Week 4: Deploy the ML Model on Flask

Name: Deploy the ML model on Flask
Report date: 05/16/2024
Internship Batch: LISUM33
Version: 1.0
Data intake by: Manhui Zhu
Data Intake reviewer: Data Glacier
Data Storage location:
https://github.com/Manhui-z/Data-Glacier-Internship/tree/main/Week%204%20Flask

Dataset details:

Name of data	Option_data.csv
Total number of observations	50000
Total number of features	6
Base format of the file	.csv
Size of the data	273.6 + KB

1. Build the ML Model and Save

1.1 Import the Packages

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import r2_score, accuracy_score

from sklearn.ensemble import RandomForestRegressor
```

1] ✓ 21.2s Python

1.2 Data Preprocessing

Data Preprocessing

```
option = pd.read_csv('option_data.csv')
option.head()
```

✓ 0.1s Python

	Unnamed: 0	Value	S	K	tau	r	BS
0	1	348.500	1394.46	1050	0.128767	0.0116	Under
1	2	149.375	1432.25	1400	0.679452	0.0113	Under
2	3	294.500	1478.90	1225	0.443836	0.0112	Under
3	4	3.375	1369.89	1500	0.117808	0.0119	Over
4	5	84.000	1366.42	1350	0.298630	0.0119	Under

```
option.info()
```

[4] ✓ 0.3s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  5000 non-null   int64
 1   Value       5000 non-null   float64
 2   S           5000 non-null   float64
 3   K           5000 non-null   int64
 4   tau         5000 non-null   float64
 5   r           5000 non-null   float64
 6   BS          5000 non-null   object
dtypes: float64(4), int64(2), object(1)
memory usage: 273.6+ KB
```

```
option.isnull().sum()
```

[3] Python

```
Unnamed: 0    0
Value         0
S             0
K             0
tau           0
r             0
BS            0
dtype: int64
```

```
option = option.drop(columns = ['Unnamed: 0'])
option.shape
```

[4] Python

```
(5000, 6)
```

```python
X = option.drop(columns = ['Value', 'BS'])
y = option['Value']

re_X_train, re_X_test, re_y_train, re_y_test = train_test_split(X, y,
                                                test_size=0.2, random_state=42)
```

Python

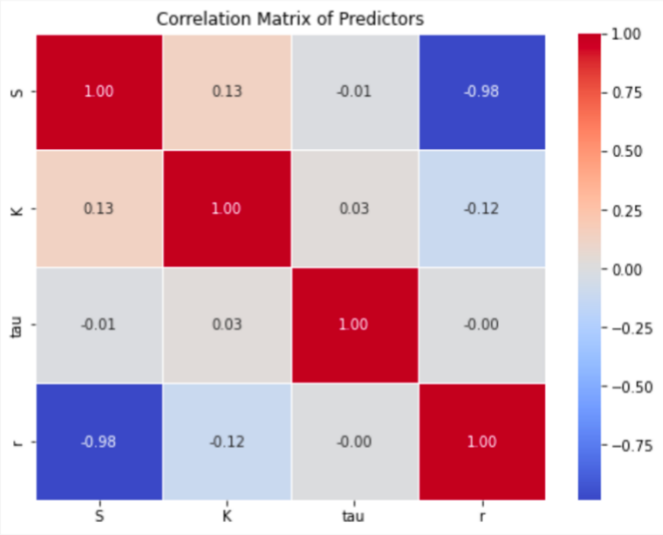
```python
re_X_train.shape
```

Python

```
(4000, 4)
```

```python
correlation_matrix = re_X_train.corr()

# Visualize the correlation matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Predictors')
plt.show()
```

Python



1.3 Build the Model

Build the Model

```python
random_forest_regressor = RandomForestRegressor(n_estimators=50, random_state=42)

# Fit the Random Forest regressor to the training data
random_forest_regressor.fit(re_X_train, re_y_train)

# Predict on the testing set
predictions = random_forest_regressor.predict(re_X_test)

# result table
result = re_X_test
result['Value'] = re_y_test
result['Predicted_Value'] = predictions.tolist()
result.head()
```

Python

	S	K	tau	r	Value	Predicted_Value
1501	1401.44	1625	0.090411	0.0116	1.000	1.4550
2586	1527.46	1350	0.232877	0.0106	224.000	219.4900
2653	1329.78	1300	0.676712	0.0123	147.625	154.8975
1055	1424.24	1375	0.358904	0.0114	116.000	120.9100
705	1341.93	1200	0.295890	0.0122	174.375	169.4900

```python
# Calculate evaluation metrics
rf_mse = mean_squared_error(re_y_test, predictions)
print("Mean Squared Error: ", rf_mse)

rf_r2 = r2_score(re_y_test, predictions)
print("R-squared (Random Forest):", rf_r2)
```

Python

```
Mean Squared Error:  55.937986762109375
R-squared (Random Forest): 0.9964970515427147
```

1.4 Save the Model

Save the model

[11]
```python
import pickle
```

Python

[12]
```python
pickle.dump(random_forest_regressor, open('model.pkl', 'wb'))
```

Python

2. Deploy the ML Model on Flask (Web App)

2.1 app.py

```
import os
os.chdir('/Users/zhumanhui/Desktop/Data Glacier/Week 4 Flask/')
print("Current Working Directory:", os.getcwd())

import pandas as pd
import numpy as np
from flask import Flask, request, render_template, url_for
import pickle

app = Flask(__name__)  # app name

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

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

# set a post method to yield predictions on page
@app.route('/predict', methods = ['POST'])
def predict():

    # obtain value of all predcitors and place them in array
    int_features = [float(x) for x in request.form.values()]
    # conbine them all into a final numpy array
    final_features = [np.array(int_features)]
    # predict the optin values by given input predictors
    prediction = model.predict(final_features)[0]

    # if the ouput is negative, the predcitor values entered are unreasonable
    if prediction < 0:
        return render_template('index.html',
            predcition_text = 'Predicted option value is negative, values entered is unrea
    # if  the output is greater than 0, return prediction
    else:
        return render_template('index.html',
            prediction_text = 'Predicted option value is: $ {}'.format(prediction))

# run app
if __name__ == '__main__':
    app.run(port = 5000, debug = True)
```

Output on my Terminal:

```
zhumanhui@shus-MacBook-Pro ~ % /usr/local/bin/python3 "/Users/zhumanhui/Desktop/Data Glacier/Week 4 Flask/app.py"
Current Working Directory: /Users/zhumanhui/Desktop/Data Glacier/Week 4 Flask
 * Serving Flask app 'app' (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
Current Working Directory: /Users/zhumanhui/Desktop/Data Glacier/Week 4 Flask
 * Debugger is active!
 * Debugger PIN: 841-984-386
127.0.0.1 - - [28/May/2024 01:16:12] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2024 01:16:12] "GET /favicon.ico HTTP/1.1" 404 -
```

2.2 Build the html file

```html
<html>

    <head>
        <meta charset="UTF-8">
        <style>
            /*This section involves the overall style of main tags*/

            * {
                font-family: Lucida Handwriting;
            }

            body {
                background-color: rgb(12, 138, 211);
                background-size: cover;
            }

            form {
                text-align: center;
            }

            h1 {
                color: white;
                text-align: center;
                font-family: Lucida Handwriting;
                font-size: 500%;
            }
```

```css
button {
    font-weight: bold;
    background-color: rgb(12, 138, 221);
    padding: 8px 16px;
    display: inline-block;
    text-decoration: none;
    border-radius: 3px;
    color: black;
    border-color: black;
    font-family: Monaco;
    border-style: solid;
}

input {
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box
}

label {
    color: white;
}

/*Margin, layout and design of paragraphs and structures*/

.para {
    text-align: center;
}

.result {
    font-weight: bold;
    background-color: rgb(12, 138, 221);
    padding: 8px 16px;
    display: inline-block;
    text-decoration: none;
    border-radius: 3px;
    color: black;
    border-color: black;
    font-family: Monaco;
    border-style: solid;
}

.pred {
    text-align: center;
}

.intro {
    font-size: 20px;
}
```

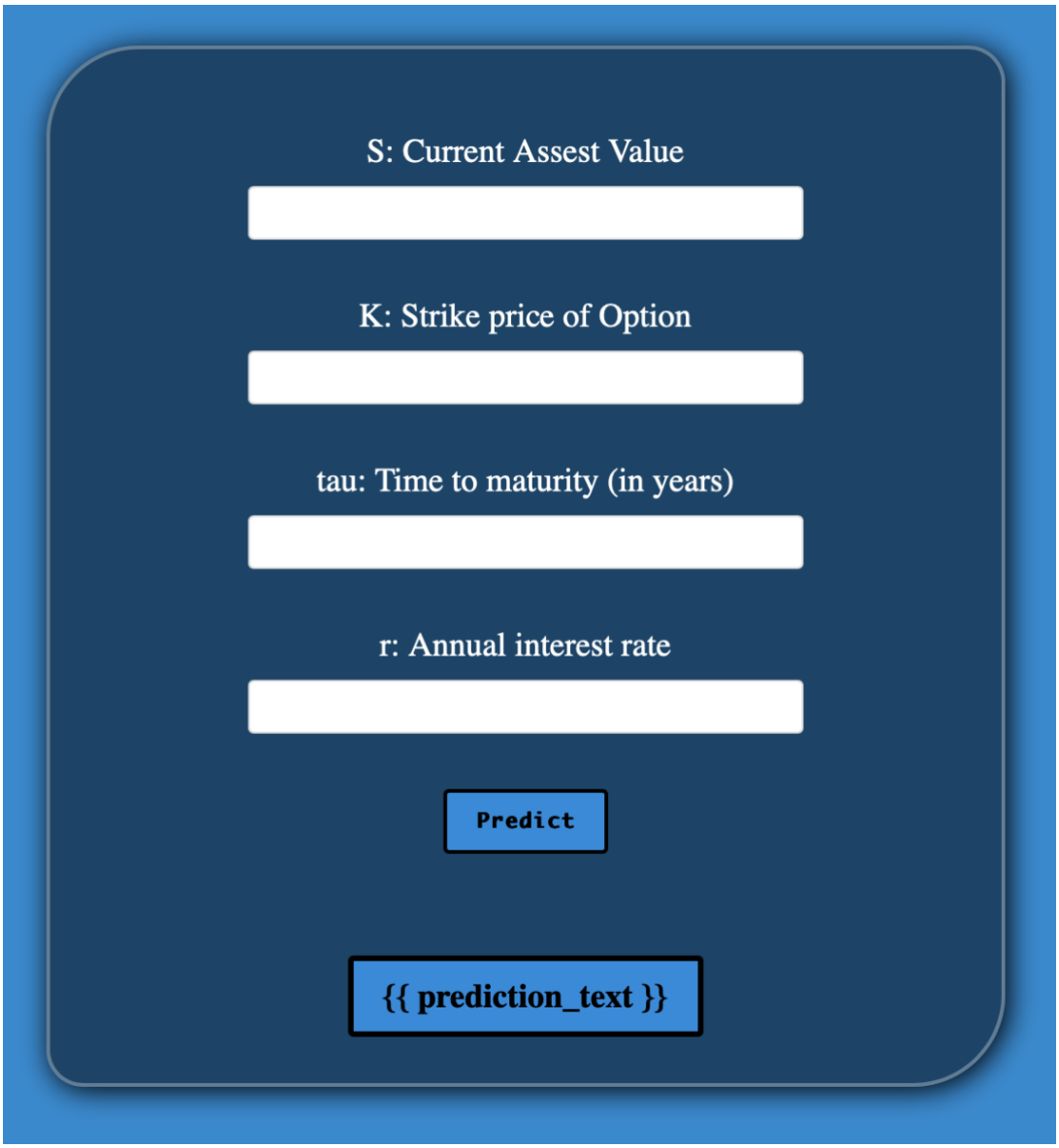
```
/*This section involves the design of the inputs in the form*/

input#s {
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    color: black;
    box-shadow: insert 1px 1px 5px rgba(0, 0, 0, 0.3);
}

input#k {
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: insert 1px 1px 5px rgba(0, 0, 0, 0.3);
}

input#tau {
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: insert 1px 1px 5px rgba(0, 0, 0, 0.3);
}

input#r {
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: insert 1px 1px 5px rgba(0, 0, 0, 0.3);
}
```

```
        /*Responsible for shadow backgrounds*/

        .table {
            display: table;
            margin: 0 auto;
            margin-left: 33.85%;
        }

        ul#horizontal-list {
            min-width: 696px;
            list-style: none;
        }

        /*This section is concerned with the link layout*/

        div.title img {
            display: inline-block;
            vertical-align: middle;
        }

        div.title h1 {
            margin-left: 150px;
            display: inline-block;
            vertical-align: middle;
            padding-left: 10%;
            font-family: "Lucida Handwriting";
        }

    </style>
</head>
```

```
<body>
    <!---Initialize structure of Title and house picture--->
    <div class="title">
        <h1>Option Value Predictor</h1>
        <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRXUvjCXm6o8x01saw
    </div>
```

```
    <!---Containerize main page for styling--->
    <div class="page">
        <!---Containerize paragraph and form for styling--->
        <div class="container">
            <br>
            <!---Initialize form structure and inputs, set method to "POST"--->
            <form action="{{url_for('predict')}}" method="post" class="info">
                <label for="name">S: Current Assest Value</label>
                <input type="text" id="s" name="S: Current Assest Value" required="required" />
                <br>
                <br>
                <label for="name">K: Strike price of Option</label>
                <input type="text" id="k" name="K: Strike price of Option" required="required" />
                <br>
                <br>
                <label for="name">tau: Time to maturity (in years)</label>
                <input type="text" id="tau" name="tau: Time to maturity (in years)" required="required" />
                <br>
                <br>
                <label for="name">r: Annual interest rate</label>
                <br>
                <input type="text" id="r" name="r: Annual interest rate"  required="required" />
                <br>
                <br>
                <button type="submit" class="btn">Predict</button>
            </form>
            <br>
                <!---Set placeholder for prediction output--->
                <div class ="pred">
                    <p class = "result"><b>{{ prediction_text }}</b></p>
                </div>
            </div>
            <br>
            <br>
        </div>
    </body>
</html>
```

2.3 Result Website



Enter the Values and get the prediction: