HIGH LEVEL DESIGN (HDD)

Flight fare prediction using machine learning

Submitted by
Laizin V
laizin2107@gmail.com

Document version control

date issued	version	description	author
30/04/2023	1.0	Initial HLD v 1.0	Laizin v

High level design (HLD)

Contents

Abstract

This project aims to predict flight prices using machine learning techniques. The proposed solution involves collecting and analyzing flight data from our dataset which is over 450000 flight records, including historical flight prices, weather conditions, flight routes, and other relevant factors.

The project will use machine learning algorithm , that is Random forest regressor, to predict flight prices accurately. The developed system will help travelers plan their trips efficiently and save money by predicting the best time to book their flights. The project's success will be evaluated based on the accuracy and efficiency of the developed model, and its ability to provide valuable insights and predictions for travelers.

users have to input a few parameters in order to predict the flight fare

High level design (HLD)

1. Introduction

1.1 why High-level document

The purpose of a High-Level Design (HLD) document is to provide an overview of the proposed system's architecture and functionality. The HLD document outlines the system's components, modules, interfaces, and their interactions with each other. It describes the overall design approach, design patterns, and design principles used in the system's development. The HLD document typically includes below details

1. implements the design aspects of the app
2. explains the architecture of the project
3. explains the core technology behind the app
4. explains non functional attributes
    - reliability
    - maintainability
    - security
    - use cases

1.2 Scope

The scope of a High-Level Design (HLD) document refers to the boundaries and limitations of the system being designed. It defines what the system will and will not do, what features and functionalities it will have, and what user requirements it will meet. The scope of the HLD document is critical in ensuring that the system meets the intended purpose and that the stakeholders' expectations are met

1.3 Definitions

IDE - Integrated development environment
MAE - Mean absolute error
MSE- Mean squared error
AWS- Amazon web server
VSCode - Visual studio code
CSV- comma seperated values

2. General description

2.1 Product perspective

This machine learning based flight fare prediction web app can be seen as a tool that provides users with valuable insights and information related to air travel. It serves as web app for users who want to plan their trips and make informed decisions about their travel expenses.

2.2 Problem statement

To create a machine learning algorithm that can predict the flight fare for a given date by comparing the date and some other features such as class, airline , duration , source and destination. we have to train this algorithm by using the dataset we have prepared . we need to find the maximum performance from the algorithm . we can use any regression algorithm to train the model

2.3 Proposed solution

Proposed solution is a machine learning algorithm trained with random forest regressor from scikit-learn . and we have used a dataset of 450000+ flight fare datas

2.4 Data requirements

To achieve maximum performance from the regression algorithm , we need a huge amount of previous flight price data . we can extract the data by doing web scrapping and also we can find from famous data sources like kaggle

- this dataset is from kaggle website
- dataset is in CSV format , so we have used pandas for dataset manipulations
- Dataset contains both numerical and categorical data , so we have also used some data transformation techniques
- in our dataset we have datas of date, class, flight code , day of the week, Airline , Source , destination, departure time , arrival time , total stops between source and destination , duration of flight , days left and fare
- some columns should be transformed and some columns should be avoided , all these operations are explained in detail

2.5 Tools used

PYTHON

Python is used as the programming language to train the algorithm and as the backend for the web app . literally python is used for all operations with differernt libraries

pyhton is well surrounded with lots of libraries which makes our job easy. the libraries are explained in coming pages

High level design (HLD)

PANDAS

as the input data is in CSV format , we have used pandas library for all kind of data operations

Pandas is a powerful data manipulation library for Python that provides data structures for efficiently storing and analyzing large amounts of data. It is often used in data science and machine learning applications to clean, preprocess, and analyze data.

SCIKIT-LEARN

Scikit-learn (also known as sklearn) is a popular machine learning library for Python that provides a wide range of tools and algorithms for building predictive models.

It is built on top of other scientific computing libraries, such as NumPy, SciPy, and Matplotlib, and is designed to be easy to use and efficient for large datasets. here the random forest regressor is what we have used to train our model

MATPLOTLIB

Matplotlib is a data visualization library for Python that provides a wide range of tools for creating high-quality plots and graphs. It is one of the most widely used data visualization libraries in the scientific and data science communities.

VS code

Visual studio code (VS Code) is used as the IDE (integrated developer environment) for this project

NUMPY

NumPy is a numerical computing library for Python that provides a wide range of tools for working with arrays, matrices, and numerical data. It is one of the foundational libraries in the Python data science ecosystem, and is widely used for numerical computing, scientific computing, and data analysis.

FLASK

Flask is a lightweight and flexible web framework for Python that allows developers to easily build web applications. It is designed to be simple and easy to use, and provides a wide range of tools for handling web requests, managing user sessions, and interacting with databases.

2.6 Objective

To develop a machine learning model that accurately predicts flight fares based on historical flight data, and to integrate this model into a user-friendly app that enables travelers to easily search for and compare flight prices across multiple airlines and certain routes

The objective of this project is to provide travelers with a tool that empowers them to make informed decisions about their travel plans, and helps them save money and time by identifying the best flight options for their needs.

This web app have its own capabilities and work within limits . as the resources are less, we can not integrate all the flight routes in the world . so we have added only a certain number of airports in india . so the users can use the app to search within these routes .

2.7 Constraints

The accuracy of the machine learning model is highly dependent on the quality and quantity of the data used to train it. The availability of historical flight data may be limited or subject to restrictions, and the data may contain errors or biases that could affect the model's accuracy . users have to land on the home page and just give some informations to get the predicted output.

3. Design details

3.1 Data collection and EDA

To make a better machine learning model , we need to make our dataset as bigger as possible and as accurate as possible . to train the model i have collected a dataset from below link

https://www.kaggle.com/datasets/yashdharme36/airfare-ml-predicting-flight-fares

- initially the dataset had 452087 rows of datas
- pandas library has used to data manipulations
- all process have been visualized in notebook with the help of matplotlib library

EDA

EDA involves analyzing and visualizing the data to gain insights and understanding of the underlying patterns, relationships, and anomalies in the data. EDA is usually the first step in the data analysis process, and it helps data scientists to identify potential problems, formulate hypotheses, and develop strategies for further analysis.

FEATURE SELECTION also done within EDA , we have explored the data in all aspects and found out the relations between all features and extracted the needed features only

High level design (HLD)

As per the visualizations and relations between features . we have done the below operations before splitting the data for train and test

- Extracted, the day of the week, date, month, and year separately from the DateTime object and dropped the date-of-journey column after adding 4 separated columns
- Changed the column name "class" to "classes" before class is keyword in python , there is a chance that may make some issues later . so we did not take any risk
- There were no null values presented in the dataset
- Then we have deleted the rows which having same datas , also called as duplicate rows
- Next we have visualixed the variation in price with different features in the dataset
- Its noticed that flight code has 1400+ unique values so we looked up the importance of flight code column . in order to avoid high complexity when encoding the columns , we have dropped the flight code column after confirming its related to class and airline columns . aslo Performed Shapiro-Wilk normality test to confirm

after these steps our dataframe had **440087** rows of data

3.2 Model training and evaluation

For training first we split the dataset into X and y . X is our features and y is our target variable

```
#SPLITTING THE FEATURES INTO TARGET VARIABLE AND FEATURES
X=df.drop(['Fare'],axis=1)
y=df['Fare']
```

High level design (HLD)

After creating X and y dataframes, we have then split the dataframes into train and test dataframes

```
# Splitting the Data into Training set and Testing Set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=42)
```

we have used train test split from scikit-learn library

Training pipeline

We have developed a pipeline using **pipeline** module from scikit-learn. we have implemented a column transformer that can transform categorical columns using **one hot encoder**
we will save the preprocessor to re use on the new dataframe .
we will use the **pickle** library to save the preprocessor

Training the model

to train the model , we have tested some regression algorithms to check which model will give us the best score. we ended up on the **random forest regressor .**

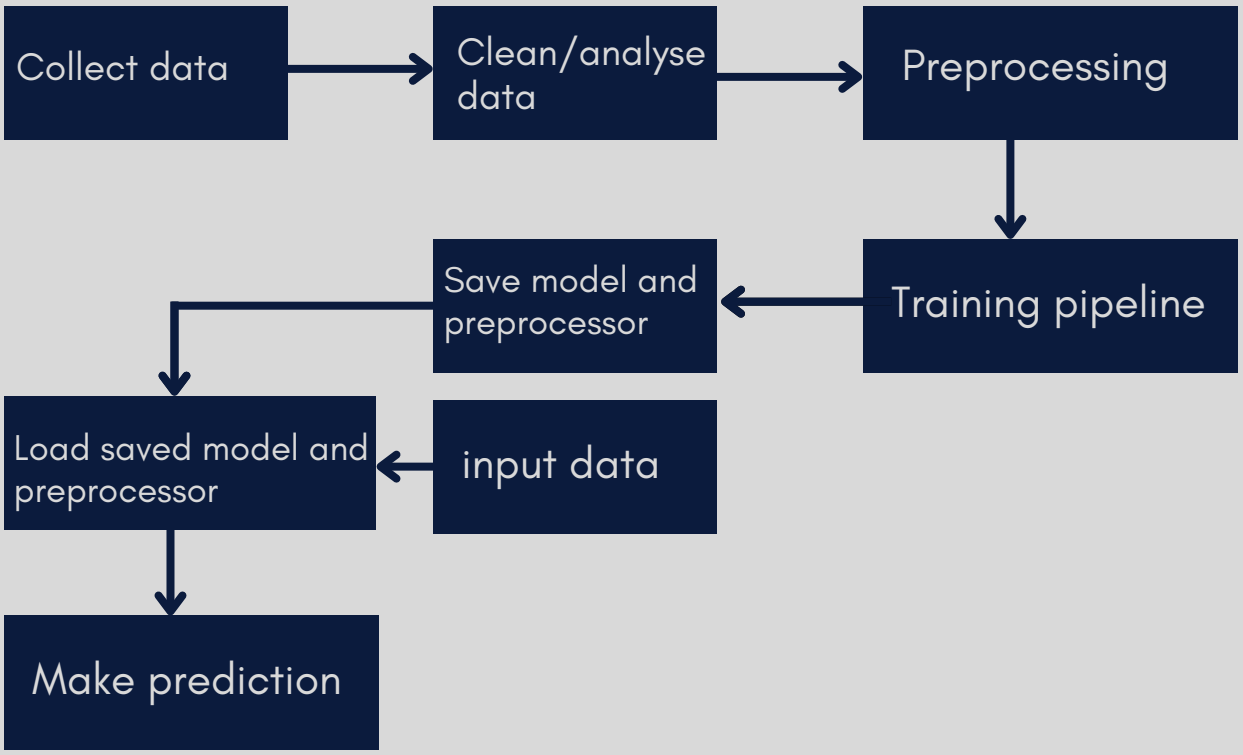
```
from sklearn.ensemble import RandomForestRegressor

# train a Random Forest model
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(x_train_processed, y_train)

# predict on test data
y_pred = rf.predict(x_test_processed)

# evaluate the model using mean squared error
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

We have evaluated the model based o the MSE value

High level design (HLD)

3.3 Deployment process

Collect data > clean data >preprocess data > split the data > train the model >save the model >



This process flow diagram shows the flow of the architecture of the model. from the collecting the dataset to making prediction with a n ew dataset

3.4 Error handling

To handle the errors , we have made the **CustomException** class that can be called in a **try except** block to log the exact error with file name and line number

In Python, the try/except block is used to handle exceptions that may occur during the execution of our code.

we can wrap the code that may raise an exception with the try statement. If an exception occurs, Python will immediately jump to the except block to handle the exception. The except block specifies the type of exception to handle (in this case, Some Exception Type) and assigns it to the variable **e**. You can then use this variable to handle the exception.

Logging

We have used python loggings to log the checkpoints throughout the program. it creates a folder called "logs" and then stores all the logs inside the folder with timestamp . so we can always check out the flow of the program

3.5 Performance and Reusability

Performance Considerations

To ensure that our Flask web app for predicting flight fares using machine learning is performant, we will take the following steps:

- Use a web server like Gunicorn or uWSGI to handle incoming requests, allowing us to handle multiple requests simultaneously and improve overall performance.
- Consider the size and complexity of our machine learning model, and use a smaller or simpler model if needed to avoid slowing down the app.
- Break our machine learning model up into smaller, more manageable components if needed.

Reusability Considerations

To ensure that our Flask web app for predicting flight fares using machine learning is reusable, we will take the following steps:

- Use a modular design pattern like MVC (Model-View-Controller) to separate our app's code into distinct components, making it easier to reuse parts of our app in other projects or modify our app as needed.

High level design (HLD)

We have made the code more modular , so that we can apply any machine learning problems or regression problems to this same solutions. although we need to apply some hyperparameter tuning in order to tune the model accuracy. we will also need to change the UI according to the new problem statement , but it will be of no problem if you are having a little bit of front end knowledge

3.6 Application capability

Our Flask web app allows users to predict the fare for a given flight using machine learning. Key features include:

- Flight Selection: Users can input the details of the flight they are interested in, such as the departure and arrival airports, travel dates, and number of passengers.
- Machine Learning Prediction: Our app uses machine learning algorithms to predict the fare for the selected flight based on historical pricing data and other relevant factors.
- Fare Comparison: Users can compare the predicted fare with current fares from various airlines to determine the best price for their travel needs.
- User Accounts: Users can create accounts to save their search history and receive personalized recommendations based on their travel preferences.
- Mobile-Responsive Design: Our app is designed to work seamlessly on desktop and mobile devices, allowing users to access flight fare predictions from anywhere.

3.7 Deployment

We are deploying the app in **Heroku .** Heroku is a cloud-based platform that allows developers to deploy, manage, and scale their web applications. Here's an overview of how to deploy your Flask web app that predicts flight fares using machine learning to Heroku

1. Create a Heroku Account: Sign up for a free Heroku account at heroku.com and create a new app.
2. Install the Heroku CLI: Install the Heroku Command Line Interface (CLI) on your local machine to interact with Heroku from your terminal. You can download the Heroku CLI from the Heroku Dev Center.
3. Create a Procfile: A Procfile is a file that tells Heroku how to run your app. Create a file named "Procfile" in the root directory of your Flask app, and add the following line:

4. Dashboard

4.1 Deployment

Accuracy of Fare Prediction

We can check the accuracy of our model and we have done it in our notebook, although in practical case, we can just go to any travel websites and check if our prediction is nearly accurate . i have checked with different parameters and features , the model gave almost accurate predictions

Conversion Rate

This web app does not allow any flight booking or any other services as this is only used for the price comparison for a specific date

User Retention

as we are not creating any customer accounts . we are just not considering the retention of users . although this is not a production level application

5. Conclusion

In conclusion, developing a Flask web app that predicts flight fares using machine learning is a valuable service for travelers looking for the best prices for their flights. By implementing key features such as a flight selection form and fare prediction results, you can provide a simple and user-friendly interface for users to access important information