Cloud and API deployment on Heroku

Name: Disha Lamba

Batch code: LISUM11

Submission date: 3 Aug 2022

Submitted to: Data Glacier

Project Details:

Objective: To predict whether the income of an adult will exceed 50k or not based on various features by developing a supervised ML algorithm.
Dataset used: UCI adult salary dataset
ML algorithm used: Decision Trees

Step 1:
Download and save the UCI adult salary dataset from Kaggle.

age	workclass	fnlwgt	education	educational-nu	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spou	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleane	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spou	Handlers-cleane	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spou	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spou	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spou	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K

The dataset contains 15 columns
Target filed: Income
The income is divided into two classes: <=50K and >50K

Step 2:

Analyze the dataset and develop the ML model using the decision trees algorithm to predict if the income will be greater than 50k or not.

GitHub model link:
https://github.com/dldisha/salary_pred/blob/main/model.py

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X = data.values[:, 0:12]
Y = data.values[:, 12]
```

```
#Splitting the dataset
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 100)

#Using decision tress to train the model using gini index accuracy
dt_clf_gini = DecisionTreeClassifier(criterion = "gini",
                                     random_state = 100,
                                     max_depth = 5,
                                     min_samples_leaf = 5)

#fitting the model
dt_clf_gini.fit(X_train, Y_train)

#x_test prediction
y_pred_gini = dt_clf_gini.predict(X_test)

print("Adult Income salary prediction dataset \n")
print ("Decision Tree using Gini Index Accuracy is: ", accuracy_score(Y_test, y_pred_gini)*100 )
```

```
Adult Income salary prediction dataset

Decision Tree using Gini Index Accuracy is:  83.10983724024977
```

The accuracy achieved: 83.109

Step 3:

Saving the trained model to the disk using the *pickle* library.

```
##Making Pickle file for our model
pickle.dump(model_gini, open("model.pkl", "wb"))
```

Step 4: Deploying the ML model

```
#importing libraries
import os
import numpy as np
import flask
import pickle
from flask import Flask, render_template, request

app=Flask(__name__)

#adult income prediction function to load the model
def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1,12)
    loaded_model = pickle.load(open("model.pkl","rb"))
    result = loaded_model.predict(to_predict)
    return result[0]
```

- Importing the libraries.
- Created the instance of the *Flask()* using *app=Flask(__name__)*.
- Creating function def ValuePredictor to load the model using *pickle.load*, predict the new values, and get the result.

```
@app.route('/')

#Triggering the function index() i.e. adult income form
@app.route('/index')
def home():
    return flask.render_template('index.html')

#Triggering the function result i.e. prediction result
@app.route('/result',methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list=list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        result = ValuePredictor(to_predict_list)

        if int(result)==1:
            prediction='Income more than 50K'
        else:
            prediction='Income less that 50K'

        return render_template("pred.html",prediction=prediction)

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

- *@app.route('/')* is used to tell flask what URL should trigger the function home(), we use *render_template('index.html')* to display the script index.html in the browser which is nothing but a from with feature list.
- After the form, the result/prediction (Income more than or less than 50k) is then passed as an argument to the template engine with the HTML page to be displayed.
- Next, *@app.route(/result)* is used to tell what URL should trigger the function result(), we use *render_template('result.html')* to display the script result.html in the browser which is nothing but showing the prediction of your income.

Step 5:

Checking the main.py file in CMD

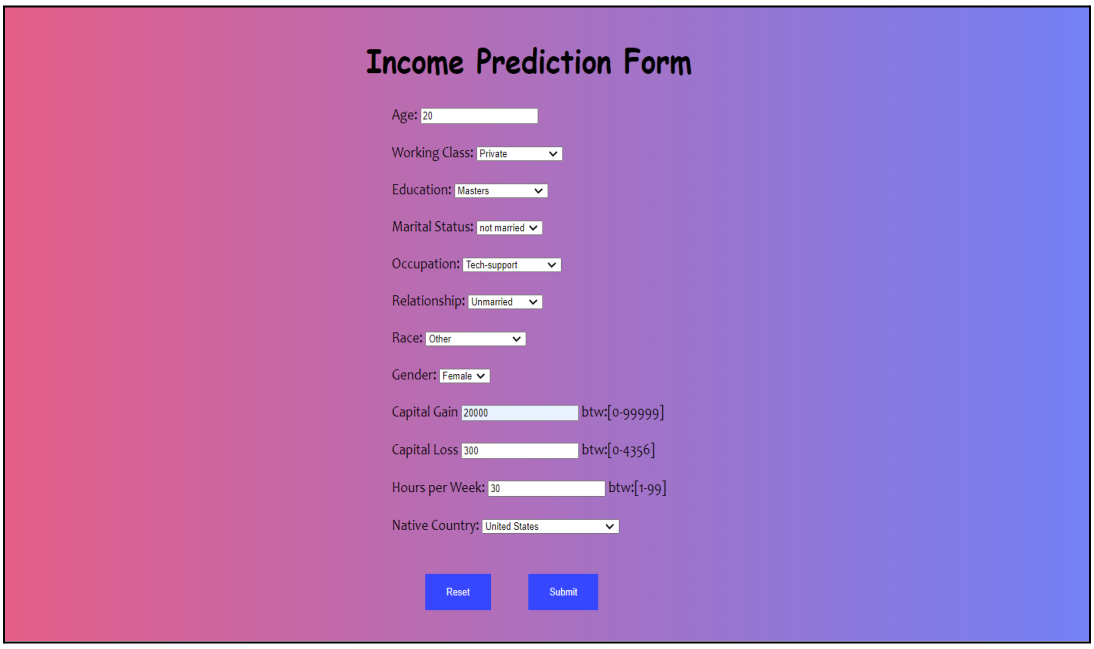
```
C:\Users\Disha Lamba\Desktop\DG\income_pred(master)
λ python main.py
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 793-170-034
```

Open the web application on http://127.0.0.1:5000/

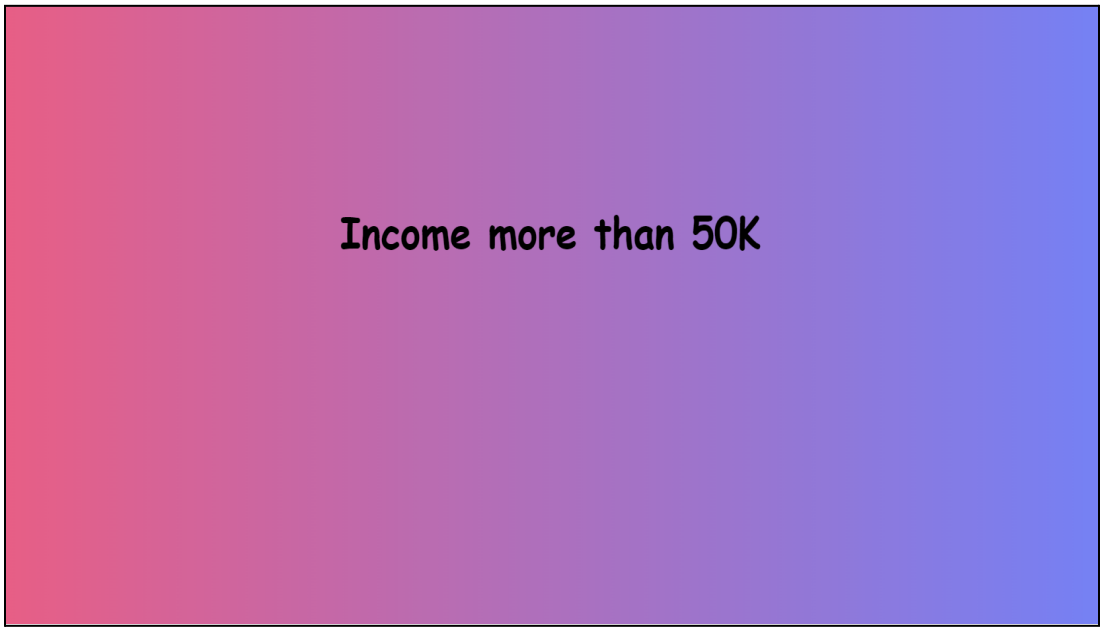
Step 6:

Run the application and it should predict the income after submitting the form and will display the output on the result page.

Index.html (Adult income from)



Result.html (Prediction)



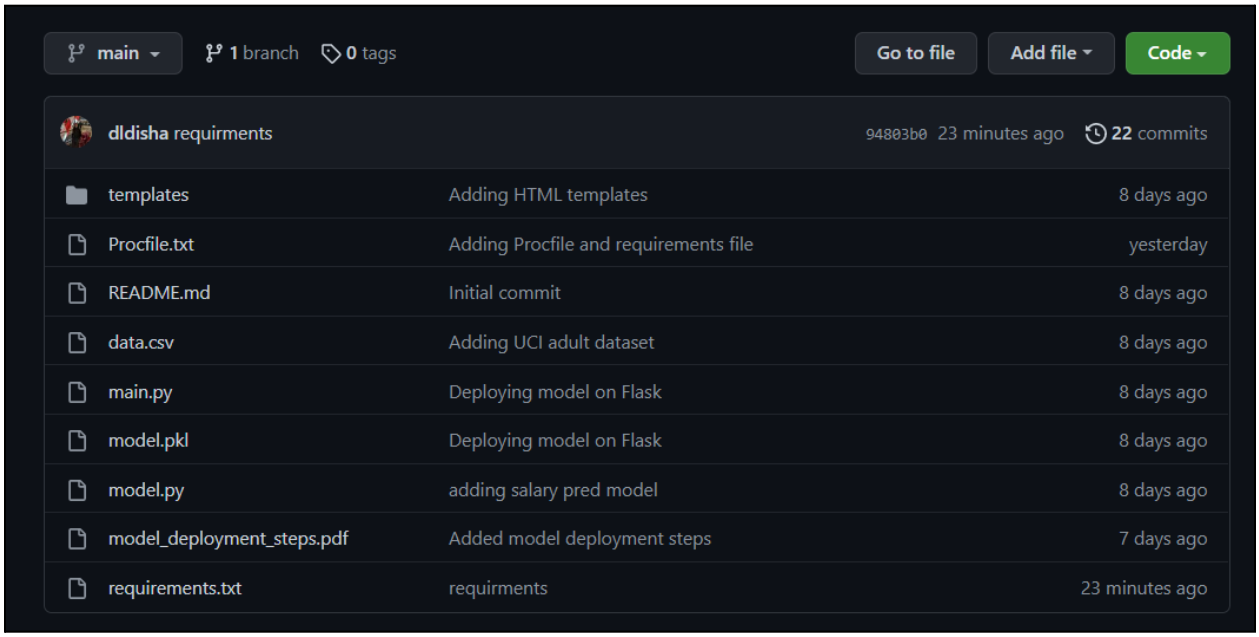
Step 7: Install Gunicorn
Use the command *pip install gunicorn* in CMD.

Step 8: Create Procfile text file
This file specifies the commands that are executed by a Heroku app on startup.
Add *web: gunicorn app:app* in the file.

Step 9: Create a Requirments.txt file
Run the command pip freeze > requirements.txt in CMD for creating a requirements.txt file that will contain all of the dependencies for the flask app.

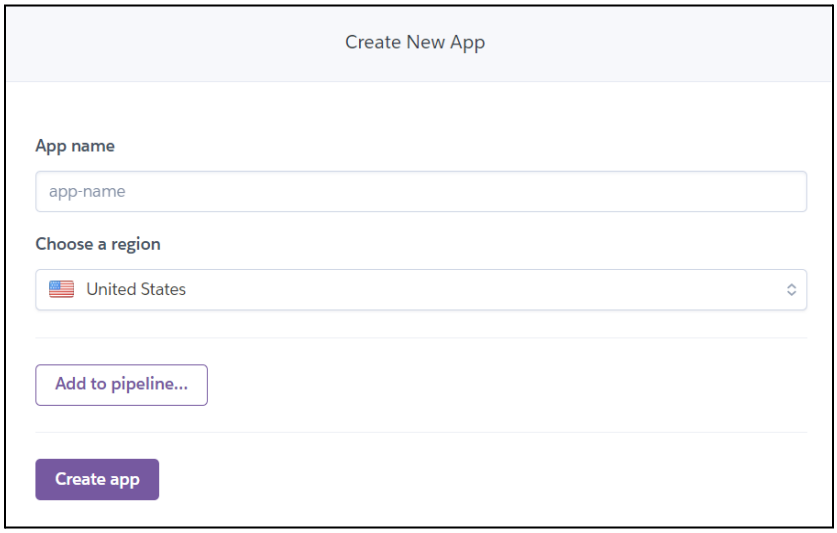
```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy>=1.9.2
scipy>=0.15.1
scikit-learn>=0.18
matplotlib>=1.4.3
pandas>=0.19
```

Step 10: Push your code on GitHub:
Link: https://github.com/dldisha/salary_pred



Step 11: Login into Heroku
Create and login into your Heroku account.

Step 12: Create a Heroku app
On the dashboard, select New -> Create new app:



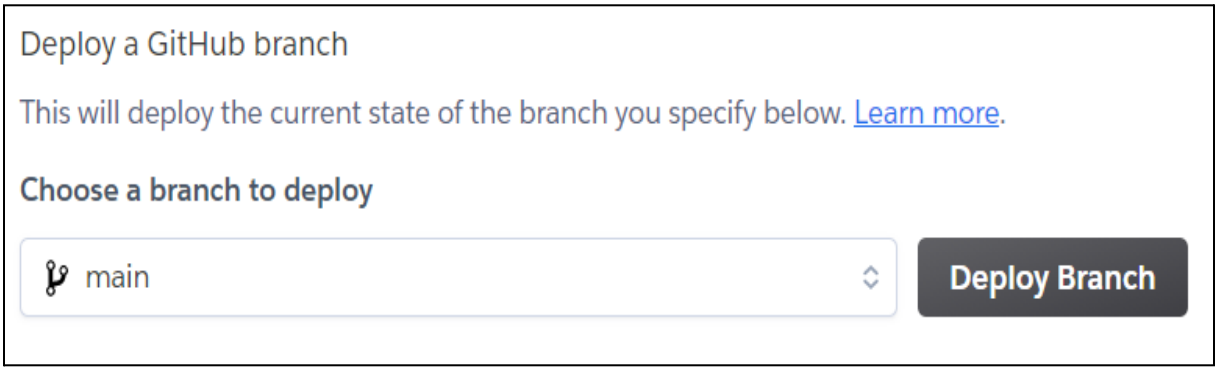
Choose a name for the application and choose a region where you'd like to host it.

Step 13: Connect your GitHub with your Heroku account

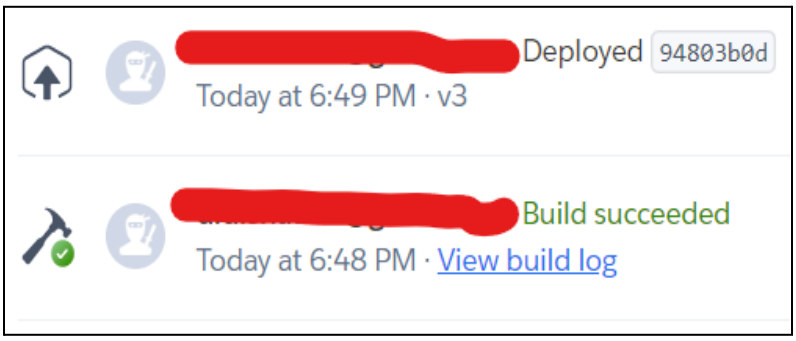
Under your app, select Deploy and connect your GitHub repo for this project with your Heroku app.

Step14: Build your app

After your app is connected with the GitHub repo, we need to deploy our GitHub branch.



Now, wait for the build to complete.



Step 15: Voila, your app is now deployed

Link: https://yourincome.herokuapp.com/