Deployment on Cloud

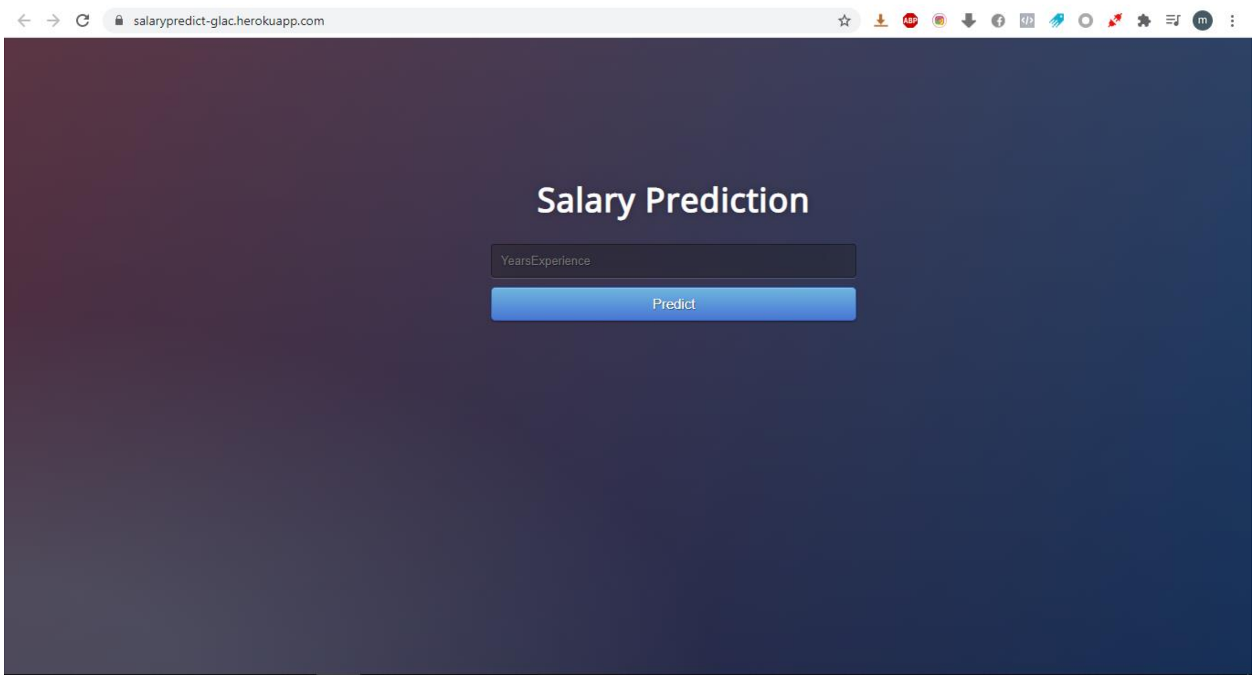
Name: Mohammad Odeh

Batch Code : LISP01

Submit Date: 30 March 2021

Snapshot of the app deployed:

Home page :



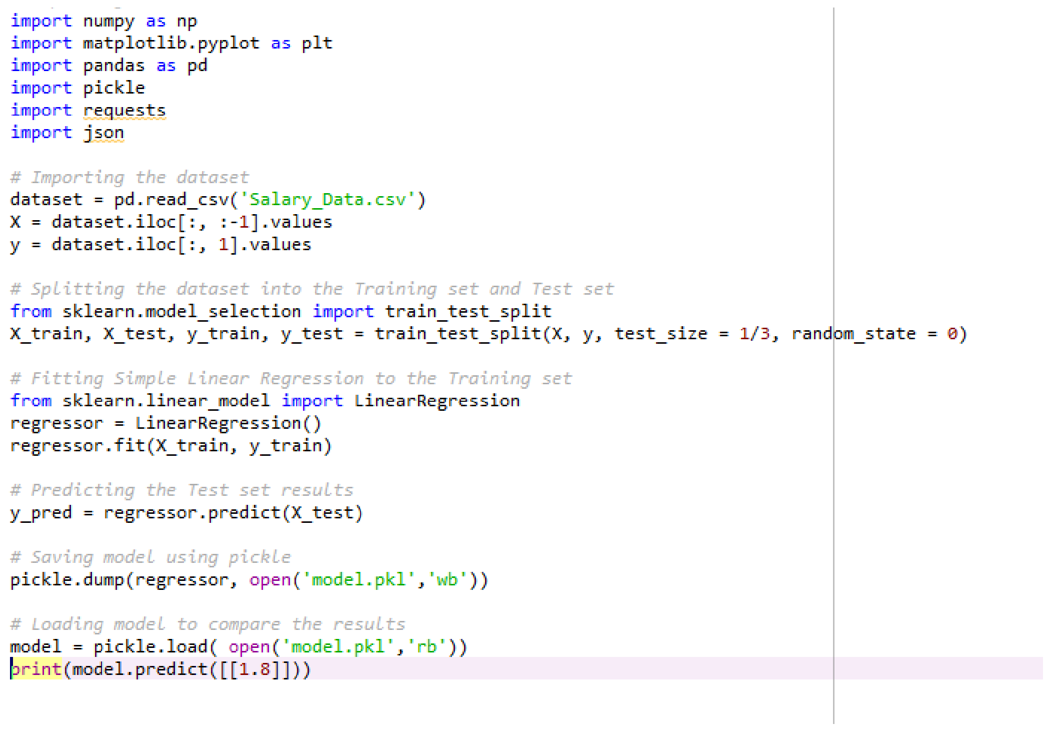
Simple app for salary prediction model

The app was deployed using Heroku service.

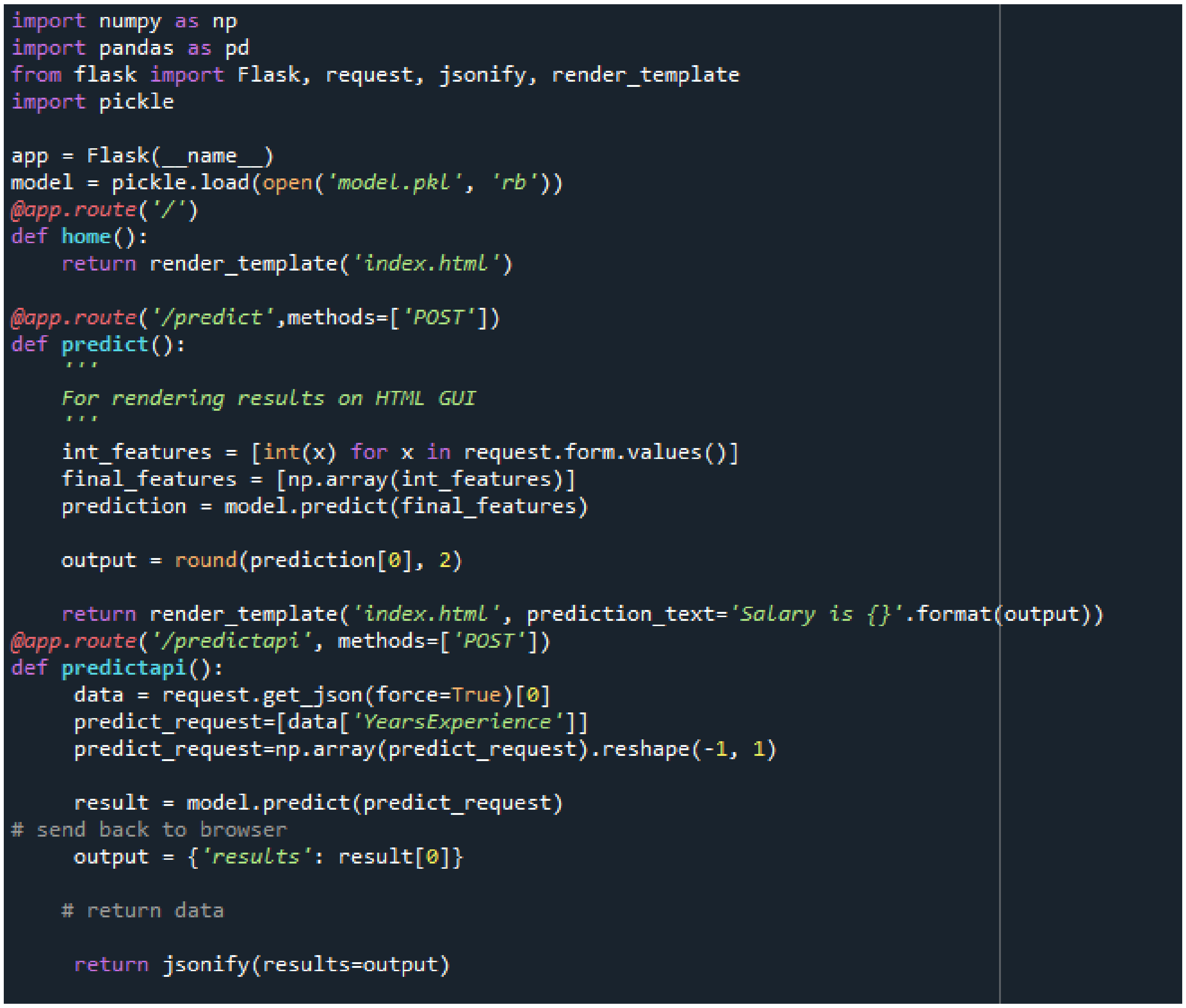
1- **Created api using flask**
2- **Test it the api locally using postman**
3- **Created requirement, runtime, and procfile for herouku**
    - **The requirement and runtime files to let Heroku install the right requirement versions for the app**
    - **Proc file is for Heroku to point it to run app.py file when deployment**

Code :

Main model



```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle
import requests
import json

# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Saving model using pickle
pickle.dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load( open('model.pkl','rb'))
print(model.predict([[1.8]]))
```

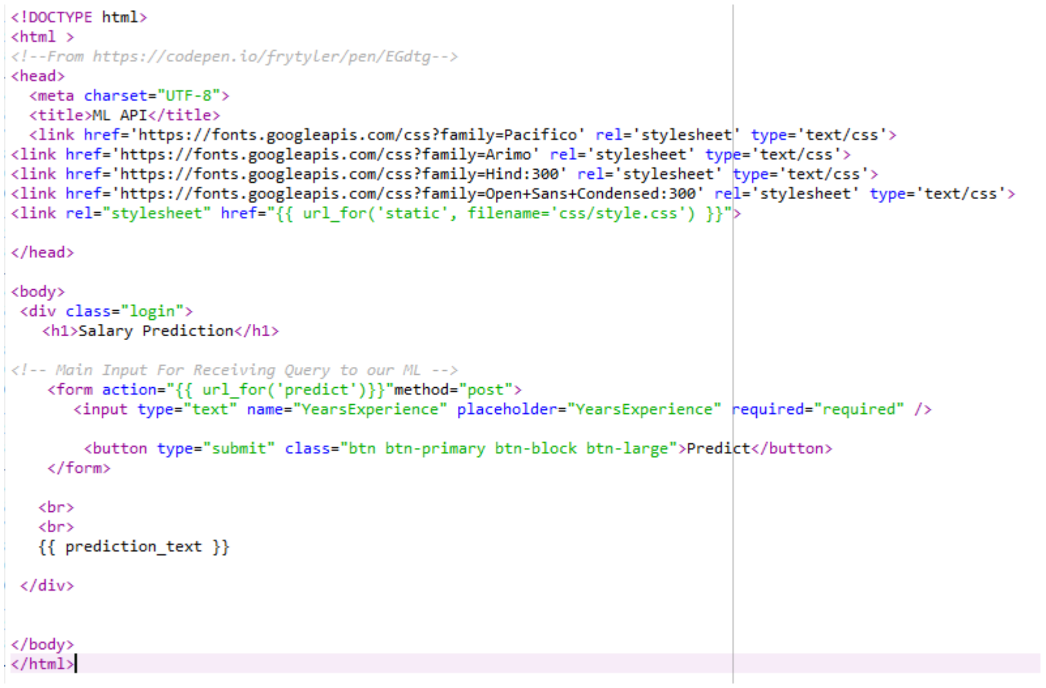


```python
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', '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()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Salary is {}'.format(output))
@app.route('/predictapi', methods=['POST'])
def predictapi():
     data = request.get_json(force=True)[0]
     predict_request=[data['YearsExperience']]
     predict_request=np.array(predict_request).reshape(-1, 1)

     result = model.predict(predict_request)
# send back to browser
     output = {'results': result[0]}

    # return data

     return jsonify(results=output)
```

Flask application:

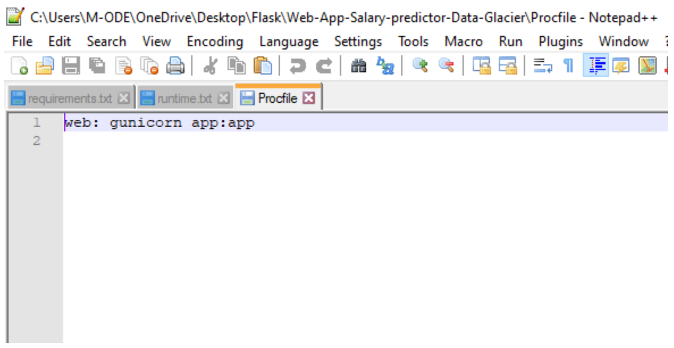
Contain both web app and api

Html:



```
<!DOCTYPE html>
<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'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
   <h1>Salary Prediction</h1>

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

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

   <br>
   <br>
   {{ prediction_text }}

 </div>


</body>
</html>
```

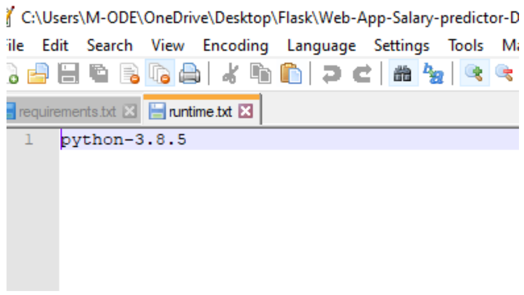
Other files (proc, requirement, runtime)



C:\Users\M-ODE\OneDrive\Desktop\Flask\Web-App-Salary-predictor-Data-Glacier\Procfile - Notepad++

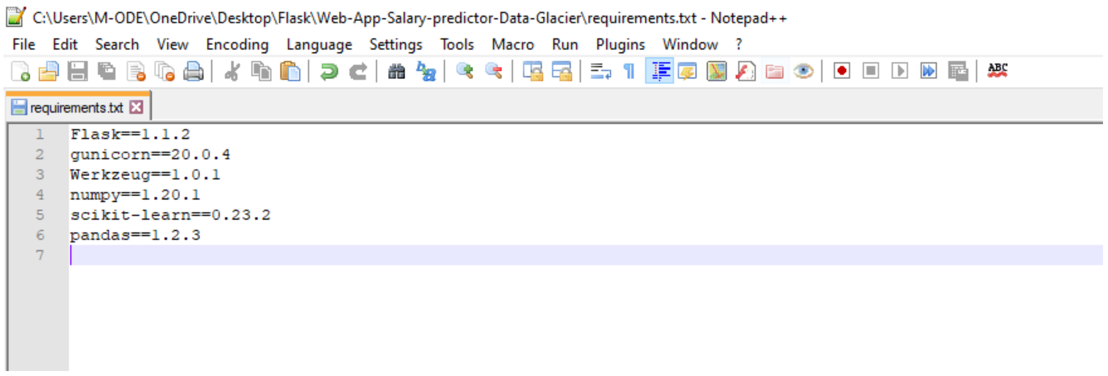
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
requirements.txt
runtime.txt
Procfile

```
web: gunicorn app:app
```



C:\Users\M-ODE\OneDrive\Desktop\Flask\Web-App-Salary-predictor-D

```
python-3.8.5
```



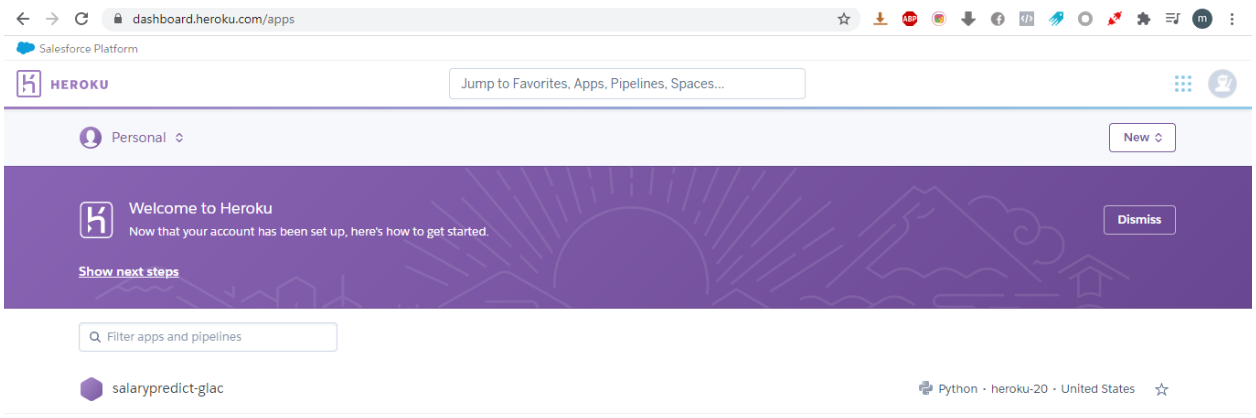
C:\Users\M-ODE\OneDrive\Desktop\Flask\Web-App-Salary-predictor-Data-Glacier\requirements.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
requirements.txt

```
Flask==1.1.2
gunicorn==20.0.4
Werkzeug==1.0.1
numpy==1.20.1
scikit-learn==0.23.2
pandas==1.2.3
```

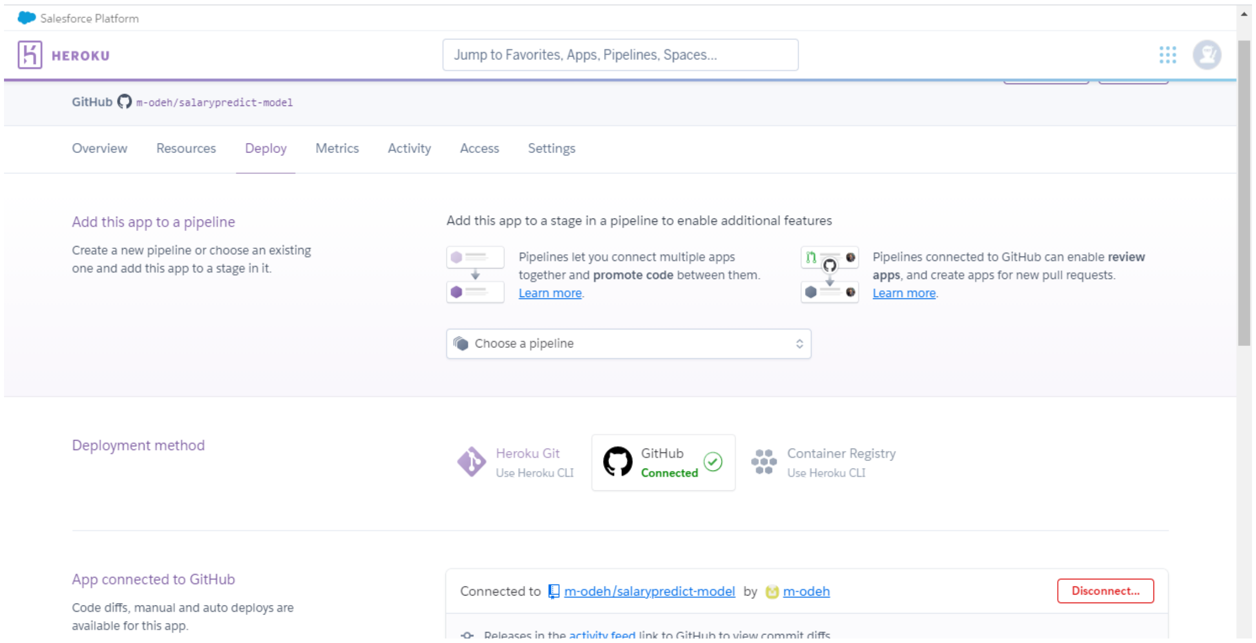
Deployment

For deployment, it can be done by pushing the code from local drive to Heroku using git, but I used another way by connecting repository from GitHub and then deploying directly from the project page on Heroku.


dashboard.heroku.com/apps
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
New
Dismiss
Show next steps
Filter apps and pipelines

Welcome to Heroku
Now that your account has been set up, here's how to get started.

salarypredict-glac — Python · heroku-20 · United States



Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Deployment method

App connected to GitHub
Code diffs, manual and auto deploys are available for this app.

Connected to m-odeh/salarypredict-model by m-odeh

Releases in the activity feed link to GitHub to view commit diffs

After connecting to github, using manual deploy , Heroku will start installing requirements and then deploy the app

https://salarypredict-glac.herokuapp.com/

(API): https://salarypredict-glac.herokuapp.com/predictapi