Name: Jessica Gallo

Batch Code: LISUM12

Submission Date: 09/04/2022

Submitted to: https://github.com/Gallo13/Graduate-Admissions-Predictor-Web-App

https://graduate-admission-prediction2.herokuapp.com/

Information:

In this project, I am using Linear Regression for a predictor app using Python and Flask to give predictions on graduate school admissions. I deployed this model with Heroku.

Data:

The data contains GRE Scores, TOEFL scores, University Rating, Statement of Purpose, Letter of Recommendation, Undergraduate GPA score and research experience.

Filename: Admission_Predict_Ver1.1.csv

Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.5	4.5	8.87	1	0.76
3	316	104	3	3.5	3.5	8	1	0.72
4	322	110	3	2.5	2.5	8.67	1	0.8
5	314	103	2	3	3	8.21	0	0.65

Building Model:

Import libraries and dataset…

```
# Libraries
import pandas as pd
import pickle
from sklearn.linear_model import LinearRegression
```

```
filename = pd.read_csv("C:/Users/Gallo/Downloads/Admission_Predict_Ver1.1.csv")
df = pd.DataFrame(filename)
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Data cleaning…

```
df.drop(['Serial No.'], inplace=True, axis=1)
df.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

Checking for missing data…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

Data Preprocessing…

Separating out the target variable in y.

```
x = df.iloc[:, :7]
y = df.iloc[:, -1]

print('x', x)
print('\ny', y)
```

```
x      GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research
0            337          118                  4  4.5   4.5  9.65         1
1            324          107                  4  4.0   4.5  8.87         1
2            316          104                  3  3.0   3.5  8.00         1
3            322          110                  3  3.5   2.5  8.67         1
4            314          103                  2  2.0   3.0  8.21         0
..           ...          ...                ...  ...   ...   ...       ...
495          332          108                  5  4.5   4.0  9.02         1
496          337          117                  5  5.0   5.0  9.87         1
497          330          120                  5  4.5   5.0  9.56         1
498          312          103                  4  4.0   5.0  8.43         0
499          327          113                  4  4.5   4.5  9.04         0

[500 rows x 7 columns]

y 0      0.92
1       0.76
2       0.72
3       0.80
4       0.65
        ...
495     0.87
496     0.96
497     0.93
498     0.73
499     0.84
Name: Chance of Admit , Length: 500, dtype: float64
```

Build the simple model using Linear Regression

```
regressor = LinearRegression()
x = x.values  # conversion of x into array to remove warning from scikit-learn 1.0
# fit model with training data
regressor.fit(x, y)
```

Save the model with pickle…

```
# save model with pickle
pickle.dump(regressor, open('admission_model.pkl', 'wb'))
```

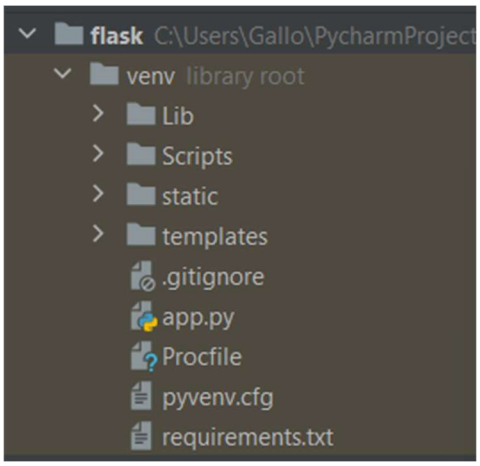
Testing results…

```
# load model to compare the results
model = pickle.load(open('admission_model.pkl', 'rb'))
print(model.predict([[320, 120, 3, 3.5, 4.5, 8.5, 0]]))
```

```
[0.75786656]
```

Flask

Directory



App.py

Python file for main python/flask web app.

```python
from flask import Flask, request, render_template, url_for
import numpy as np
import pickle


app = Flask(__name__)
model= pickle.load(open("C:/Users/Gallo/Jupyter/admission_model.pkl", 'rb'))

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

@app.route('/predict', methods=['POST'])
def predict():
    int_features = [x for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    output = prediction[0]  # do i need the index??
    return render_template('index.html', prediction_text='Graduate admittance chances are: {}%'.format(output))

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

Templates/base.html

Base HTML file for basic HTML for file to run. Can be applied to any web app.

```
<!DOCTYPE HTML>
<HTML lan="en">
<HEAD>
    <META charset="UTF-8">
    <META name="viewport" content="width=device-width, initial-scale=1.0">
    <LINK rel=stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
    {% block head %}{% endblock %}
</HEAD>
<BODY>
{% block body %}{% endblock %}
</BODY>
</HTML>
```

Templates/index.html

```
{% block body %}
<h1 style="text-align:center">Graduation Admissions Predictor</h1>
<!-- Input -->
<form action="{{ url_for('predict')}}" method="post">
    <table>
        <tr>
            <td align="left">GRE Score</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="GRE" name="GRE Score" placeholder="out of 340" required="required" /> </td>
        </tr>
        <tr>
            <td align="left">TOEFL Score</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="TOEFL" name="TOEFL Score" placeholder="out of 120" required="required" /></td>
        </tr>
        <tr>
            <td align="left">University Rating</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="Uni Rating" name="University Rating" placeholder="out of 5" required="required" /></td>
        </tr>
        <tr>
            <td align="left">Statement of Purpose</td>
        </tr>
         <tr>
            <td align="left"><input type="number" id="SOP" name="SOP" placeholder="out of 5" required="required" /></td>
        </tr>
```

```
        <tr>
            <td align="left">Letter of Reccommendation Strength</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="LOR" name="LOR" placeholder="out of 5" required="required" /></td>
        </tr>
        <tr>
            <td align="left">Undergraduate GPA</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="GPA" name="Undergraduate GPA" placeholder="out of 10" required="required" /></td>
        </tr>
        <tr>
            <td align="left">Research Experience</td>
        </tr>
        <tr>
            <td align="left"><input type="number" id="Experience" name="Research Experience" placeholder="either 0 (No) or 1 (Yes)" required="required
        </tr>
        <tr>
            <td><button type="submit" class="btn btn-primary btn-block btn-large">Predict</button></td>
        </tr>
        <tr>
            <td align="left">{{ prediction_text}}</td>
        </tr>
    </table>
</form>
{% endblock %}
```

Static/css/main.css

Stylesheet for webpage.

```
html
    {
        background-image:url({{ url_for('static', filename='images/particle_background.png') }})
        background-size: cover;
    }
body
    {
        font-family: sans-serif;
        height: 700px;
        background-color: blue; /* For browsers that do not support gradients */
        background-image: linear-gradient(to bottom right, blue, yellow);
    }
form
    {
        background-color:rgba(0,180,255,0.4);
        margin: 0 auto;
        width:350px;
    }
td
    {
        font-size: 20px;
        font-family: sans-serif;
    }
input[type=number]
    {
        width: 100%;
        padding: 10px 5px;
    }
```

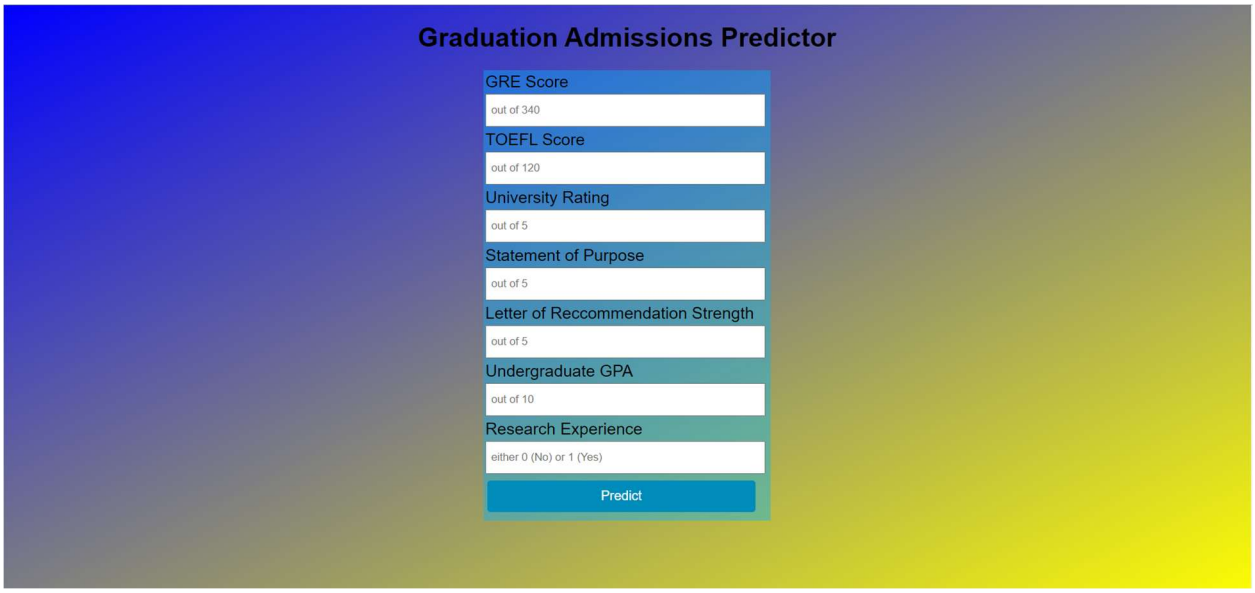
```
button
    {
        background-color: #008CBA;
        border-radius: 4px;
        border: none;
        color: white;
        padding: 10px 25px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 4px 2px;
        cursor: pointer;
        width: 100%;
    }
```

Requirements.txt

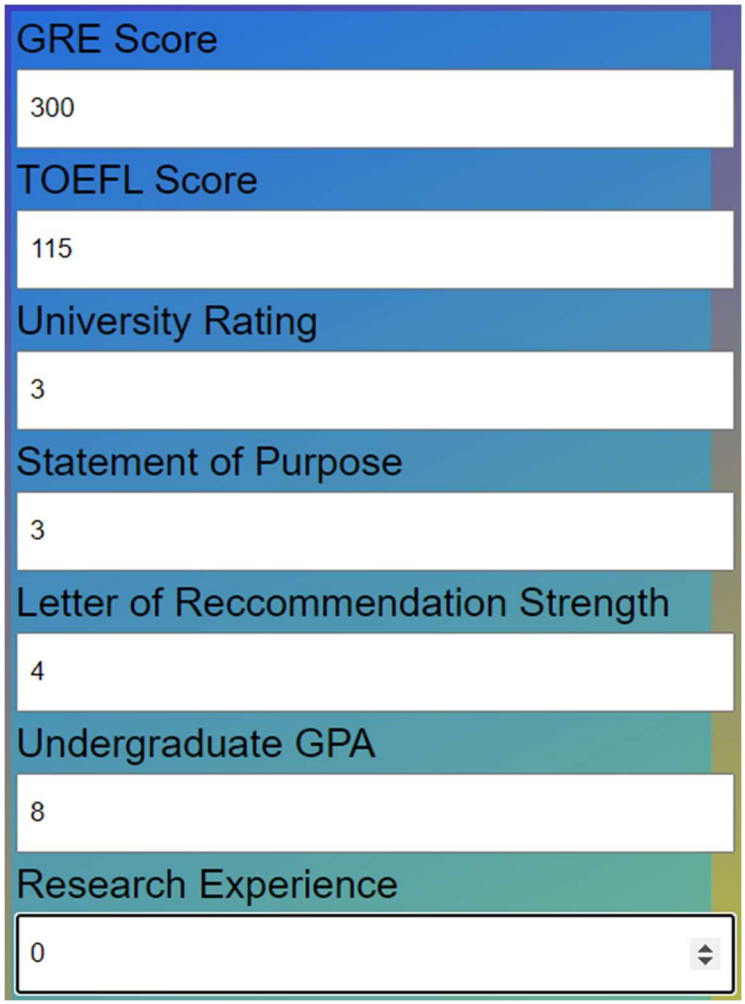
```
click==8.1.3
colorama==0.4.5
Flask==2.2.2
gunicorn==20.1.0
importlib-metadata==4.12.0
itsdangerous==2.1.2
Jinja2==3.1.2
joblib==1.1.0
MarkupSafe==2.1.1
numpy==1.21.6
scikit-learn==1.0.2
scipy==1.7.3
sklearn==0.0
threadpoolctl==3.1.0
typing_extensions==4.3.0
Werkzeug==2.2.2
zipp==3.8.1
```

Running the webapp locally at localhost:5000

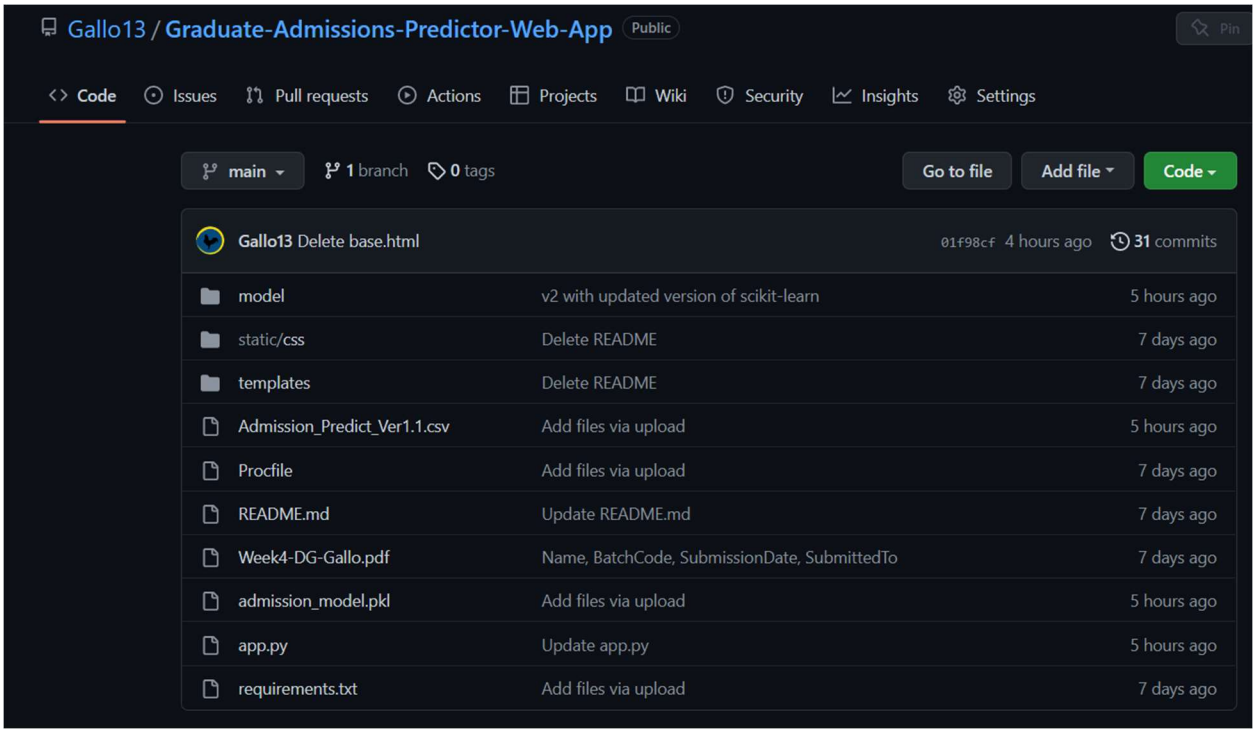
```
C:\Users\Gallo\PycharmProjects\flask\venv\Scripts\python.exe C:/Users/Gallo/PycharmProjects/flask/venv/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 608-243-002
```



Testing input and prediction

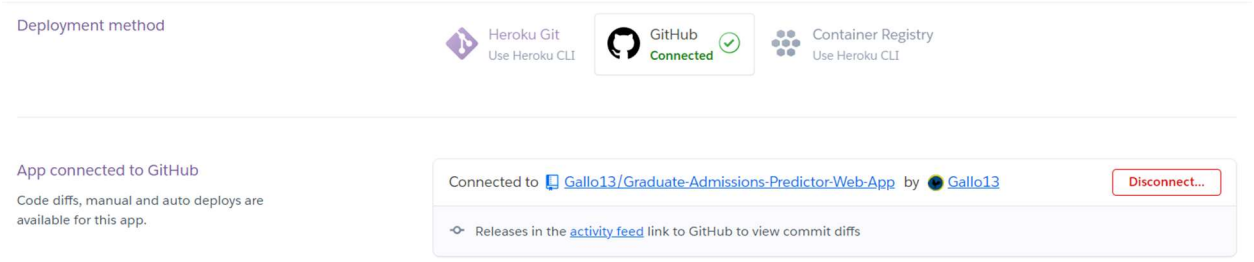

GRE Score
300
TOEFL Score
115
University Rating
3
Statement of Purpose
3
Letter of Reccommendation Strength
4
Undergraduate GPA
8
Research Experience
0
Predict
Graduate admittance chances are:
0.6383916010787356%

Github Repo

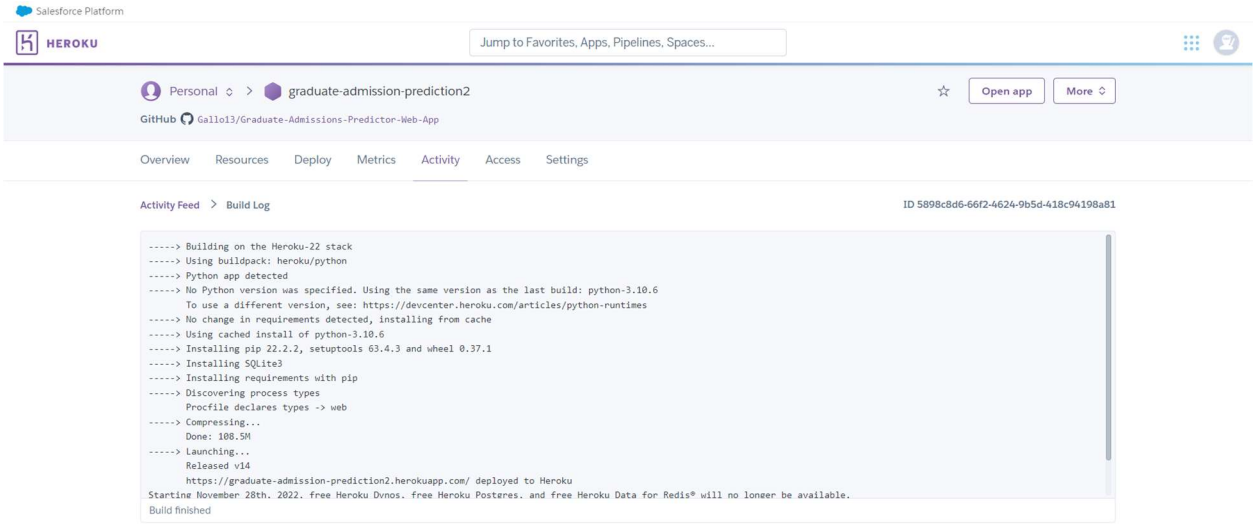


Deployment to Heroku

Create a new project and connect Github Repo to Heroku.



Deploy main branch:



We can now access the webapp at: https://graduate-admission-prediction2.herokuapp.com/