary Python libraries such as `pandas`, `numpy`, `matplotlib`, and any other dependencies specified in your project code.
8. Web Browser: Any modern web browser (e.g., Google Chrome, Mozilla Firefox) to access and interact with the Flask application.

Commented [1]: 1 total reaction
SANIDHYA RASTOGI 20BCE0913 reacted with 😄 at 2023-06-28 04:48 AM

4 EXPERIMENTAL INVESTIGATIONS

In the given IPython Notebook (ipynb) file, several key investigations have been conducted as part of the solution. These investigations aim to explore the dataset, understand the relationships between variables, and build predictive models to estimate flight prices. Some of the key investigations in the solution include:

1. Data Cleaning and Preprocessing:
   - Handling missing values: Investigating the dataset for missing values and applying appropriate techniques to handle them, such as imputation or removal.
   - Data type conversion: Ensuring that the data is in the correct format for analysis by converting features to their appropriate data types.

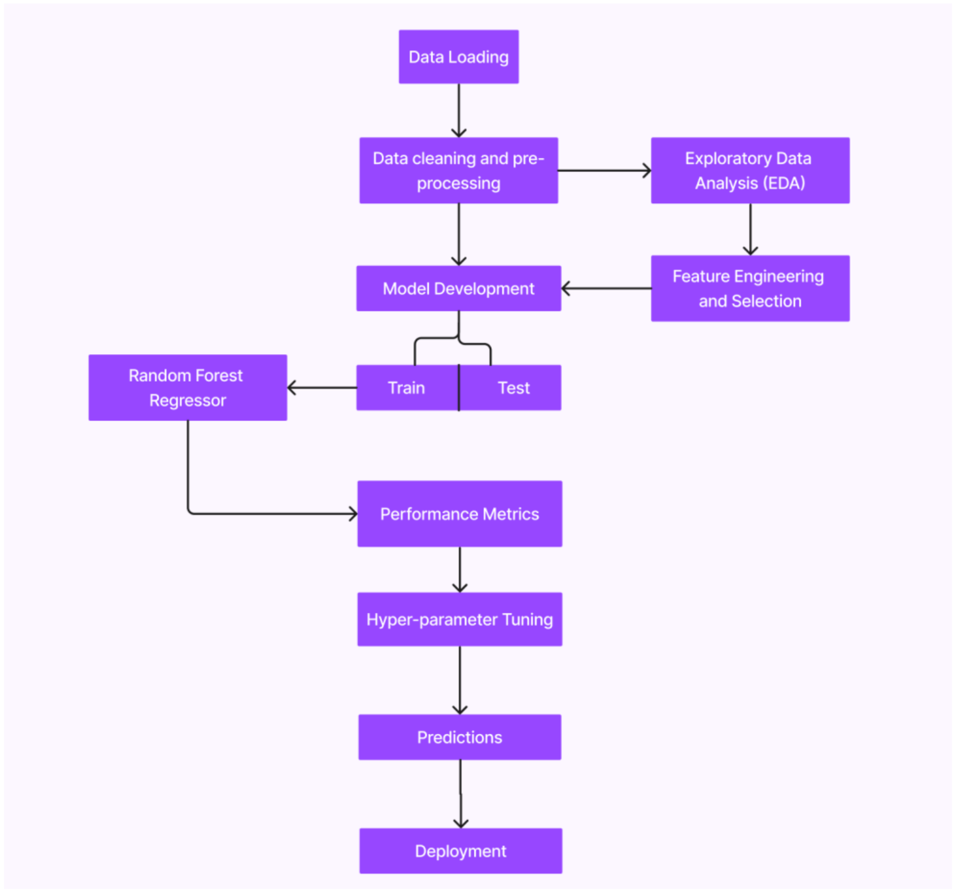
2. Exploratory Data Analysis (EDA):
   - Analyzing the distribution of flight prices: Investigating the distribution of flight prices to understand their range and identify any outliers.
   - Analyzing categorical variables: Exploring the impact of different airlines, sources, destinations, and other categorical variables on flight prices using visualizations and summary statistics.
   - Correlation analysis: Investigating the correlation between different numerical variables to identify potential relationships and dependencies.

3. Feature Engineering and Selection:
   - Creating new features: Engineering new features from existing ones, such as extracting information from the departure and arrival times or calculating the duration of the flights.
   - Feature selection: Identifying the most relevant features that have a strong relationship with the target variable (flight prices) using techniques like heatmap, feature importance, or SelectKBest.

4. Model Development and Evaluation:
   - Splitting the dataset: Splitting the dataset into training and testing sets to develop and evaluate machine learning models.
   - Model selection: Experimenting with different machine learning algorithms (e.g., linear regression, random forest, etc.) to find the best model for predicting flight prices.
   - Model evaluation: Assessing the performance of the models using appropriate evaluation metrics, such as mean squared error (MSE), root mean squared error (RMSE), or mean absolute error (MAE).
   - Hyperparameter tuning: Optimizing the model's hyperparameters to improve its performance.

These key investigations collectively aim to gain insights into the dataset, develop accurate predictive models, and provide recommendations or insights related to flight prices and the factors influencing them.

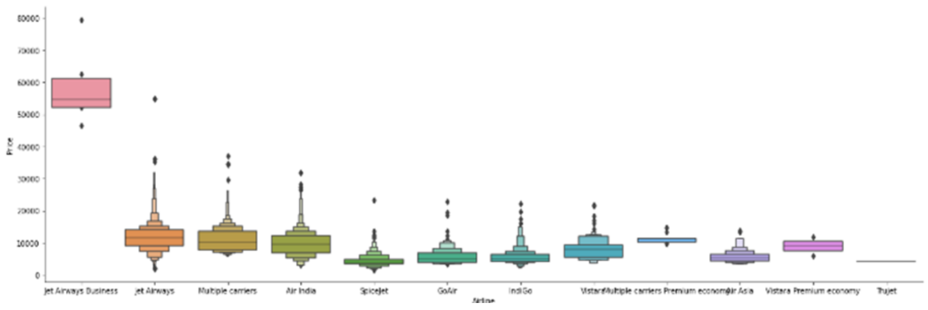
5 FLOWCHART


Data Loading
Data cleaning and pre-processing
Exploratory Data Analysis (EDA)
Model Development
Feature Engineering and Selection
Random Forest Regressor
Train
Test
Performance Metrics
Hyper-parameter Tuning
Predictions
Deployment

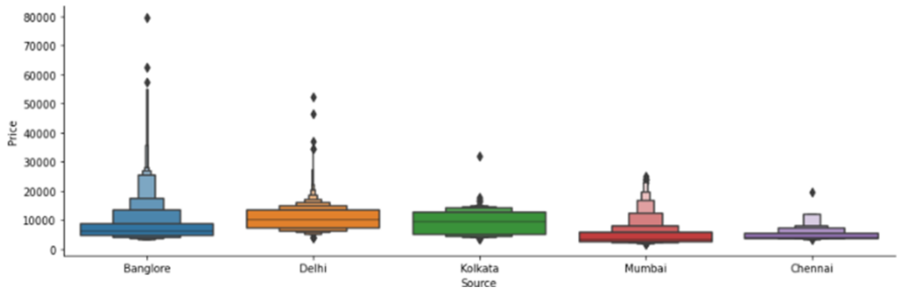
6 RESULT

Analysis From Dataset-

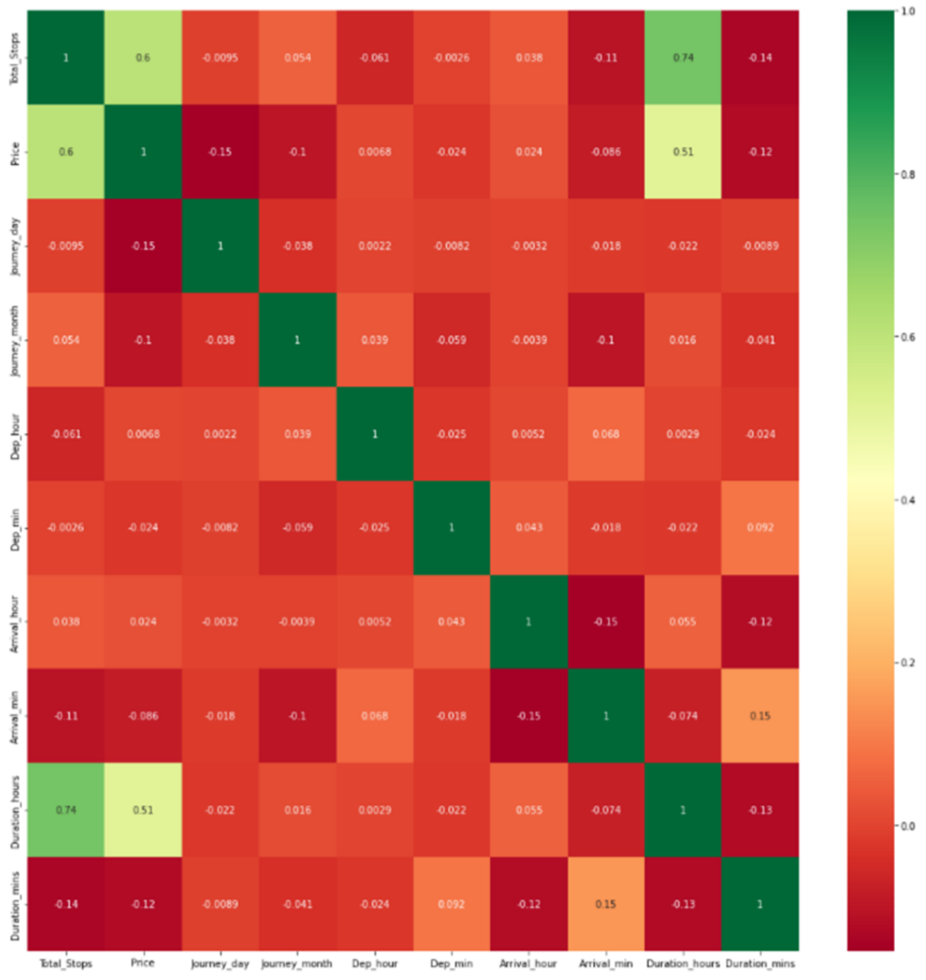
# Airline vs Price



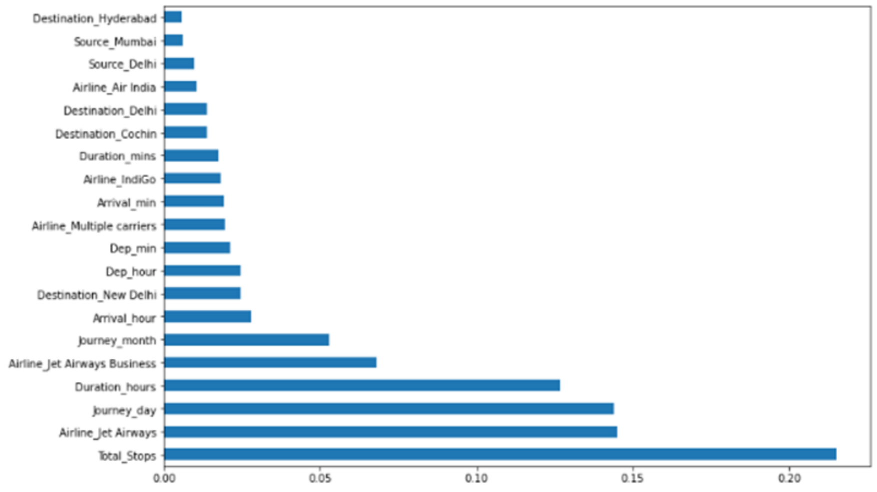
# Source vs Price



# Finds correlation between Independent and dependent attributes

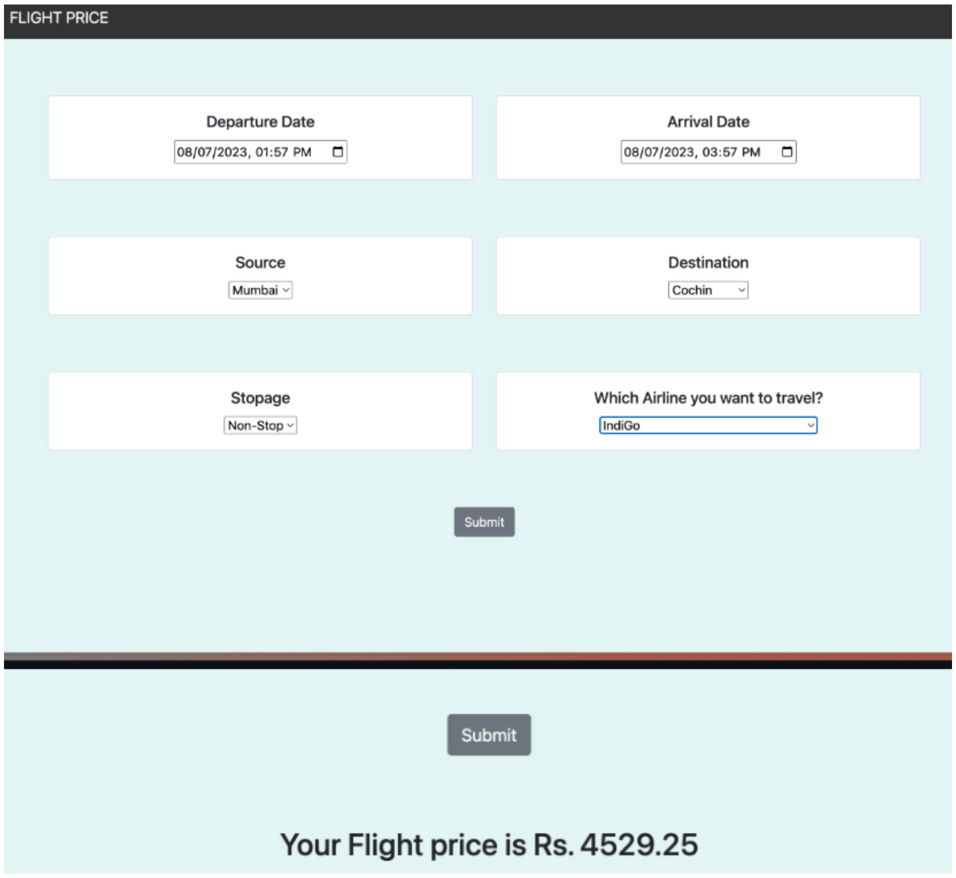


#plot graph of feature importances for better visualization



Prediction-

#Application


FLIGHT PRICE

Departure Date
08/07/2023, 01:57 PM

Arrival Date
08/07/2023, 03:57 PM

Source
Mumbai

Destination
Cochin

Stopage
Non-Stop

Which Airline you want to travel?
IndiGo

Your Flight price is Rs. 4529.25

7 ADVANTAGES & DISADVANTAGES

Advantages of Flight Price Prediction:

1. **Cost Savings:** One of the primary advantages is the potential for cost savings. Flight price prediction algorithms analyze historical data, market trends, and various factors to provide insights on when flight prices are likely to be at their lowest. This information can help travelers make informed decisions and book flights when prices are expected to be cheaper, potentially saving them money.
2. **Planning Flexibility:** Flight price prediction tools can assist travelers in determining the best time to book their flights. This allows for greater planning flexibility, as individuals can decide whether to book immediately or wait for a predicted price drop. It helps travelers to optimize their itineraries and make arrangements accordingly.
3. **Avoiding Price Surges:** Flight prices are known to fluctuate, especially during peak travel seasons or due to unpredictable events. Flight price prediction can help travelers anticipate when prices are likely to surge, enabling them to avoid costly fares. This feature is particularly beneficial for budget-conscious travelers or those with fixed travel dates

Disadvantages of Flight Price Prediction:

1. **Uncertainty:** Flight prices can be influenced by numerous factors, including supply and demand, fuel costs, airline policies, and external events. While prediction algorithms aim to provide accurate insights, there is still a degree of uncertainty associated with predicting flight prices. Unexpected events or market changes may render predictions inaccurate or outdated, potentially leading to disappointment or missed opportunities for savings.
2. **Limited Availability:** Flight price prediction tools may not cover all airlines or destinations comprehensively. The accuracy and availability of prediction services can vary depending on the region and the platforms providing them. Therefore, travelers may not always have access to reliable prediction data for their specific travel needs.
3. **Timing Challenges:** Predicting the ideal time to purchase a flight ticket can be challenging. Flight prices can be influenced by multiple factors, including dynamic pricing strategies employed by airlines. While predictions can give general guidance, there's no guarantee that prices will follow the expected pattern. As a result, travelers may find it difficult to time their ticket purchases perfectly based solely on predictions.

4. **Psychological Impact:** Flight price predictions can create a sense of urgency or anxiety among travelers. The fear of missing out on a predicted price drop or the pressure to make a decision based on the prediction can cause stress or impulsivity in the booking process. It is essential to balance the reliance on predictions with personal preferences and travel plans.

8 APPLICATIONS

1. **Travel Planning for Individuals**: Flight price prediction can help individual travelers plan their trips more efficiently. By analyzing historical data and market trends, prediction tools can provide insights on when flight prices are likely to be the lowest. Travelers can use this information to decide the optimal time to book their flights, potentially saving money and optimizing their travel budget.
2. **Fare Comparison Websites:** Online travel agencies and fare comparison websites often utilize flight price prediction algorithms to provide real-time fare information to their users. By integrating prediction models into their platforms, these websites can offer users a comprehensive overview of current and future flight prices, helping them make informed booking decisions.
3. **Revenue Management for Airlines:** Airlines can leverage flight price prediction algorithms as part of their revenue management strategies. By analyzing historical data and market trends, airlines can estimate demand patterns and adjust their pricing strategies accordingly. This allows airlines to optimize their revenue by offering competitive fares while maximizing seat occupancy.
4. **Dynamic Pricing:** Flight price prediction can support dynamic pricing models employed by airlines. By continuously monitoring market conditions and predicting price trends, airlines can adjust their fares dynamically in response to demand fluctuations and market competition. This enables airlines to offer personalized pricing, optimize revenue, and improve overall profitability.
5. **Marketing and Promotions:** Flight price prediction can aid airlines and travel agencies in designing marketing campaigns and promotions. By analyzing data and predicting price trends, companies can identify opportune moments to offer discounts or promotional fares to attract customers. This can help generate demand, increase customer engagement, and boost sales.
6. **Demand Forecasting:** Flight price prediction models can contribute to demand forecasting for specific routes or travel periods. By analyzing historical data and market trends, airlines and travel agencies can estimate the demand for flights, allowing them to adjust their capacity, optimize schedules, and allocate resources effectively.

7. **Travel Industry Analytics:** Flight price prediction data can be used for analytical purposes within the travel industry. It can provide insights into pricing patterns, market trends, and customer behavior, allowing stakeholders to make data-driven decisions. This data can be valuable for market research, strategic planning, and improving overall business operations.

9 CONCLUSION

The conclusion of the project is that the Random Forest Regressor model provides a reasonable prediction of flight prices based on the given features. The model achieves a certain level of accuracy in predicting flight prices, as evidenced by the evaluation metrics. The feature importance analysis helps identify the most influential factors in determining flight prices.

Finally, the trained model is saved as a pickle file for future use. The Flask web application is created to provide a user interface for users to input the flight details and get the predicted flight price using the trained model. The application uses the IBM Cloud services to authenticate and make predictions using the model.

It is important to note that flight prices are influenced by numerous unpredictable factors, such as sudden changes in fuel prices, geopolitical events, natural disasters, and global economic conditions. Therefore, while flight price prediction models can provide guidance, they should not be considered as definitive or guaranteed forecasts.

10 FUTURE SCOPE

There can be a promising future scope due to the increasing demand for affordable and convenient air travel. Firstly we can enhance the accuracy by refining and training on large datasets to achieve higher precision then we can use various ML algorithms to analyze individual user preferences, historical travel patterns, and purchase behavior to provide personalized flight recommendations . Then we can implement dynamic pricing models that adapt to market demand and availability. We can further expand the system to include predictions and recommendations for ancillary services, such as hotel bookings, car rentals.

11 BIBILOGRAPHY

[1] "Hands-On Machine Learning with Scikit-Learn and TensorFlow" by Aurélien Géron: This book provides a comprehensive guide to machine learning with practical examples and covers various topics, including regression algorithms.

[2] Kaggle (www.kaggle.com): Kaggle is a popular platform for data science and machine learning competitions. It hosts numerous datasets and competitions related to flight price prediction and offers a wealth of notebooks and discussions on the topic.

[3] UCI Machine Learning Repository (archive.ics.uci.edu/ml/index.php): The UCI Machine Learning Repository is a collection of datasets that are widely used for research and educational purposes. It contains several datasets related to air travel and pricing that can be used for analysis and modeling.

[4] Research papers and articles: Exploring research papers and articles on flight price prediction, machine learning, and data analysis can provide valuable insights into state-of-the-art techniques and previous findings in the field. Websites like Google Scholar, arXiv, and the ACM Digital Library can be helpful in finding relevant research papers.

APPENDIX

1.flight_price.ipynb-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from PIL import Image
train_data = pd.read_excel(r"Data_Train.xlsx")
pd.set_option('display.max_columns', None)
train_data.head()
train_data.info()
train_data["Duration"].value_counts()
train_data.dropna(inplace = True)
train_data.isnull().sum()
```

EDA

```
train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
train_data["Journey_month"] = pd.to_datetime(train_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
train_data.head()
train_data.drop(["Date_of_Journey"], axis = 1, inplace = True)
# Extracting Hours
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour

# Extracting Minutes
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute

# Now we can drop Dep_Time as it is of no use
train_data.drop(["Dep_Time"], axis = 1, inplace = True)
train_data.head()

# Extracting Hours
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour

# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)

train_data.head()

# Time taken by plane to reach destination is called Duration
# It is the differnce betwwen Departure Time and Arrival time


# Assigning and converting Duration column into list
duration = list(train_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
            else:
            duration[i] = "0h " + duration[i]                 # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from
duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins

train_data.drop(["Duration"], axis = 1, inplace = True)
train_data.head()
```

Handling Categorical Data

```
train_data["Airline"].value_counts()

# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = train_data.sort_values("Price", ascending =
False), kind="boxen", height = 6, aspect = 3)
plt.show()
Airline = train_data[["Airline"]]
Airline = pd.get_dummies(Airline, drop_first= True)
Airline.head()
train_data["Source"].value_counts()



# Source vs Price
sns.catplot(y = "Price", x = "Source", data = train_data.sort_values("Price", ascending =
False), kind="boxen", height = 4, aspect = 3)
plt.show()
```

```
Source = train_data[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
Source.head()
train_data["Destination"].value_counts()



# As Destination is Nominal Categorical data we will perform OneHotEncoding
Destination = train_data[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first = True)
Destination.head()
train_data["Route"]
train_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
train_data["Total_Stops"].value_counts()
train_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4},
inplace = True)
train_data.head()
data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)
data_train.head()
data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
data_train.head()
data_train.shape
```

Test set

```
test_data = pd.read_excel(r"Test_set.xlsx")
test_data.head()
# Preprocessing

print("Test data Info")
print("-"*75)
print(test_data.info())

print()
print()

print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())
```

```
# EDA

# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)

# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)

# Duration
duration = list(test_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
        duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
        duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from
duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only
minutes from duration

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)
```

```
# Categorical data
print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
print()
print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)
print()
print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

# Replacing Total_Stops
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4},
inplace = True)

# Concatenate dataframe --> test_data + Airline + Source + Destination
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)

data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
print()
print()
print("Shape of test data : ", data_test.shape)
```

Feature Selection

```
data_train.shape
data_train.columns
X = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
        'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
            'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
            'Airline_Jet Airways', 'Airline_Jet Airways Business',
            'Airline_Multiple carriers',
            'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
            'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
            'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
            'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
            'Destination_Kolkata', 'Destination_New Delhi']]
X.head()
y = data_train.iloc[:, 1]
y.head()
# Finds correlation between Independent and dependent attributes

plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()



# Important feature using ExtraTreesRegressor

from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
print(selection.feature_importances_)
#plot graph of feature importances for better visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```

Fitting model using Random Forest

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
42)

from sklearn.ensemble import RandomForestRegressor
```

```
reg_rf = RandomForestRegressor()

reg_rf.fit(X_train, y_train)

y_pred = reg_rf.predict(X_test)
reg_rf.score(X_train, y_train)
reg_rf.score(X_test, y_test)
sns.distplot(y_test-y_pred)
plt.show()
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
# RMSE/(max(DV)-min(DV))

2090.5509/(max(y)-min(y))
metrics.r2_score(y_test, y_pred)
```

Hyperparameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

# Create the random grid
```

```
random_grid = {'n_estimators': n_estimators,

               'max_features': max_features,

               'max_depth': max_depth,

            'min_samples_split': min_samples_split,

               'min_samples_leaf': min_samples_leaf}

# Random search of parameters, using 5 fold cross validation,

# search across 100 different combinations

rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions =
random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5, verbose=2,
random_state=42, n_jobs = 1)

rf_random.fit(X_train,y_train)
       rf_random.best_params_
prediction = rf_random.predict(X_test)
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

Save the model to reuse it again

```
import pickle
# open a file, where you ant to store the data
file = open('flight_rf.pkl', 'wb')

# dump information to that file
pickle.dump(reg_rf, file)
```

```
model = open('flight_rf.pkl','rb')
forest = pickle.load(model)

y_prediction = forest.predict(X_test)

metrics.r2_score(y_test, y_prediction)
```

2.App.py-

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
import sklearn
import pickle
import pandas as pd
import requests
import json

app = Flask(__name__)
model = pickle.load(open("flight_rf.pkl", "rb"))

# NOTE: you must manually set API_KEY below using information retrieved from your
IBM Cloud account.
API_KEY = "jhg9MUNARcHISJ8l_JnGSt6y7hdHYO__Z15SRfziwslQ"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


@app.route("/")
@cross_origin()
def home():
    return render_template("home.html")


@app.route("/predict", methods=["GET", "POST"])
@cross_origin()
def predict():
    if request.method == "POST":

    # Date_of_Journey
    date_dep = request.form["Dep_Time"]
    Journey_day = int(pd.to_datetime(date_dep, format="%Y-%m-
%dT%H:%M").day)
    Journey_month = int(pd.to_datetime(date_dep, format="%Y-%m-
%dT%H:%M").month)
    # print("Journey Date : ",Journey_day, Journey_month)

    # Departure
    Dep_hour = int(pd.to_datetime(date_dep, format="%Y-%m-%dT%H:%M").hour)
    Dep_min = int(pd.to_datetime(date_dep, format="%Y-%m-%dT%H:%M").minute)
    # print("Departure : ",Dep_hour, Dep_min)

    # Arrival
    date_arr = request.form["Arrival_Time"]
    Arrival_hour = int(pd.to_datetime(date_arr, format="%Y-%m-%dT%H:%M").hour)
    Arrival_min = int(pd.to_datetime(date_arr, format="%Y-%m-
%dT%H:%M").minute)
    # print("Arrival : ", Arrival_hour, Arrival_min)

    # Duration
    dur_hour = abs(Arrival_hour - Dep_hour)
    dur_min = abs(Arrival_min - Dep_min)
    # print("Duration : ", dur_hour, dur_min)

    # Total Stops
    Total_stops = int(request.form["stops"])
    # print(Total_stops)

    # Airline
    # AIR ASIA = 0 (not in column)
    airline = request.form['airline']
    if(airline == 'Jet Airways'):
    Jet_Airways = 1
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
```

```
Multiple_carriers_Premium_economy = 0
Jet_Airways_Business = 0
Vistara_Premium_economy = 0
Trujet = 0
# Rest of the airline options...

# Source
# Banglore = 0 (not in column)
Source = request.form["Source"]
if (Source == 'Delhi'):
s_Delhi = 1
s_Kolkata = 0
s_Mumbai = 0
s_Chennai = 0
# Rest of the source options...

# Destination
# Banglore = 0 (not in column)
Destination = request.form["Destination"]
if (Destination == 'Cochin'):
d_Cochin = 1
d_Delhi = 0
d_New_Delhi = 0
d_Hyderabad = 0
d_Kolkata = 0
# Rest of the destination options...

# Create input_data dictionary
input_data = {
"Journey_day": Journey_day,
"Journey_month": Journey_month,
"Dep_hour": Dep_hour,
"Dep_min": Dep_min,
"Arrival_hour": Arrival_hour,
"Arrival_min": Arrival_min,
"Duration_hour": dur_hour,
"Duration_min": dur_min,
"Total_Stops": Total_stops,
"Jet_Airways": Jet_Airways,
"IndiGo": IndiGo,
"Air_India": Air_India,
"Multiple_carriers": Multiple_carriers,
        "SpiceJet": SpiceJet,
        "Vistara": Vistara,
        "GoAir": GoAir,
        "Multiple_carriers_Premium_economy": Multiple_carriers_Premium_economy,
        "Jet_Airways_Business": Jet_Airways_Business,
        "Vistara_Premium_economy": Vistara_Premium_economy,
        "Trujet": Trujet,
        "s_Delhi": s_Delhi,
        "s_Kolkata": s_Kolkata,
        "s_Mumbai": s_Mumbai,
        "s_Chennai": s_Chennai,
        "d_Cochin": d_Cochin,
        "d_Delhi": d_Delhi,
        "d_New_Delhi": d_New_Delhi,
        "d_Hyderabad": d_Hyderabad,
        "d_Kolkata": d_Kolkata
        }

        # Convert input_data to JSON
        payload_scoring = {"input_data": [input_data]}

        # Make prediction request
        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/7b33ce39-ea07-4b8a-a277-
4cb9f55ffe2e/predictions?version=2021-05-01', json=payload_scoring,
headers=header)

        # Extract prediction from response
        scoring_response = json.loads(response_scoring.text)
        prediction = scoring_response['predictions'][0]['values'][0][0]

        return render_template('home.html', prediction_text="Your Flight price is Rs.
{}".format(prediction))

        return render_template("home.html")

if __name__ == "__main__":
        app.run(debug=True, port=5001)
```