

Model Deployment

In the Milestone, you will see the model deployment

Save The Best Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
pickle.dump(rfr,open('model1.pkl','wb'))
```

Integrate With Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the users where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

Building Html Pages

For this project create two HTML files namely

- home.html
- predict.html
- submit.html

and save them in the templates folder.

Build Python Code

Import the libraries

```

from flask import Flask, render_template, request
import numpy as np
import pickle

```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```

model = pickle.load(open(r"model1.pkl", 'rb'))

```

Render HTML page:

```

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

```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```

@app.route("/predict")
def home1():
    return render_template('predict.html')

@app.route("/pred", methods=['POST', 'GET'])
def predict():
    x = [[int(x) for x in request.form.values()]]
    print(x)

    x = np.array(x)
    print(x.shape)

    print(x)
    pred = model.predict(x)
    print(pred)
    return render_template('submit.html', prediction_text=pred)

```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

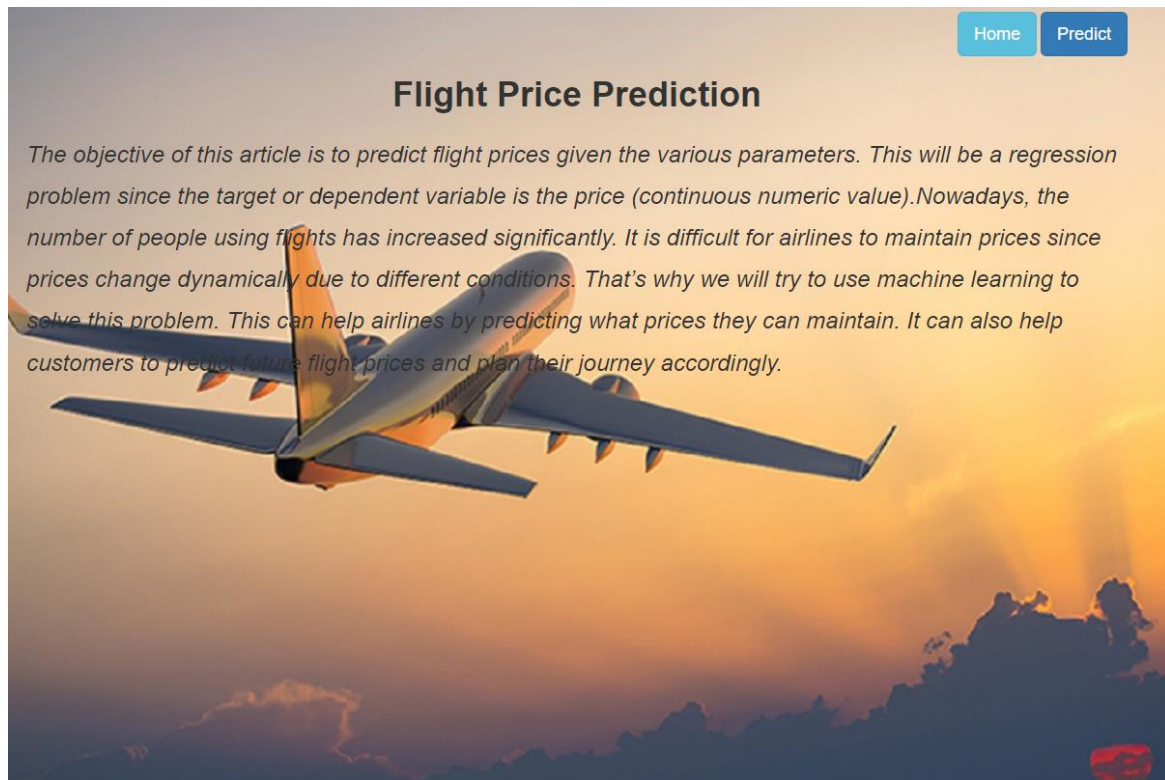
```
if __name__ == "__main__":  
    app.run(debug=False)
```

Run The Web Application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* Serving Flask app "app" (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a p  
Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now, Go to the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



Now, when you click on Predict button you will get redirected to the prediction page.

The image shows the 'Flight Price Prediction' form. It has the same title and 'Home'/'Predict' buttons as the previous page. The form contains several input fields for flight details: airline (a dropdown menu showing 'Air Asia'), source (a dropdown menu showing 'Bangalore'), destination (a dropdown menu showing 'Bangalore'), depdate (a date picker), depmonth (a month picker), depyear (a year picker), depctimehour (a time picker for departure), deptimemins (a time picker for departure minutes), artime (a date picker for arrival), and artimehour (a time picker for arrival). The background is the same sunset sky with an airplane.

Home Predict

Flight Price Prediction

airline

source

destination

depdate

depmonth

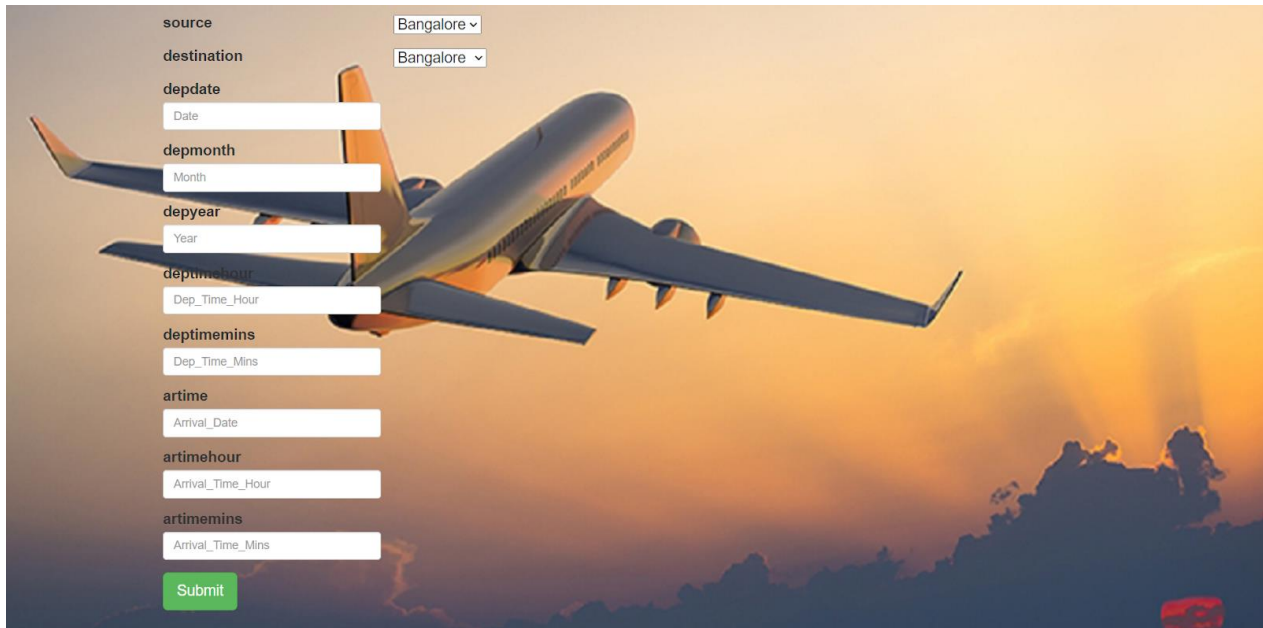
depyear

depctimehour

deptimemins

artime

artimehour



source

destination

depdate

depmonth

depyear

deptimehour

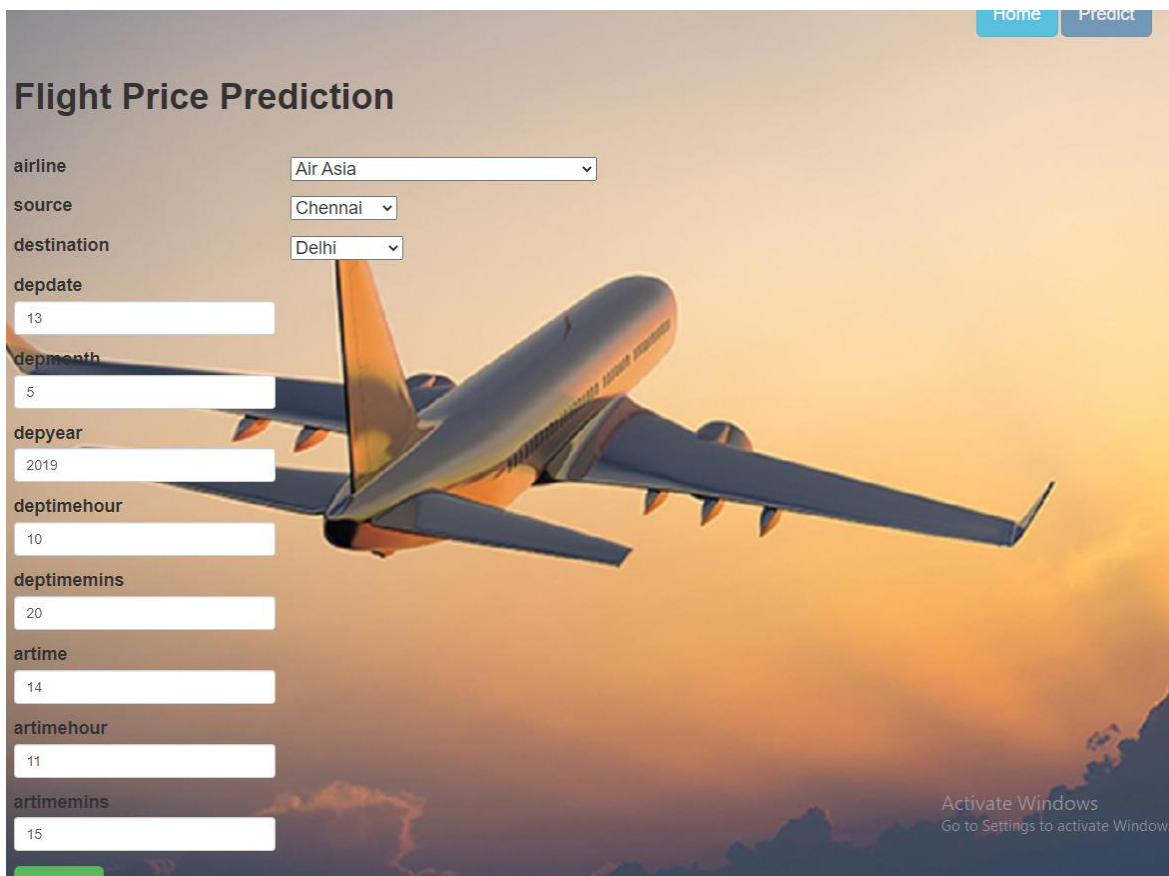
deptimemins

artime

artimehour

artimemins

Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.



[Home](#) [Predict](#)

Flight Price Prediction

airline

source

destination

depdate

depmonth

depyear

deptimehour

deptimemins

artime

artimehour

artimemins

Activate Windows
Go to Settings to activate Windows

Now when you click on submit button from right top corner you will get redirected to submit.html

