# Week 4 Assignment: Deployment on Flask

**Name**: Freddy F. Tapia C.

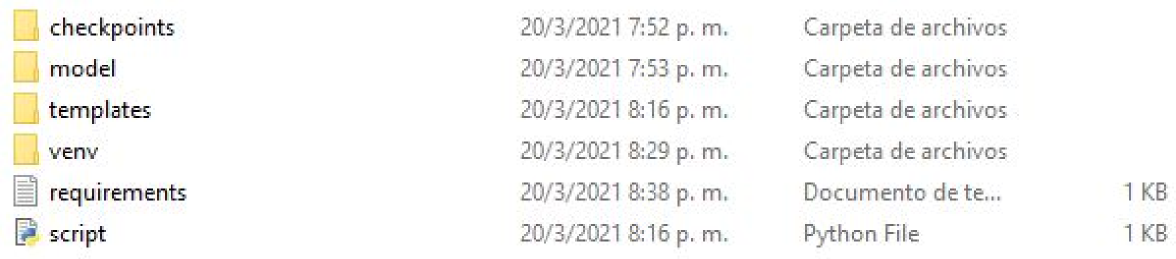
**Submission date**: 2021-03-21

**Submitted to**: Data Glacier

The following document will show the steps that were necessary to complete the task for week 4.

- **Step 1**: create structure and folders for the assignment. This structure must to be similar to,

```
-- checkpoints
---- model_freddy.pkl
-- model
---- Week4_model.py
-- templates
---- index.html
-- venv
-- requirements.txt
-- script.py
```



| | | | |
|---|---|---|---|
| checkpoints | 20/3/2021 7:52 p. m. | Carpeta de archivos | |
| model | 20/3/2021 7:53 p. m. | Carpeta de archivos | |
| templates | 20/3/2021 8:16 p. m. | Carpeta de archivos | |
| venv | 20/3/2021 8:29 p. m. | Carpeta de archivos | |
| requirements | 20/3/2021 8:38 p. m. | Documento de te... | 1 KB |
| script | 20/3/2021 8:16 p. m. | Python File | 1 KB |

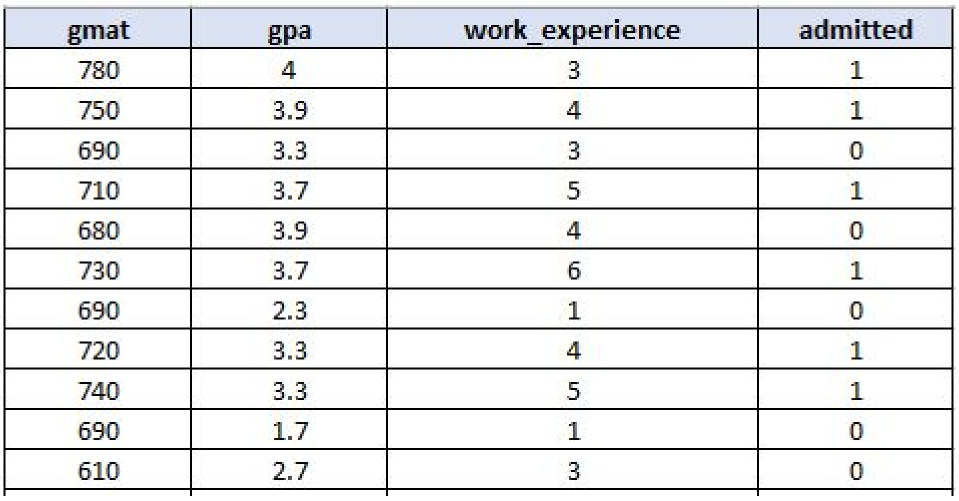
Each folder contains,

1. **checkpoints**: contains the model saved in a pkl format.
2. **model**: contains a python script that was used to generate the model and work with the simple dataset.
3. **templates**: contains a html file ("index.html") which will provide an interface for the user.
4. **venv**: folder created when a virtual environment is generated.

The "requirements.txt" file is useful to know which packages the API will use. The "script.py" file is the main file for the project it contains the code for the API.

- **Step 2**: find a simple data set and build a model in order to do a prediction. Generate a pkl file with the model.

  The head of dataset that i used for this project are,

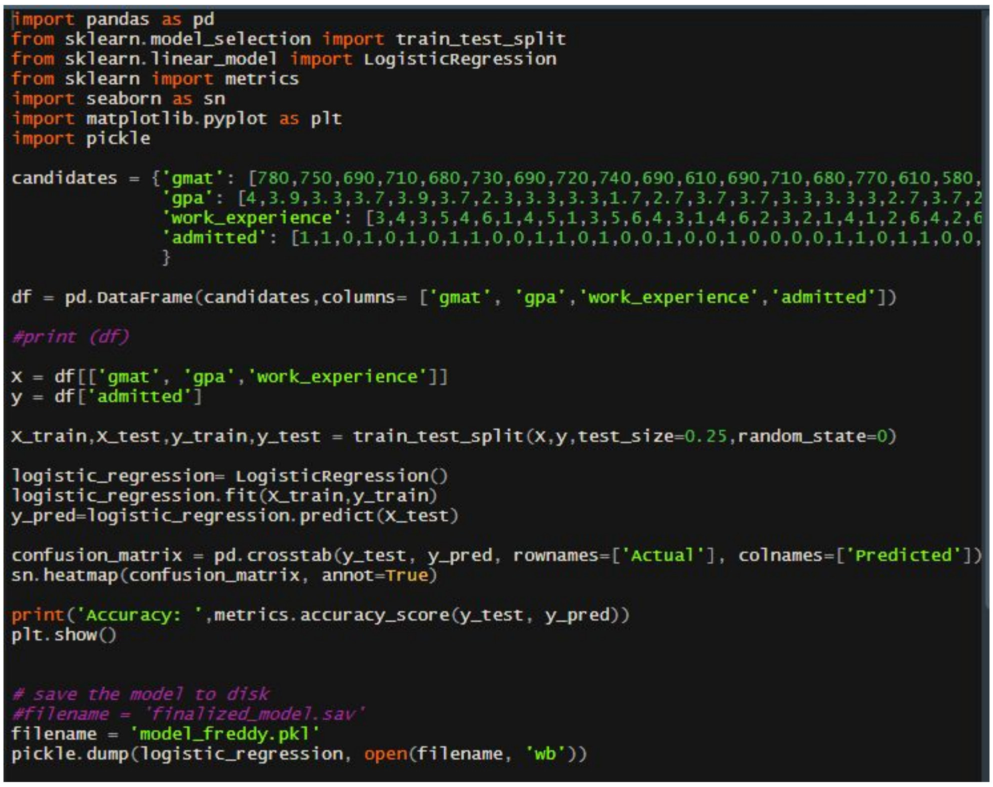


| gmat | gpa | work_experience | admitted |
|---|---|---|---|
| 780 | 4 | 3 | 1 |
| 750 | 3.9 | 4 | 1 |
| 690 | 3.3 | 3 | 0 |
| 710 | 3.7 | 5 | 1 |
| 680 | 3.9 | 4 | 0 |
| 730 | 3.7 | 6 | 1 |
| 690 | 2.3 | 1 | 0 |
| 720 | 3.3 | 4 | 1 |
| 740 | 3.3 | 5 | 1 |
| 690 | 1.7 | 1 | 0 |
| 610 | 2.7 | 3 | 0 |

  where,

  1. gmat: grade at Graduate Management Admission Test.
  2. gpa: grade Point Average.
  3. work_experience: number of years of experience at work.
  4. admitted: binary column, 1 if the student was admitted and 0 if the student was rejected.

The file where the process to obtain the pkl file and build the model is “Week4_model.py”.

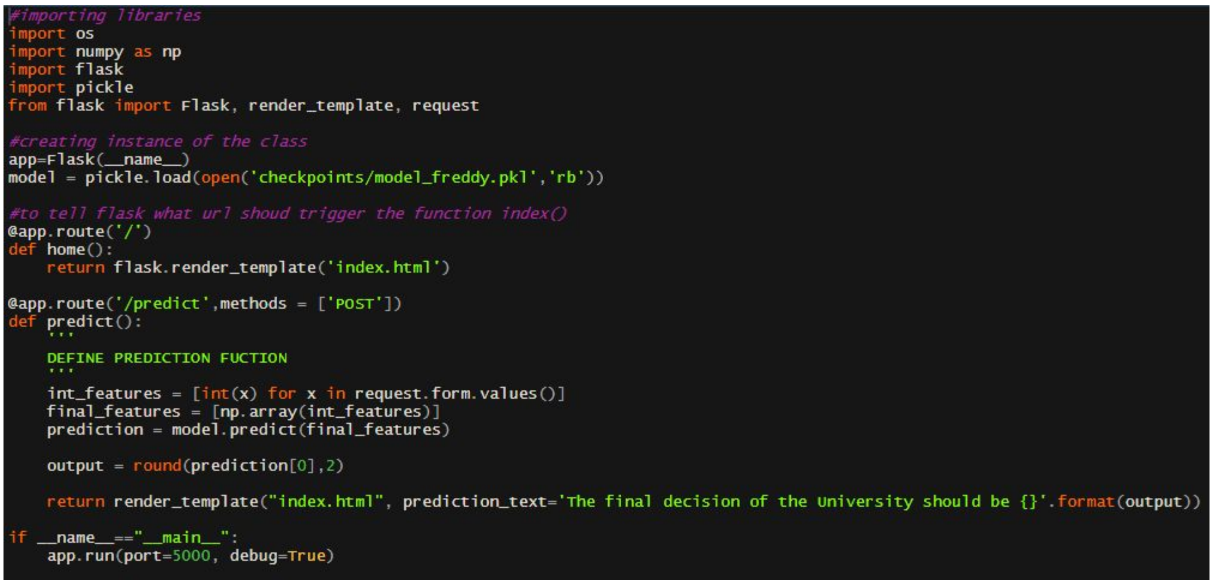


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt
import pickle

candidates = {'gmat': [780,750,690,710,680,730,690,720,740,690,610,690,710,680,770,610,580,
              'gpa': [4,3.9,3.3,3.7,3.9,3.7,2.3,3.3,3.3,1.7,2.7,3.7,3.7,3.3,3.3,3,2.7,3.7,2
              'work_experience': [3,4,3,5,4,6,1,4,5,1,3,5,6,4,3,1,4,6,2,3,2,1,4,1,2,6,4,2,6
              'admitted': [1,1,0,1,0,1,0,1,1,0,0,1,1,0,1,0,0,1,0,0,1,0,0,0,0,1,1,0,1,1,0,0,
              }

df = pd.DataFrame(candidates,columns= ['gmat', 'gpa','work_experience','admitted'])

#print (df)

X = df[['gmat', 'gpa','work_experience']]
y = df['admitted']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)

logistic_regression= LogisticRegression()
logistic_regression.fit(X_train,y_train)
y_pred=logistic_regression.predict(X_test)

confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix, annot=True)

print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
plt.show()


# save the model to disk
#filename = 'finalized_model.sav'
filename = 'model_freddy.pkl'
pickle.dump(logistic_regression, open(filename, 'wb'))
```

- **Step 3**: create the main file for the API (named “script.py” in this case) and write all the functions that we will need.

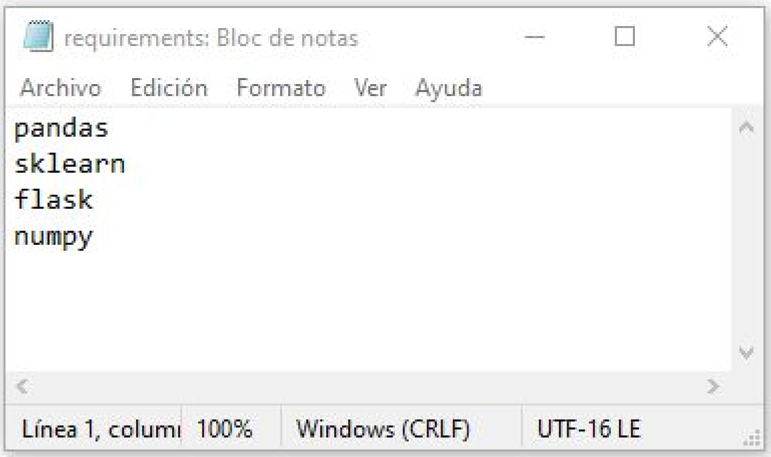
In this file i need to define two main functions, among others important objects,

1. **home**: function that shows the html file.
2. **predict**: function that will get the input of the user and then use the model to do a prediction.



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

#creating instance of the class
app=Flask(__name__)
model = pickle.load(open('checkpoints/model_freddy.pkl','rb'))

#to tell flask what url shoud trigger the function index()
@app.route('/')
def home():
    return flask.render_template('index.html')

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

    output = round(prediction[0],2)

    return render_template("index.html", prediction_text='The final decision of the University should be {}'.format(output))

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

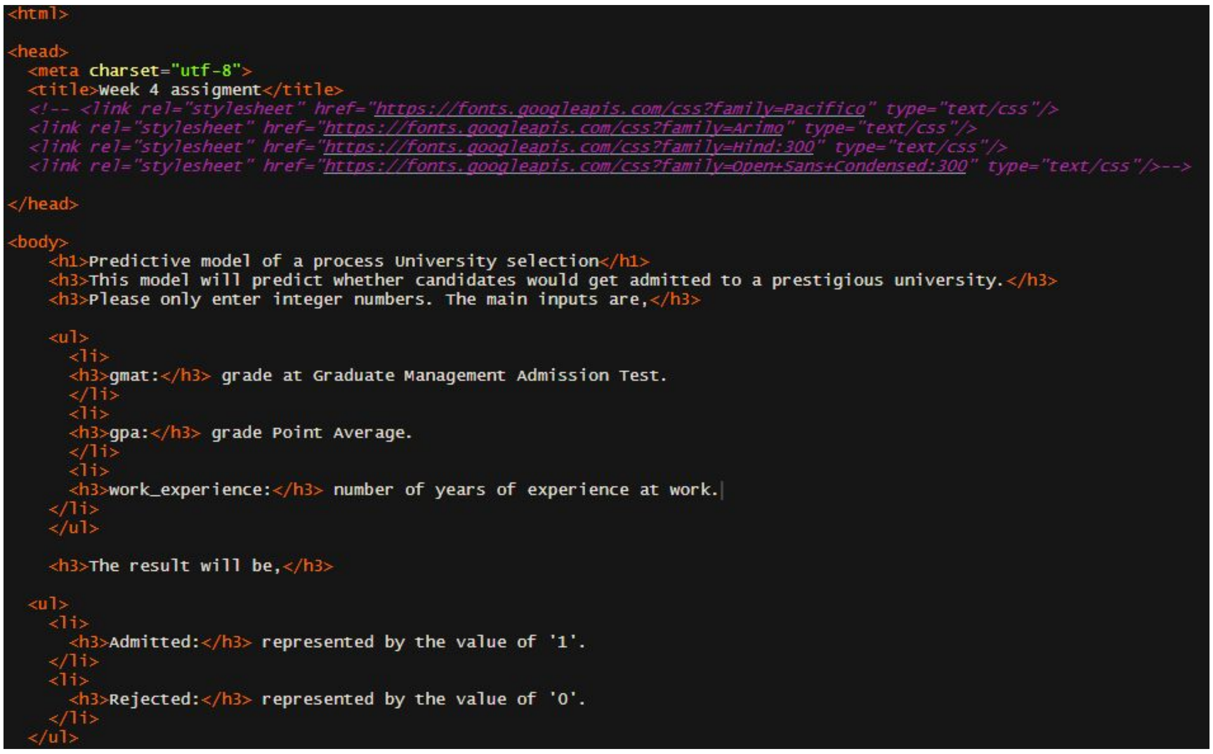
- **Step 4**: create a “requirements.txt” file in order to know the specific version and packages that we need to use. This will be helpful at the moment when a virtual environment is created.



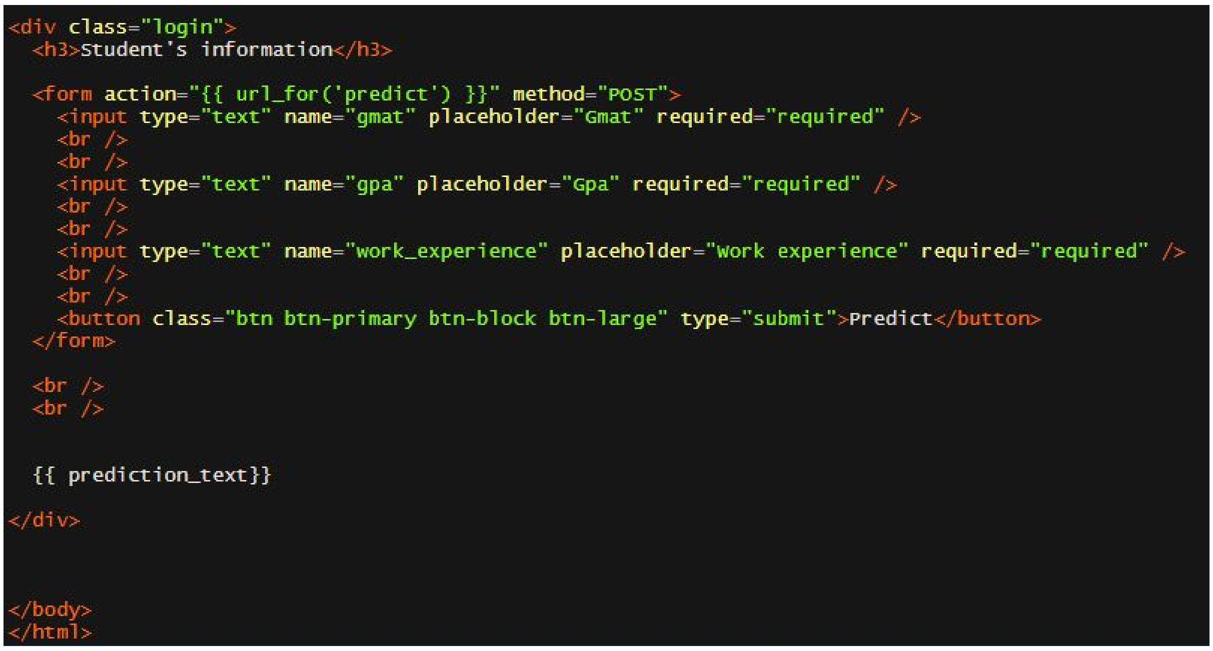
```
pandas
sklearn
flask
numpy
```

- **Step 5**: create the html template (named “index.html” in this case) in order to generate a place for the input of the model and provide an interface to the user.

The most important section of this file is the last “div” where the inputs if the model are requested, in order to connect this file and the API the argument “action” needs to be “{{ url_for(‘predict’) }}‘”.



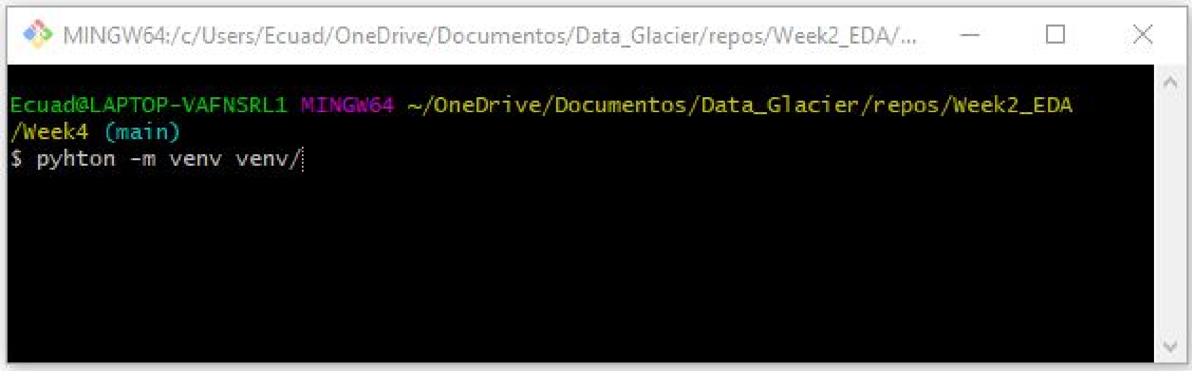
```html
<html>

<head>
  <meta charset="utf-8">
  <title>Week 4 assigment</title>
  <!-- <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Pacifico" type="text/css"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Arimo" type="text/css"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Hind:300" type="text/css"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300" type="text/css"/>-->

</head>

<body>
    <h1>Predictive model of a process University selection</h1>
    <h3>This model will predict whether candidates would get admitted to a prestigious university.</h3>
    <h3>Please only enter integer numbers. The main inputs are,</h3>

    <ul>
      <li>
      <h3>gmat:</h3> grade at Graduate Management Admission Test.
      </li>
      <li>
      <h3>gpa:</h3> grade Point Average.
      </li>
      <li>
      <h3>work_experience:</h3> number of years of experience at work.
    </li>
    </ul>

    <h3>The result will be,</h3>

  <ul>
    <li>
      <h3>Admitted:</h3> represented by the value of '1'.
    </li>
    <li>
      <h3>Rejected:</h3> represented by the value of '0'.
    </li>
  </ul>
```



```html
<div class="login">
  <h3>Student's information</h3>

  <form action="{{ url_for('predict') }}" method="POST">
    <input type="text" name="gmat" placeholder="Gmat" required="required" />
    <br />
    <br />
    <input type="text" name="gpa" placeholder="Gpa" required="required" />
    <br />
    <br />
    <input type="text" name="work_experience" placeholder="Work experience" required="required" />
    <br />
    <br />
    <button class="btn btn-primary btn-block btn-large" type="submit">Predict</button>
  </form>

  <br />
  <br />


  {{ prediction_text}}

</div>



</body>
</html>
```

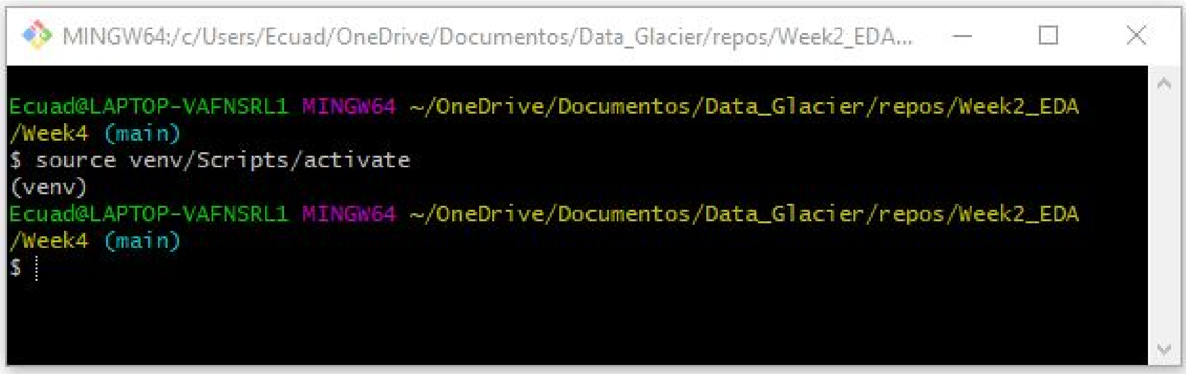
- **Step 6**: generate a virtual environment, activate it and install the packages that are in the “requirements.txt” file. After that process runs the API.

To generate a virtual environment, this command will be used “python -m venv venv/”. This will create a folder named “venv” in the current path.



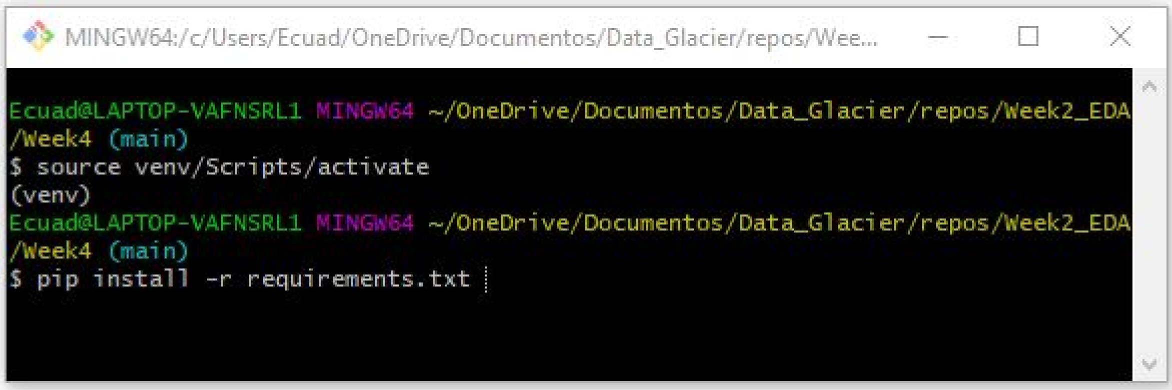
```
MINGW64:/c/Users/Ecuad/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA/...

Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ pyhton -m venv venv/
```

To activate the virtual environment, this command will be used “source venv/Scripts/activate”,



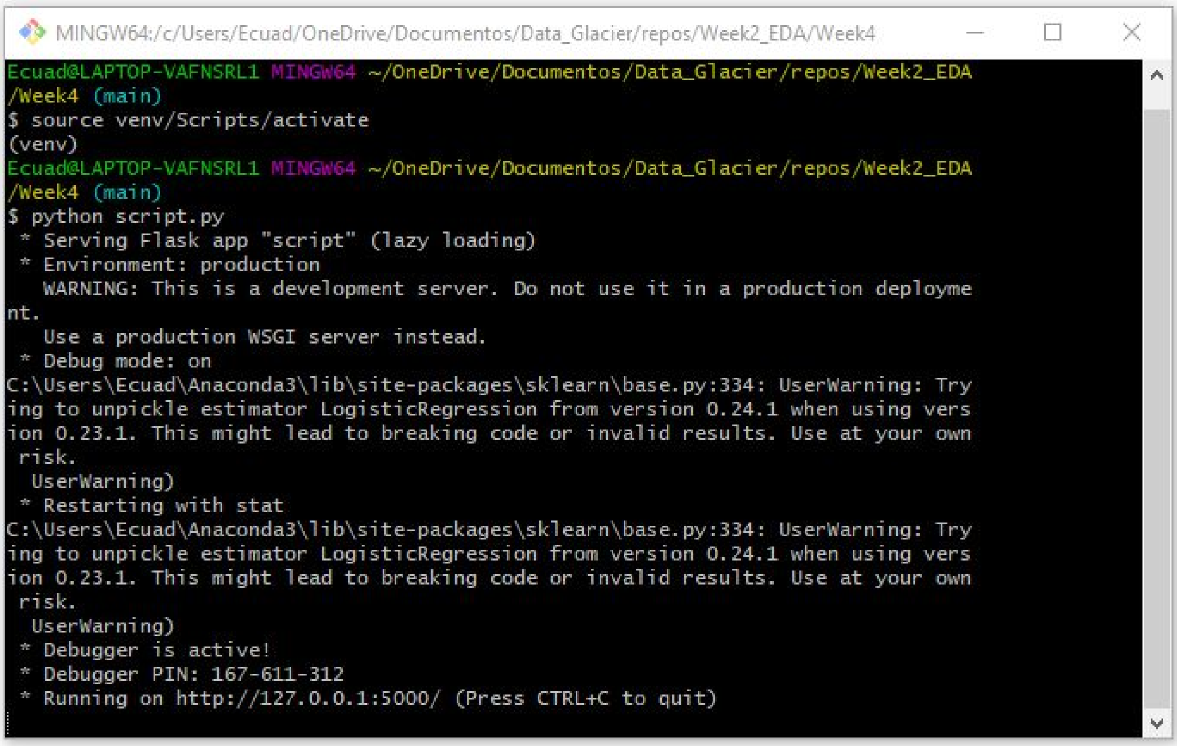
```
MINGW64:/c/Users/Ecuad/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA...

Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ source venv/Scripts/activate
(venv)
Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$
```

To install the packages that are in “requirements.txt” files, this command will be used “pip install -r requirements.txt”,



```
MINGW64:/c/Users/Ecuad/OneDrive/Documentos/Data_Glacier/repos/Wee...

Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ source venv/Scripts/activate
(venv)
Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ pip install -r requirements.txt
```

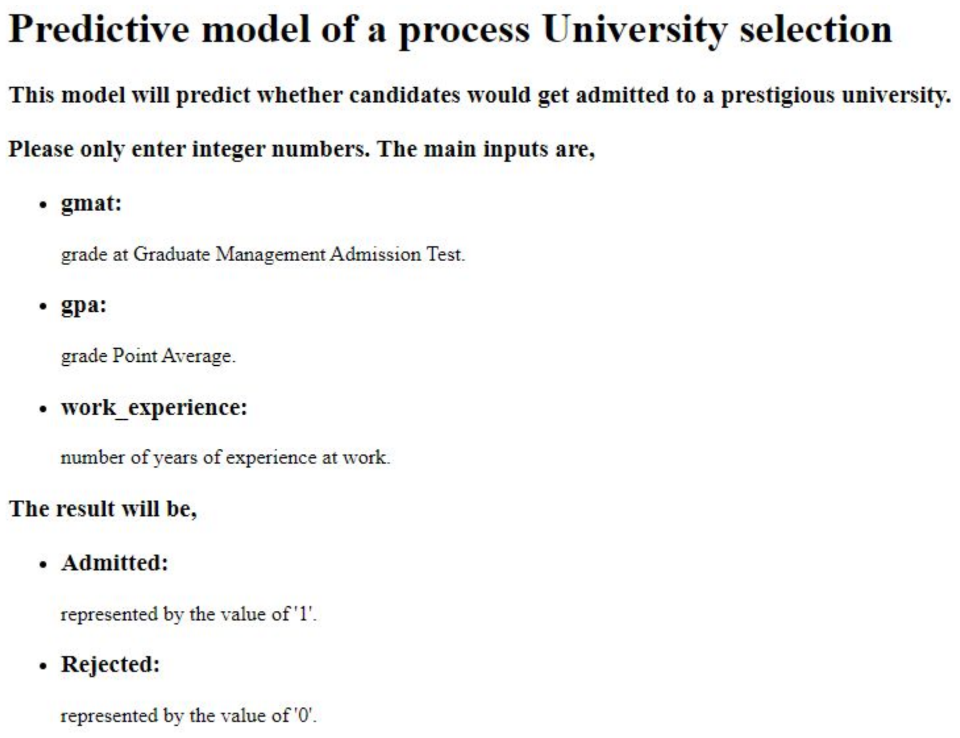
Finally to run the API, this command will be used “python script.py”


MINGW64:/c/Users/Ecuad/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA/Week4

```
Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ source venv/Scripts/activate
(venv)
Ecuad@LAPTOP-VAFNSRL1 MINGW64 ~/OneDrive/Documentos/Data_Glacier/repos/Week2_EDA
/Week4 (main)
$ python script.py
 * Serving Flask app "script" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
C:\Users\Ecuad\Anaconda3\lib\site-packages\sklearn\base.py:334: UserWarning: Try
ing to unpickle estimator LogisticRegression from version 0.24.1 when using vers
ion 0.23.1. This might lead to breaking code or invalid results. Use at your own
 risk.
  UserWarning)
 * Restarting with stat
C:\Users\Ecuad\Anaconda3\lib\site-packages\sklearn\base.py:334: UserWarning: Try
ing to unpickle estimator LogisticRegression from version 0.24.1 when using vers
ion 0.23.1. This might lead to breaking code or invalid results. Use at your own
 risk.
  UserWarning)
 * Debugger is active!
 * Debugger PIN: 167-611-312
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- **Step 7**: go to the url of the web app and use it, do a pair of examples in order to see the results.

The first section of the web app is an explanation of how to use the app and what is the main purpose of it.



**Predictive model of a process University selection**

**This model will predict whether candidates would get admitted to a prestigious university.**

**Please only enter integer numbers. The main inputs are,**

- **gmat:**

  grade at Graduate Management Admission Test.

- **gpa:**

  grade Point Average.

- **work_experience:**

  number of years of experience at work.

**The result will be,**

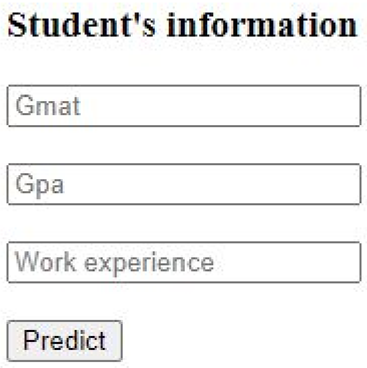
- **Admitted:**

  represented by the value of '1'.

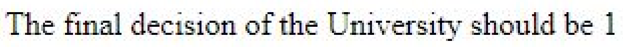
- **Rejected:**

  represented by the value of '0'.

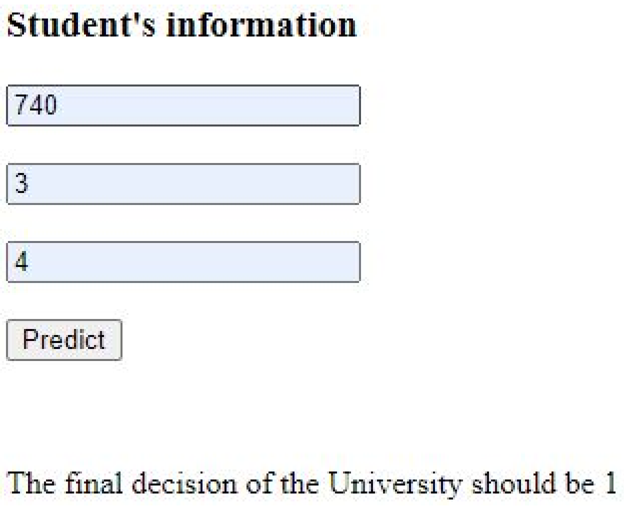
The second section is for the inputs,



and finally the third section is for the results of the prediction of the model,


The final decision of the University should be 1

Example of approved student:


Student's information

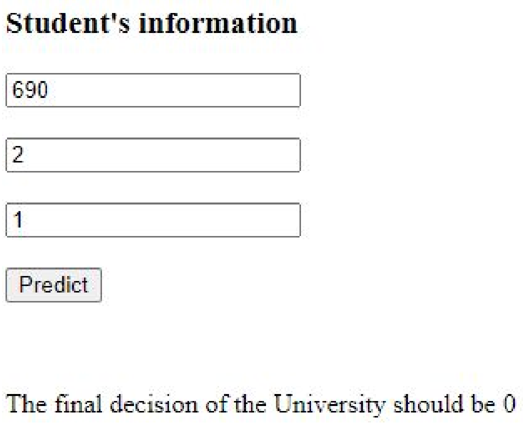
740

3

4

The final decision of the University should be 1

Example of rejected student:


Student's information

690

2

1

The final decision of the University should be 0