Week 4: Deployment on Flask

Name: Deployment on Flask
Report date: 28-FEB-2024
Internship Batch: LISUM30
Version:
Data intake by: Noura Alsahli
Data intake reviewer : Data Glacier
Data storage location: https://github.com/Fnoura/week4

Tabular data details:

Total number of observations	5000
Total number of files	1
Total number of features	7
Base format of the file	csv
Size of the data	709.2KB

1- assessing the dataset

[1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
```

[2]:
```
house_data=pd.read_csv('USA_Housing.csv')
```

[3]:
```
house_data.head()
```

[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

[4]:
```
house_data.rename(columns = {'Avg. Area Income':'Area_Income'}, inplace = True)
house_data.rename(columns = {'Avg. Area House Age':'Area_House_Age'}, inplace = True)
house_data.rename(columns = {'Avg. Area Number of Rooms':'Area_Number_Rooms'}, inplace = True)
house_data.rename(columns = {'Avg. Area Number of Bedrooms':'Area_Number_Bedrooms'}, inplace = True)
house_data.rename(columns = {'Area Population':'Area_Populations'}, inplace = True)
```

[5]:
```
house_data.head()
```

[5]:

	Area_Income	Area_House_Age	Area_Number_Rooms	Area_Number_Bedrooms	Area_Populations	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

[6]:
```
house_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Area_Income           5000 non-null   float64
 1   Area_House_Age        5000 non-null   float64
 2   Area_Number_Rooms     5000 non-null   float64
 3   Area_Number_Bedrooms  5000 non-null   float64
 4   Area_Populations      5000 non-null   float64
 5   Price                 5000 non-null   float64
 6   Address               5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

[8]:
```
house_data.describe()
```

[8]:

	Area_Income	Area_House_Age	Area_Number_Rooms	Area_Number_Bedrooms	Area_Populations	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

[9]:
```
house_data.isnull().sum()
```

[9]:
```
Area_Income             0
Area_House_Age          0
Area_Number_Rooms       0
Area_Number_Bedrooms    0
Area_Populations        0
Price                   0
Address                 0
dtype: int64
```

dtype: int64

[10]:
```
house_data=house_data.drop(['Address'], axis=1)
house_data.head()
```

[10]:

	Area_Income	Area_House_Age	Area_Number_Rooms	Area_Number_Bedrooms	Area_Populations	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05

[11]:
```
house_data.shape
```

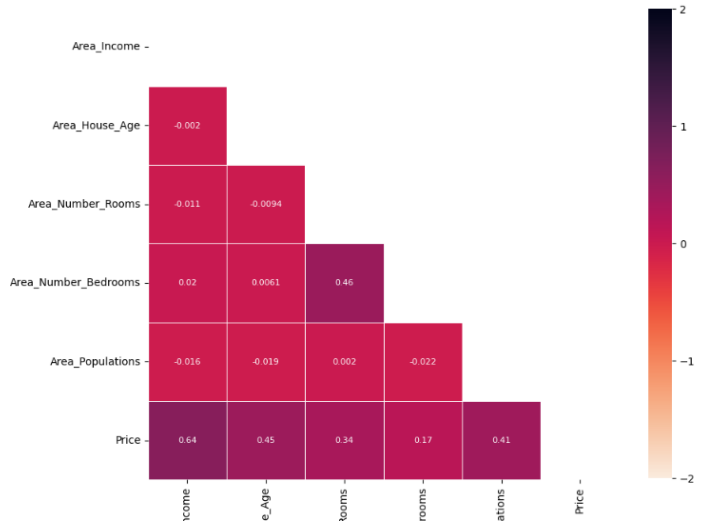
[11]:
```
(5000, 6)
```

[12]:
```
house_data.tail()
```

[12]:

	Area_Income	Area_House_Age	Area_Number_Rooms	Area_Number_Bedrooms	Area_Populations	Price
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

[13]:
```
corr=house_data.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]= True
f, ax = plt.subplots(figsize=(10, 10))
heatmap = sns.heatmap(corr, mask = mask,
                      square = True,
                      linewidths = .5,
                      cmap = "rocket_r",
                      cbar_kws = {'shrink': .8,
                                  "ticks" : [-2, -1, 0, 1, 2]},
                      vmin = -2,
                      vmax = 2,
                      annot = True,
                      annot_kws = {"size":8})
# Add the column names as labels
ax.set_yticklabels(corr.columns)
ax.set_xticklabels(corr.columns)
sns.set_style({'xtick.bottom': True}, {'ytick.left': True});
```



2- constructing the model:

Using ML to estimate the house prices once the data preprocessed.
The model was generated using RFR and linear regression where it was Found that linear regression approach preformed the best.

```
[14]: from sklearn.model_selection import train_test_split

x = house_data.drop(['Price'],axis=1)
y = house_data['Price']

X_train, X_test, y_train, y_test = train_test_split(
    x, y, train_size=0.70,test_size=0.30, random_state=0)
print(X_train.shape, X_test.shape)
```

```
(3500, 5) (1500, 5)
```

```
[15]: from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_percentage_error
from sklearn import metrics

model_RFR=RandomForestRegressor(n_estimators = 1000, random_state = 42)
model_RFR.fit(X_train,y_train)
y_pred=model_RFR.predict(X_test)

print('R2 Value:',metrics.r2_score(y_test, model_RFR.predict(X_test)))
print('Accuracy',100- (np.mean(np.abs((y_test - y_pred ) / y_test)) * 100))
pd.Series(model_RFR.feature_importances_, index=x.columns).sort_values(ascending=False)
```

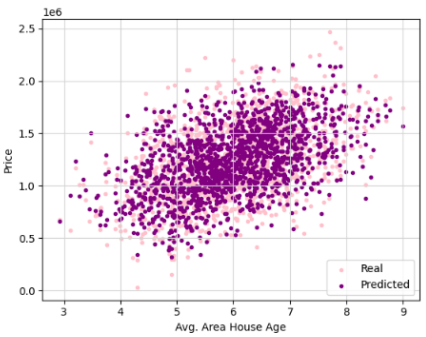
```
R2 Value: 0.888424780124613
Accuracy 90.40888727311193
```

```
[15]: Area_Income             0.428232
Area_House_Age          0.237280
Area_Populations        0.188185
Area_Number_Rooms       0.128525
Area_Number_Bedrooms    0.017778
dtype: float64
```

```
[18]: import matplotlib.pyplot as plt


x_axis = X_test.Area_House_Age


plt.scatter(x_axis, y_test, color = 'pink', marker = '.', label = 'Real')
plt.scatter(x_axis, y_pred, color = 'purple',marker = '.',label = 'Predicted')
plt.xlabel('Avg. Area House Age')
plt.ylabel('Price')
plt.grid(color ='#D3D3D3')
plt.legend(loc ='lower right')
plt.show()
```




```
[19]:
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error


mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test.values,y_pred)

print('Mean Absolute Error:', round(mae, 2))
print('Mean Squared Error:', round(mse, 2))
print('R-squared scores:', round(r2, 2))
```

```
Mean Absolute Error: 95982.06
Mean Squared Error: 14431051360.91
R-squared scores: 0.89
```

```
[20]:
from sklearn.linear_model import LinearRegression
model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

print("Model Coefficients:", model.coef_)
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Coefficient of Determination:", r2_score(y_test, y_pred))

pd.Series(model.coef_, index=x.columns).sort_values(ascending=False)
```

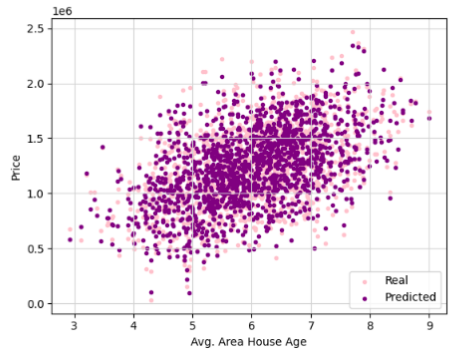
```
Model Coefficients: [2.16187374e+01 1.66145180e+05 1.21010577e+05 1.76003780e+03
 1.51647974e+01]
Mean Absolute Error: 81563.14733994487
Coefficient of Determination: 0.9200757649412041
```

```
[20]: Area_House_Age          166145.179949
Area_Number_Rooms       121010.576873
Area_Number_Bedrooms      1760.037796
Area_Income                 21.618737
Area_Populations            15.164797
dtype: float64
```

[21]:
```
import matplotlib.pyplot as plt


x_axis = X_test.Area_House_Age


plt.scatter(x_axis, y_test, color = 'pink', marker = '.', label = 'Real')
plt.scatter(x_axis, y_pred, color = 'purple',marker = '.',label = 'Predicted')
plt.xlabel('Avg. Area House Age')
plt.ylabel('Price')
plt.grid(color ='#D3D3D3')
plt.legend(loc ='lower right')
plt.show()
```



3- saving the model and installing flask

[22]:
```
import warnings
warnings.filterwarnings('ignore')

predict = model.predict(X_test)
result = X_test
result['Price'] = y_test
result['Predic_Price'] = predict.tolist()
result.head()
```

[22]:

	Area_Income	Area_House_Age	Area_Number_Rooms	Area_Number_Bedrooms	Area_Populations	Price	Predic_Price
398	61200.726175	5.299694	6.234615	4.23	42789.692217	894251.068636	969608.346806
3833	63380.814670	5.344664	6.001574	2.45	40217.333577	932979.360621	953868.155486
4836	71208.269301	5.300326	6.077989	4.01	25696.361741	920747.911288	907506.328361
4572	50343.763518	6.027468	5.160240	4.35	27445.876739	691854.921027	493325.260323
636	54535.453719	5.278065	6.871038	4.41	30852.207006	732733.236293	718221.210115

[23]:
```
import pickle
pickle.dump(model, open('model.pkl', 'wb'))
```

[24]:
```
!pip install flask --quiet
!pip install flask-ngrok --quiet
```

[26]:
```
pip install Flask-PyNgrok
```

```
Collecting Flask-PyNgrokNote: you may need to restart the kernel to use updated packages.

  Downloading Flask-PyNgrok-1.0.3.tar.gz (2.3 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: Flask in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask-PyNgrok) (3.0.2)
Collecting pyngrok (from Flask-PyNgrok)
  Downloading pyngrok-7.1.3-py3-none-any.whl.metadata (7.6 kB)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask->Flask-PyNgrok)
(3.0.1)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask->Flask-PyNgrok) (3.
1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask->Flask-PyNgro
k) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask->Flask-PyNgrok) (8.
1.6)
Requirement already satisfied: blinker>=1.6.2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from Flask->Flask-PyNgrok)
```

4- deployment on flask

```
Installing collected packages: pyngrok, Flask-PyNgrok
Successfully installed Flask-PyNgrok-1.0.3 pyngrok-7.1.3
```

```
[*]: # import Flask from flask module
from flask import Flask

# import run_with_ngrok from flask_ngrok to run the app using ngrok
from flask_ngrok import run_with_ngrok
from flask import Flask, request, render_template
app = Flask(__name__) #app name
run_with_ngrok(app)

model = pickle.load(open('model.pkl','rb'))

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

@app.route('/', methods = ['POST'])
def predict():
    int_features = [int(x) for x in request.form.values()]
    features = [np.array(int_features)]
    prediction = model.predict(features)

    output = round(prediction[0], 2)

    if output < 0:
        return render_template('index.html', prediction_text = " Values entered not reasonable")
    elif output >= 0:
        return render_template('index.html', prediction_text = 'Predicted Price of the house is: ${}'.format(output))

#Run app
if __name__ == "__main__":
    app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

```
[31]: mkdir "templates" -Force
```

```
A subdirectory or file templates already exists.
Error occurred while processing: templates.
```

5- HTML code

index.html

C: > Users > Admin > Desktop > index.html > html > head > style > h1

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>House Price Predictor</title>
    <style>
        /* Reset default browser styles */
        body, h1, p {
            margin: 0;
            padding: 0;
        }

        /* Overall container styling */
        .container {
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
            font-family: Arial, sans-serif;
            background-color: #f9f9f9;
            border: 1px solid #ddd;
            border-radius: 5px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }

        /* Section headings */
        h1 {
            font-size: 24px;
            margin-bottom: 20px;
            color: #333;
        }

        /* Input fields */
        input[type="number"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 3px;
            font-size: 14px;
        }

        /* Button styling */
        button {
            background-color: #007bff;
            color: #fff;
            border: none;
            padding: 10px 20px;
            border-radius: 3px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        button:hover {
            background-color: #0056b3;
        }

        /* Result text */
        .result {
            font-size: 18px;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>House Price Predictor</h1>
        <form>
            <label for="income">Average Area Income:</label>
            <input type="number" id="income" placeholder="Enter income">

            <!-- Add similar input fields for other parameters (house age, rooms, bedrooms, population) -->

            <button type="submit">Predict Price</button>
        </form>
        <p class="result">{{ prediction_text }}</p>
    </div>
</body>
</html>
```

6- the webapp


127.0.0.1:5000
House Price Predictor
Average Area Income:
Enter income
Predict Price