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(RCV,open('flight.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 uses where he has to enter the values for predictions. The enter 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 one HTML files namely

- index.html

and save them in the templates folder.

Build Python Code

Import the libraries

```
# importing the necessary dependencies
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os
```

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('flight.pkl','rb'))

app = Flask(__name__)#initializing the app
```

Render HTML page:

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


@app.route('/prediction',methods =['POST'])
```

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:

```
def predict():
    name = request.form['name']
    month = request.form['month']
    dayofmonth = request.form['dayofmonth']
    dayofweek = request.form['dayofweek']
    origin = request.form['origin']
    if(origin == "msp"):
        originl,origin2,origin3,origin4,orgin5 = 0,0,0,0,1
    if(origin == "dtw"):
        originl,origin2,origin3,origin4,orgin5 = 1,0,0,0,0
    if(origin == "jfk"):
         originl,origin2,origin3,origin4,orgin5 = 0,0,1,0,0
    if(origin == "sea"):
         originl,origin2,origin3,origin4,orgin5 = 0,1,0,0,0
    if(origin == "alt"):
         originl,origin2,origin3,origin4,orgin5 = 0,0,0,1,0
```

```
destination = request.form['destination']
if(destination == "msp"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,0,0,1
if(destination == "dtw"):
    destination1,destination2,destination3,destination4,destination5 = 1,0,0,0,0
if(destination == "jfk"):
     destination1,destination2,destination3,destination4,destination5 = 0,0,1,0,0
if(destination == "sea"):
     destination1,destination2,destination3,destination4,destination5 = 0,1,0,0,0
if(destination == "alt"):
     destination1,destination2,destination3,destination4,destination5 = 0,0,0,1,0
dept = request.form['dept']
arrtime = request.form['arrtime']
actdept = request.form['actdept']
dept15=int(dept)-int(actdept)
total = [[name,month,dayofmonth,dayofweek,originl,origin2,origin3,origin4,orgin5,destination1,destination2,destination3,destination4,destination5,i
#print(total)
y_pred = model.predict(total)

print(y_pred)

if(y_pred==[0.]):
    ans="The Flight will be on time"
else:
    ans="The Flight will be delayed"
return render_template("index.html",showcase = ans)
```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Main Function:

```
if __name__ == '__main__':
    app.run(debug = True)
```

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 production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

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

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


127.0.0.1:5000
Prediction of Flight Delay
Enter the Flight Number : 1399
Month : 2
Day of Month : 4
Day of Week : 5
origin JFK
destination SEA
Scheduled Departure Time : 1
Scheduled Arrival Time : 22
Actual Departure Time : 1
SUBMIT


localhost:5000/prediction
Prediction of Flight Delay
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
origin MSP
destination MSP
Scheduled Departure Time :
Scheduled Arrival Time :
Actual Departure Time :
SUBMIT
The Flight will be on time

-