Deployment on Flask

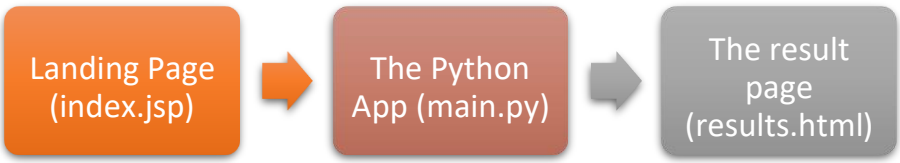
Name – Parth Shah

Batch code – LISUM01

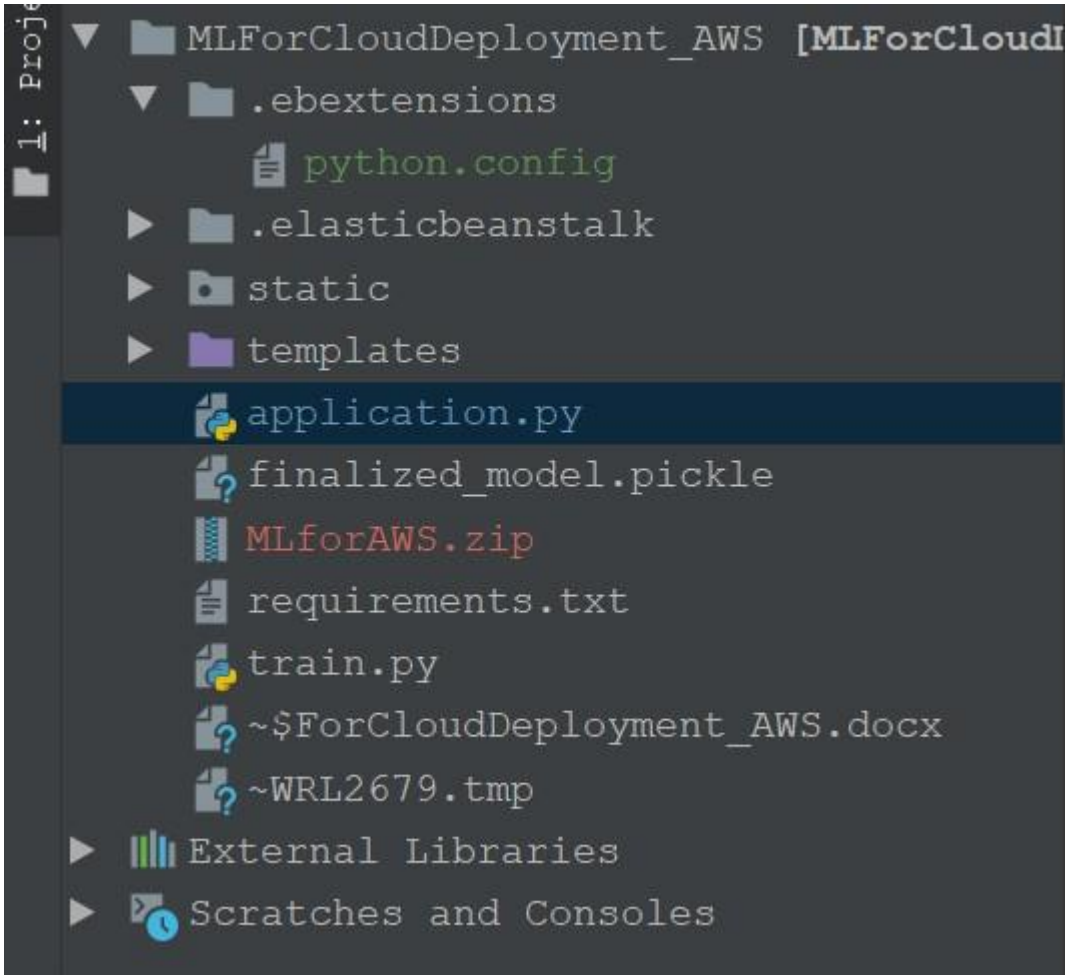
Submission Date – 03/07/2021

Flask App:

As we'll expose the created model as a web API to be consumed by the client/client APIs, we'd do it using the flask framework. The flow of our flask app will be:



Create the project structure, as shown below:



Only create the files and folders(marked in yellow), and put the saved model file in the same folder as your app.py file.

Index.html

```
• {% extends 'base.html' %}

  {% block head %}

  <title>Search Page</title>
  <link rel="stylesheet" href="{{ url_for('static',
  filename='css/style.css') }}">
  {% endblock %}

  {% block body %}
  <div class="content">
      <h1 style="text-align: center">Predict Your chances for
  Admission</h1>

      <div class="form">
          <form action="/predict" method="POST">
              <input type="number" name="gre_score" id="gre_score"
  placeholder="GRE Score">
              <input type="number" name="toefl_score" id="toefl_score"
  placeholder="TOEFL Score">
              <input type="number" name="university_rating"
```

```
# importing the necessary dependencies
from flask import Flask, render_template,
request,jsonify from flask_cors import CORS,cross_origin
import pickle

app = Flask(__name__) # initializing a flask app
   @app.route('/',methods=['GET'])  # route to display the home page
     @cross_origin() def homePage():
  return render_template("index.html")

   @app.route('/predict',methods=['POST','GET']) # route to show the
                          predictions in a web UI
@cross_origin() def
      index():
                        if request.method == 'POST':
                                        try:
           #  reading the inputs given by the user
         gre_score=float(request.form['gre_score'])
      toefl_score = float(request.form['toefl_score'])
                      university_rating =
 float(request.form['university_rating'])             sop =
        float(request.form['sop'])             lor =
        float(request.form['lor'])             cgpa =
    float(request.form['cgpa'])             is_research =
                  request.form['research']
     if(is_research=='yes'):                 research=1
              else:                 research=0
                  filename = 'finalized_model.pickle'
  loaded_model = pickle.load(open(filename, 'rb')) # loading the model
                          file from the storage
               # predictions using the loaded model file





prediction=loaded_model.predict([[gre_score,toefl_score,university_rati
                       ng,sop,lor,cgpa,research]])
  print('prediction is', prediction)             #
       showing the prediction results in a UI
                        return
render_template('results.html',prediction=round(100*prediction[0]))
except Exception as e:             print('The Exception message is:
     ',e)             return 'something is wrong.'     # return
      render_template('results.html')     else:         return
                    render_template('index.html')

if __name__ == "__main__":
           #app.run(host='127.0.0.1', port=8001, debug=True)
                 app.run(debug=True) # running the app
```

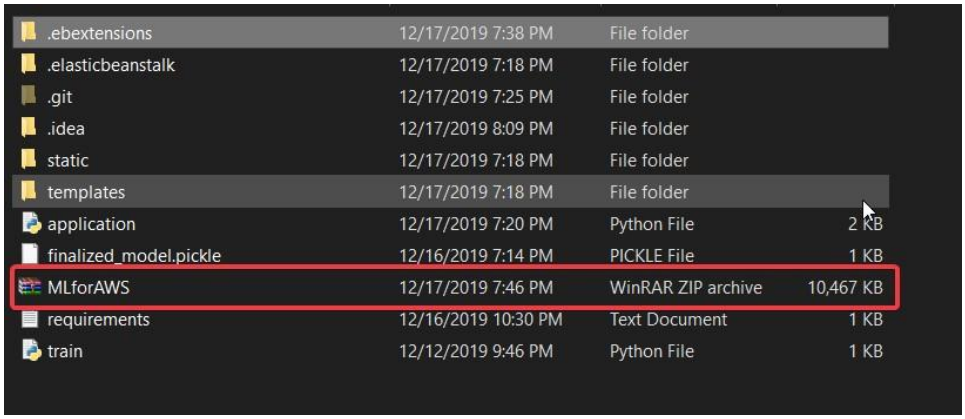
application.py:

results.html:

```
<!DOCTYPE html>
<html lang="en" >

<head>
  <meta charset="UTF-8">
  <title>Review Page</title>

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normali
ze.min.css">


      <link rel="stylesheet" href="./style.css">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">


</head>

<body>

  <div class="table-users">
   <div class="header">Prediction</div>

      <p>Your chance for admission is {{prediction}} percent</p>
</div>



</body>

</html>
```

Deployment to AWS:

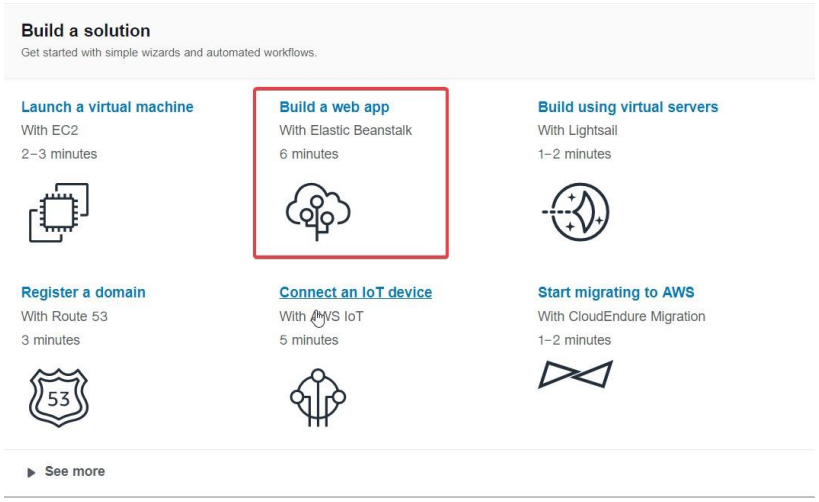
- The python application file should be named application.py
- Create a requirements.txt using pip freeze > requirements.txt from the project folder
- Create a folder '.ebextensions' and create a file 'python.config' inside it. Make sure to populate the content of python.config, as shown above.
- Create the zip file from the project folder itself.



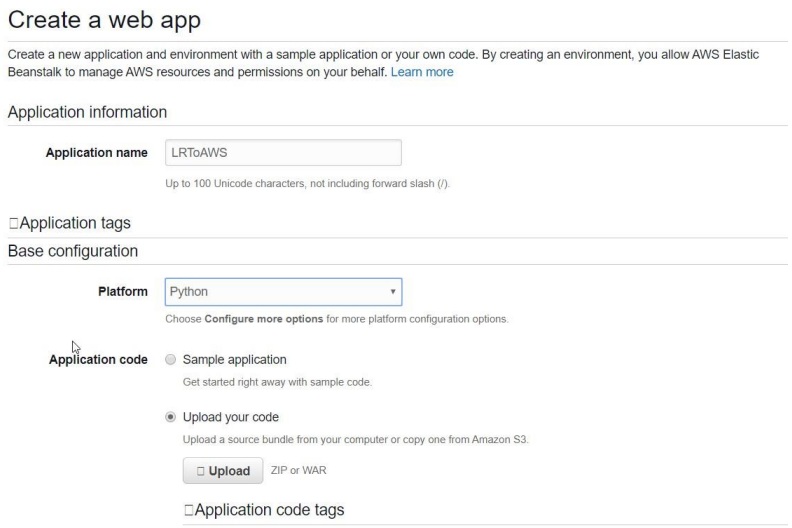
.ebextensions	12/17/2019 7:38 PM	File folder	
.elasticbeanstalk	12/17/2019 7:18 PM	File folder	
.git	12/17/2019 7:25 PM	File folder	
.idea	12/17/2019 8:09 PM	File folder	
static	12/17/2019 7:18 PM	File folder	
templates	12/17/2019 7:18 PM	File folder	
application	12/17/2019 7:20 PM	Python File	2 KB
finalized_model.pickle	12/16/2019 7:14 PM	PICKLE File	1 KB
MLforAWS	12/17/2019 7:46 PM	WinRAR ZIP archive	10,467 KB
requirements	12/16/2019 10:30 PM	Text Document	1 KB
train	12/12/2019 9:46 PM	Python File	1 KB

Deployment Process

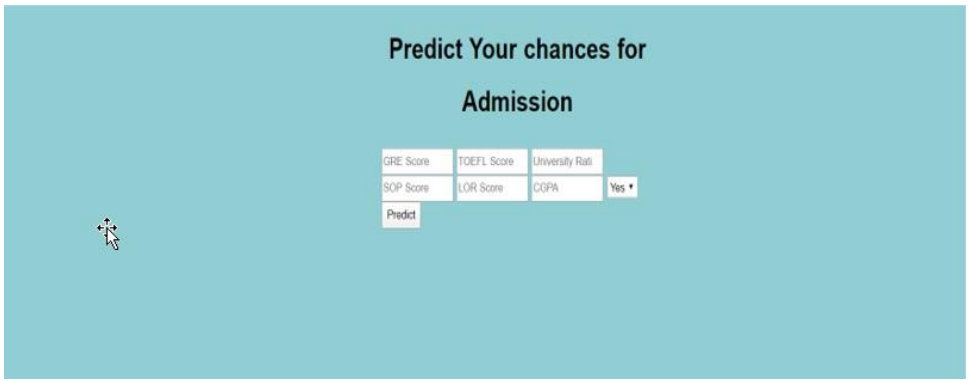
- Go to https://aws.amazon.com/ and create an account if already don't have one.
- Go to the console and go to the 'Build a web app' section and click it.



- Give the name of the application, give platform as python, and select the option to upload your code.



Final Result:



Thank You!