Name: Ameya Damle

Batch Code: LISUM23: 30

Submission Date: 30-July-2023

Submitted to: Data Glacier

Deployment of Flask Web App

Step 1:

Developing a model:

Predict the price of a house based on feature 'area' using Linear Regression Model.

Step 2:

Saving the trained model as pickle file in the root folder.

```
#saving the model in disk
pickle.dump(regressor, open('model.pkl','wb'))
```

```
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[7420]]))
```

```
[10544616.40344053]
```

Step 3:

Deployment of model using flask.

```
# Model Deployment
app = Flask(__name__)
model = pickle.load(open('model.pkl','rb'))
```

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

```
@app.route('/',methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([[np.array(data['exp'])]])
    output = prediction[0]
    return jsonify(output)
```

```
if __name__ == '__main__':
    app.run(port=5000)
```

```
* Serving Flask app '__main__'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production W
* Running on http://127.0.0.1:5000
```

The script creates a Flask app and loads a machine learning model from a pickled file (model.pkl). The app has two routes defined: one for the home page ("/") that renders the "index.html" template, and another for handling POST requests sent to the root URL ("/") for making predictions. When the user submits the input data using the form on the home page, a POST request is sent to the server, and the predict() function is executed to make predictions using the model. The result of the prediction is then returned as a JSON response to the client.
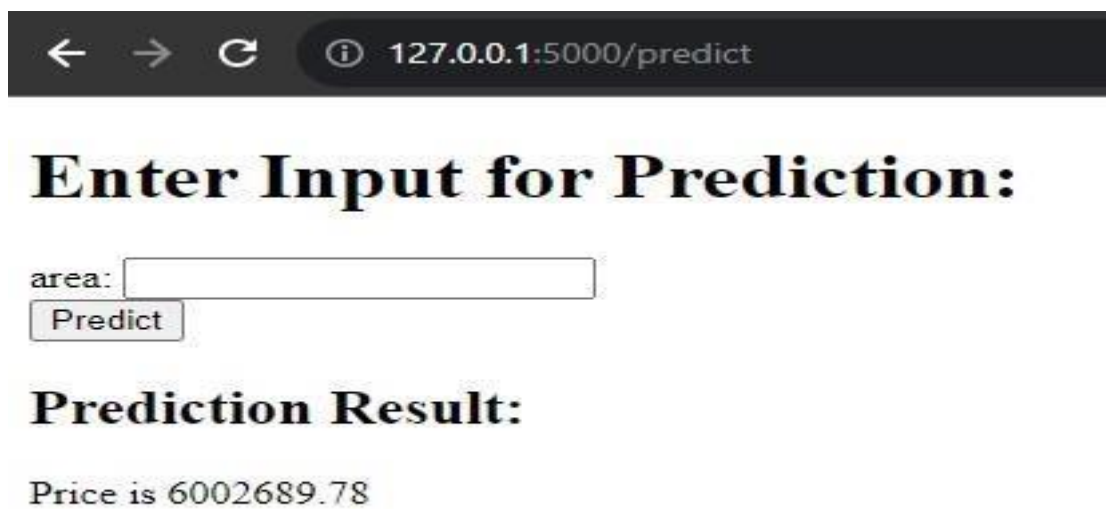
Step 4:

Checking the app.py file in cmd.

Step 5:

Launching the webapp by typing address "127.0.0.1:5000" in any browser.

Deployment of web application in Azure Cloud:

Step 6:
Creating a web app in Azure. While creating the Web application make sure you enable GitHub Actions in Deployment step and register your GitHub account as well.

Create Web App ...

Resource Group * ⓘ (New) HousePrice ⌄
Create new

Instance Details

Need a database? Try the new Web + Database experience. ⤤

Name * HousePriceAPI ✓
.azurewebsites.net

Publish * ⦿ Code ◯ Docker Container ◯ Static Web App

Runtime stack * Python 3.8 ⌄

Operating System * ⦿ Linux ◯ Windows

Region * East US ⌄
ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

Create Web App ...

Basics **Deployment** Networking Monitoring Tags Review + create

Enable GitHub Actions to continuously deploy your app. GitHub Actions is an automation framework that can build, test, and deploy your app whenever a new commit is made in your repository. If your code is in GitHub, choose your repository here and we will add a workflow file to automatically deploy your app to App Service. If your code is not in GitHub, go to the Deployment Center once the web app is created to set up your deployment. Learn more ⤤

GitHub Actions settings

Continuous deployment ◯ Disable ⦿ Enable

GitHub Actions details

Select your GitHub details, so Azure Web Apps can access your repository. You must have write access to your chosen repository to deploy with GitHub Actions.

GitHub account daameya
Change account ⓘ

Organization * daameya ⌄

Repository * House-Price-Prediction ⌄

Branch * main ⌄

Review + create   < Previous   Next : Networking >

Step 7:
Review and create the web application and it automatically activate a continuous pipeline with the repository in your GitHub profile.

Step 8:

Use the web address now available in the Actions section of the repository. After the deployment is successful you will be able to see a workflow folder in your repository which is created automatically.