ML deployment using Flask.

1. A simple data created for finding the probability of buying mobile based on AGE, SALARY, GENDER.
2. Used logistic regression for finding the prediction.
3. Using pickle module file dumped into the location.

```
In [16]: from sklearn.metrics import confusion_matrix,accuracy_sco
         ac=accuracy_score(y_test,y_predict)
         ac

Out[16]: 0.8

In [18]: X_test

Out[18]: array([[-0.17996851,  1.49453589, -0.5        ],
                [ 0.4949134 ,  0.86125797,  2.        ],
                [-1.30477168, -1.08079432, -0.5        ],
                [-0.62988978, -0.48973492, -0.5        ],
                [ 1.61971657, -0.78526462, -0.5        ]])

In [19]: import pickle
         pickle.dump(sc,open('scaler.pickle','wb'))
         pickle.dump(classifier,open('mobile_pursc.pkl','wb'))

In [20]: mobile_sc=pickle.load(open('scaler.pickle','rb'))
         model=pickle.load(open('mobile_pursc.pkl','rb'))

In [ ]:
```

4. App.py file created for web interaction with Html file.

```
from flask import Flask, request, jsonify, render_template

import numpy as np
import pickle
app = Flask(__name__)
model=pickle.load(open('mobile_pursc.pkl','rb'))
mobile_sc=pickle.load(open('scaler.pickle','rb'))

@app.route('/')
def home ():
    return render_template('index.html')
@app.route ('/predict',methods=['POST'])

def predict():
        ##For rendering results on HTML GUI
    int_features = [int(x) for x in request.form.values()]
    pre_final_features =[np.array(int_features)]
    final_features = mobile_sc.transform(pre_final_features)
    prediction = model.predict(final_features)
    #('predictio value is ',prediction[0])
    if (prediction[0]== 1):

        output = "True"
    elif(prediction[0] == 0):
        output = "False"
    else:
        output = "Not suree"
    return render_template('index.html', prediction_text='This user will buy mobile: state {}'.format(output)
if __name__ == "__main__":
     app.run(debug=True)
```

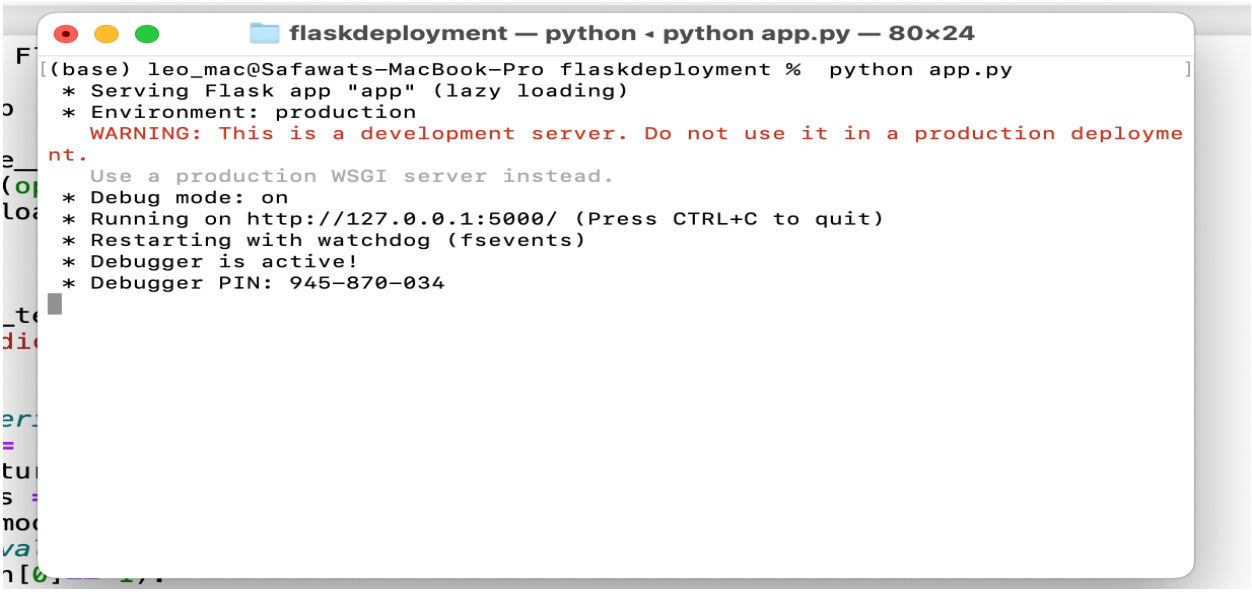
ML deployment using Flask.

5. Simple index.html file created to interact with app.py and create GUI

```
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
<meta charset="UTF-8">
<title>ML API</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

</head>
<body>
<div class="login">
  <h1>MOBILE PURCHASE PREDICTION</h1>

   <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('predict')}}"method="POST">
    <input type="text" name="age" placeholder="age" required="required" />
      <input type="text" name="salary" placeholder="salary" required="required" />
        <input type="text" name="female" placeholder="female" required="required" />


    <button type="submit" class="ban btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

</div>
<body>
</html>
```

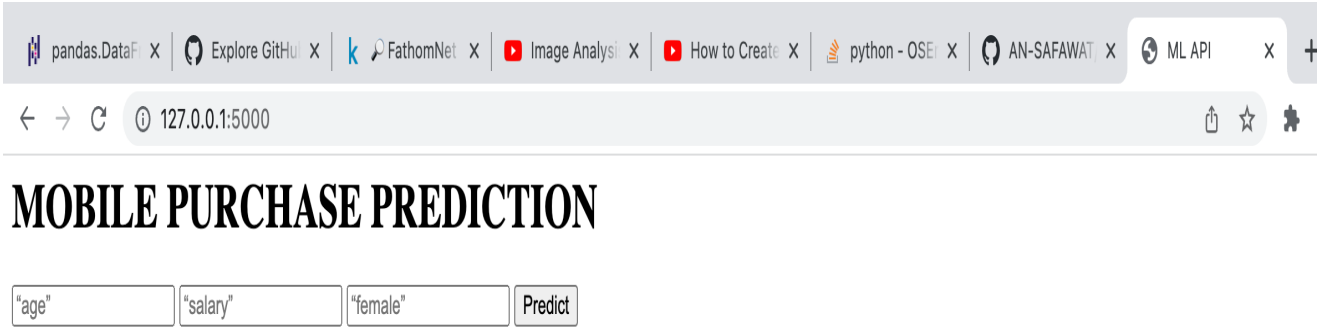
How that works:

1. Created a folder named flaskdeployment where I kept all the files.
2. Using command line interface go to the location and run app.py



ML deployment using Flask.

We can see the port and address for the web application.



Submission date:05/01/2023
Submitted to Data Glacier.