Flask App Deployment on AWS Elastic Beanstalk

Name: Asher Chok

Batch code: LISUM17

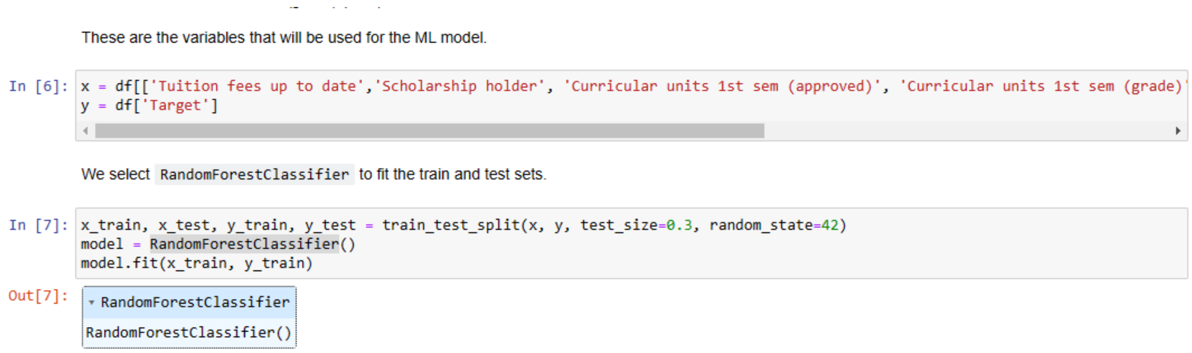
Submission date: 1/28/2023

Submitted to: Data Glacier

Dataset from: [Predict students' dropout and academic success | Kaggle]

Local Flask App Deployment

A notebook was created for making a simple ML model for the student dataset.



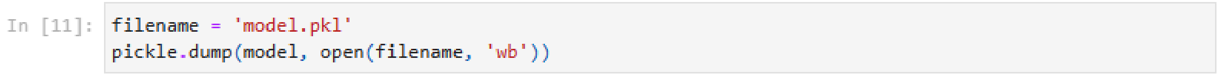
These are the variables that will be used for the ML model.

```
In [6]: x = df[['Tuition fees up to date','Scholarship holder', 'Curricular units 1st sem (approved)', 'Curricular units 1st sem (grade)'
        y = df['Target']
```

We select `RandomForestClassifier` to fit the train and test sets.

```
In [7]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
        model = RandomForestClassifier()
        model.fit(x_train, y_train)

Out[7]: ▾ RandomForestClassifier
        RandomForestClassifier()
```

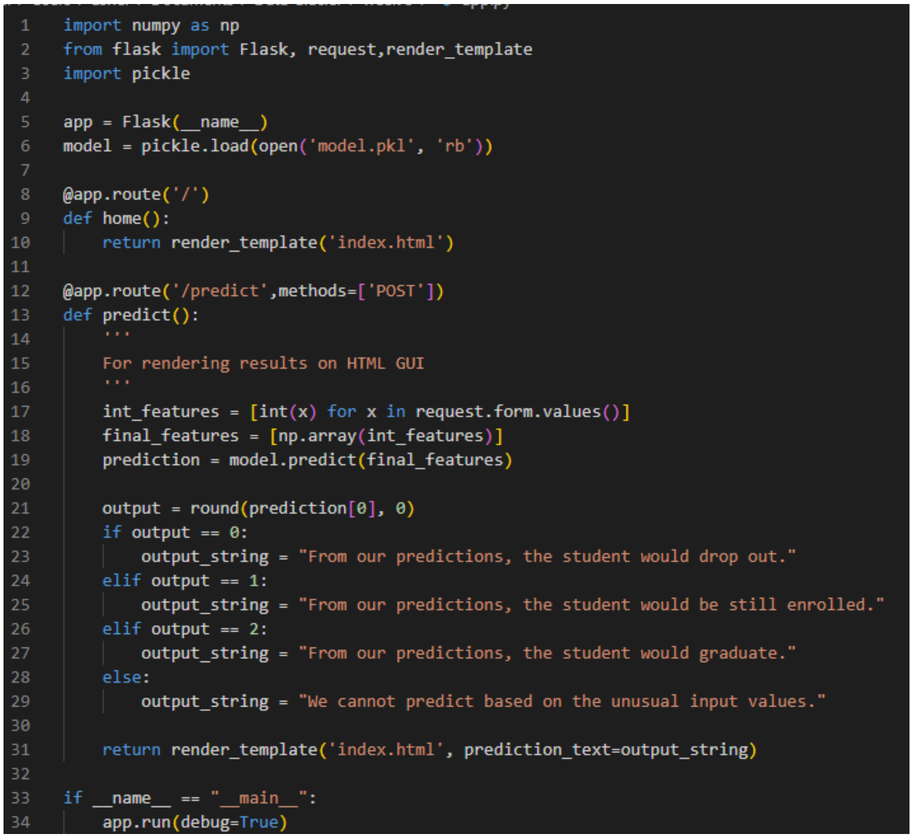
The model was then pickled (a process that saves model to a file on disk in binary format that can be used to make prediction) with the code block as shown below.



```
In [11]: filename = 'model.pkl'
         pickle.dump(model, open(filename, 'wb'))
```

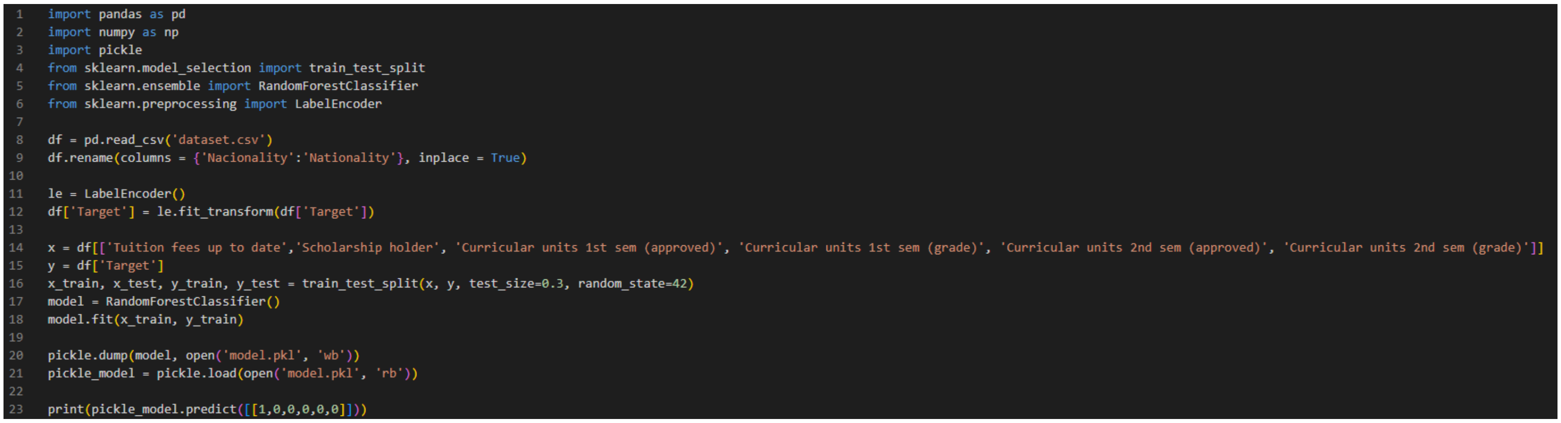
Two python files were written for Flask deployment:

1) *app.py*: A python file for creating and configuring the Flask application, handling requests and defining routes and views.



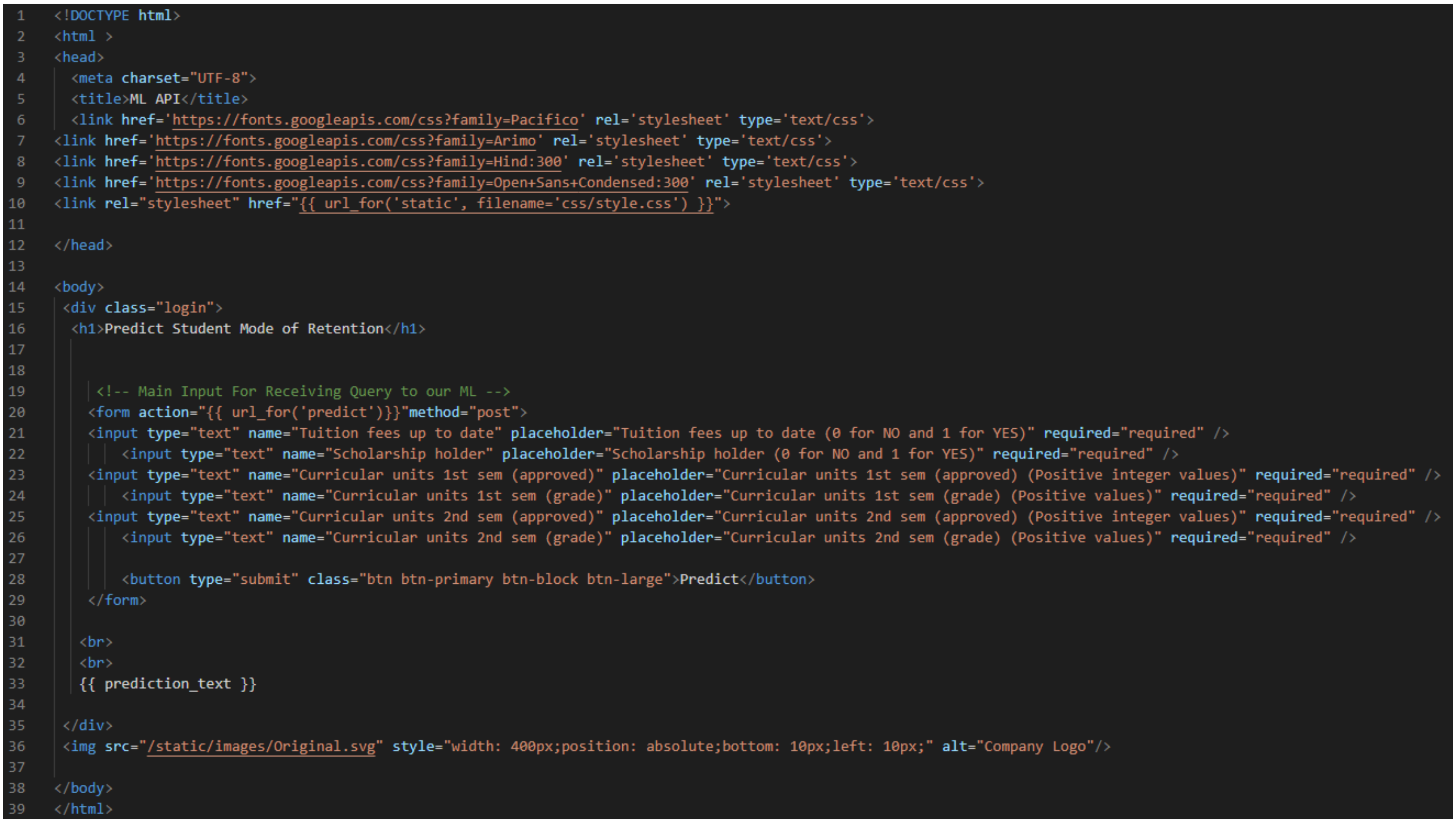
```python
import numpy as np
from flask import Flask, request,render_template
import pickle

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

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

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    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], 0)
    if output == 0:
        output_string = "From our predictions, the student would drop out."
    elif output == 1:
        output_string = "From our predictions, the student would be still enrolled."
    elif output == 2:
        output_string = "From our predictions, the student would graduate."
    else:
        output_string = "We cannot predict based on the unusual input values."

    return render_template('index.html', prediction_text=output_string)

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

2) *model.py*: A python file that loads the trained ML model and provides an interface for making predictions.



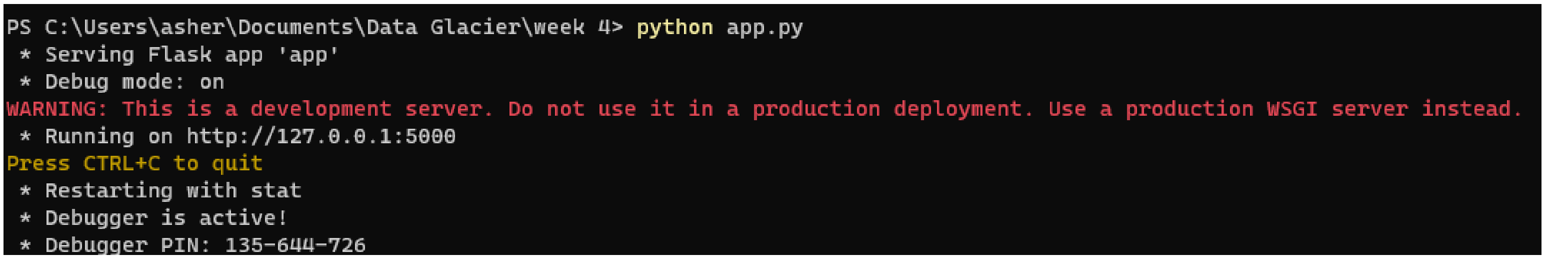
```python
import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('dataset.csv')
df.rename(columns = {'Nacionality':'Nationality'}, inplace = True)

le = LabelEncoder()
df['Target'] = le.fit_transform(df['Target'])

x = df[['Tuition fees up to date','Scholarship holder', 'Curricular units 1st sem (approved)', 'Curricular units 1st sem (grade)', 'Curricular units 2nd sem (approved)', 'Curricular units 2nd sem (grade)']]
y = df['Target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
model = RandomForestClassifier()
model.fit(x_train, y_train)

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

print(pickle_model.predict([[1,0,0,0,0,0]]))
```

Then a HTML file *index.html* was created for the Flask web app.



```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
  <h1>Predict Student Mode of Retention</h1>


    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
    <input type="text" name="Tuition fees up to date" placeholder="Tuition fees up to date (0 for NO and 1 for YES)" required="required" />
        <input type="text" name="Scholarship holder" placeholder="Scholarship holder (0 for NO and 1 for YES)" required="required" />
    <input type="text" name="Curricular units 1st sem (approved)" placeholder="Curricular units 1st sem (approved) (Positive integer values)" required="required" />
        <input type="text" name="Curricular units 1st sem (grade)" placeholder="Curricular units 1st sem (grade) (Positive values)" required="required" />
    <input type="text" name="Curricular units 2nd sem (approved)" placeholder="Curricular units 2nd sem (approved) (Positive integer values)" required="required" />
        <input type="text" name="Curricular units 2nd sem (grade)" placeholder="Curricular units 2nd sem (grade) (Positive values)" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
 <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company Logo"/>

</body>
</html>
```

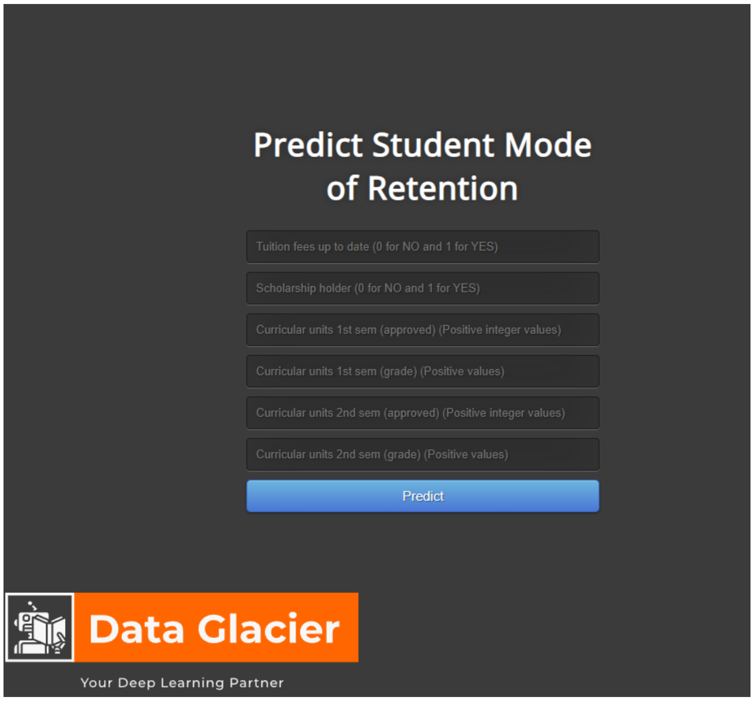
Pickle was then installed in Windows PowerShell terminal with *pip install pickle* command. Then, the following command *python app.py* was run to display the local web app:



```
PS C:\Users\asher\Documents\Data Glacier\week 4> python 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: 135-644-726
```

CTRL+click on the link http://127.0.0.1:5000 redirects user to the development server.

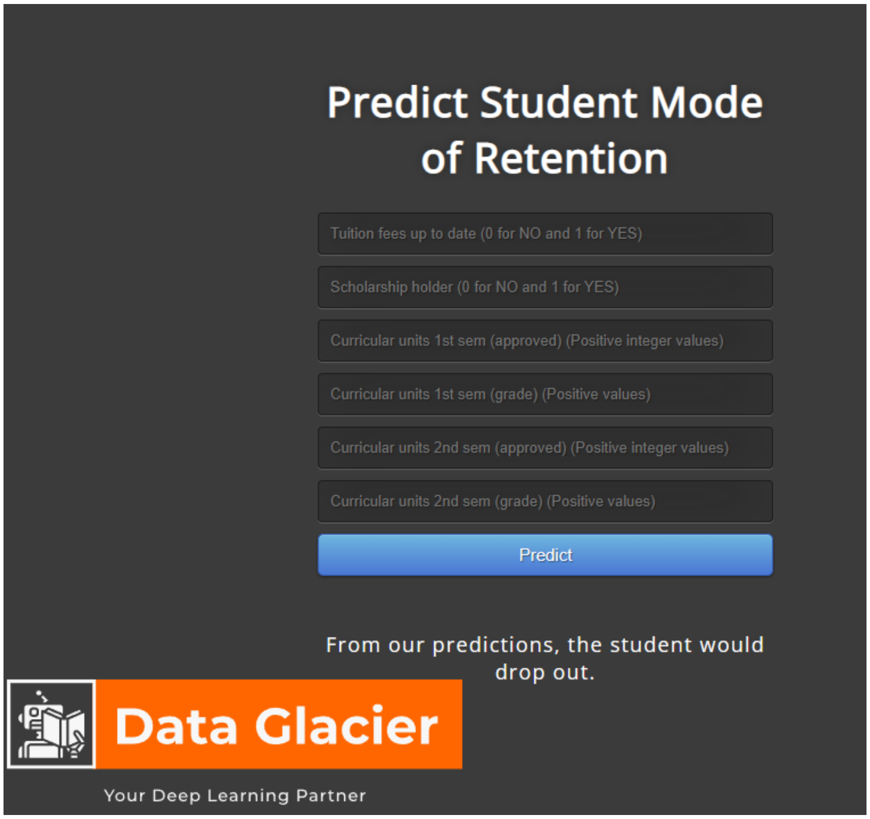
The displayed webpage on localhost shows:



Predict Student Mode of Retention

Tuition fees up to date (0 for NO and 1 for YES)
Scholarship holder (0 for NO and 1 for YES)
Curricular units 1st sem (approved) (Positive integer values)
Curricular units 1st sem (grade) (Positive values)
Curricular units 2nd sem (approved) (Positive integer values)
Curricular units 2nd sem (grade) (Positive values)
Predict
Data Glacier
Your Deep Learning Partner

When we try to enter sample row values into the fields, we get the expected target value from the original dataset which means that the model works as expected. For demonstrating purpose, below is what was shown when [1,0,0,0,0,0] was entered into the fields:



Predict Student Mode of Retention

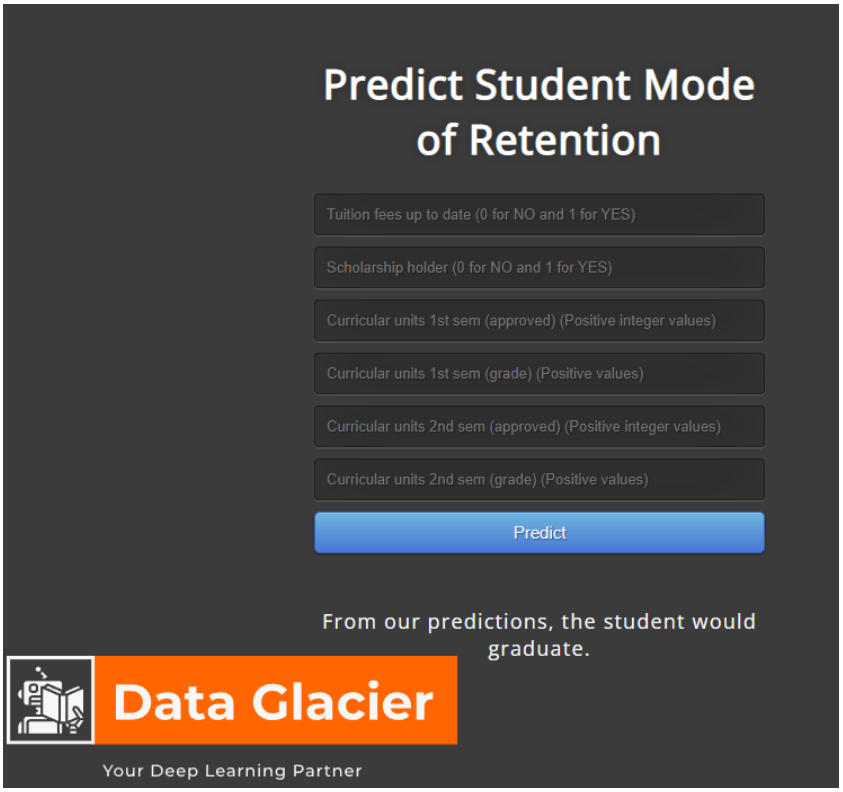
Tuition fees up to date (0 for NO and 1 for YES)
Scholarship holder (0 for NO and 1 for YES)
Curricular units 1st sem (approved) (Positive integer values)
Curricular units 1st sem (grade) (Positive values)
Curricular units 2nd sem (approved) (Positive integer values)
Curricular units 2nd sem (grade) (Positive values)
Predict
Data Glacier
Your Deep Learning Partner

From our predictions, the student would drop out.

Another example case can be tested where a student:

- has tuition fees up to date
- is a scholarship holder
- has 15 curricular units for the two semesters

gives output value of 2, which returns the desired string from *app.py*.



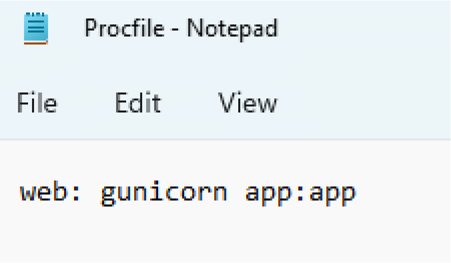
Predict Student Mode of Retention

Tuition fees up to date (0 for NO and 1 for YES)
Scholarship holder (0 for NO and 1 for YES)
Curricular units 1st sem (approved) (Positive integer values)
Curricular units 1st sem (grade) (Positive values)
Curricular units 2nd sem (approved) (Positive integer values)
Curricular units 2nd sem (grade) (Positive values)
Predict
Data Glacier
Your Deep Learning Partner

From our predictions, the student would graduate.

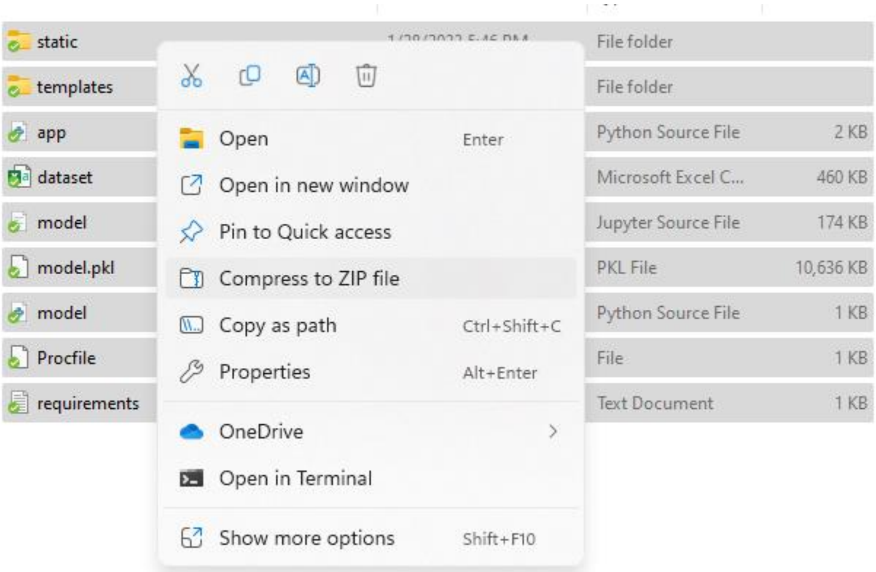
Flask App Deployment with AWS Elastic Beanstalk

Before deploying the app to AWS Elastic Beanstalk, the requirement file must be modified to match Elastic Beanstalk's python environment to allow seamless deployment of the flask app. Additionally, it is important to include a *Procfile* in the root directory of the project, as Elastic Beanstalk uses this file to determine the command to run the app. Without a *Procfile*, the deployment will fail.



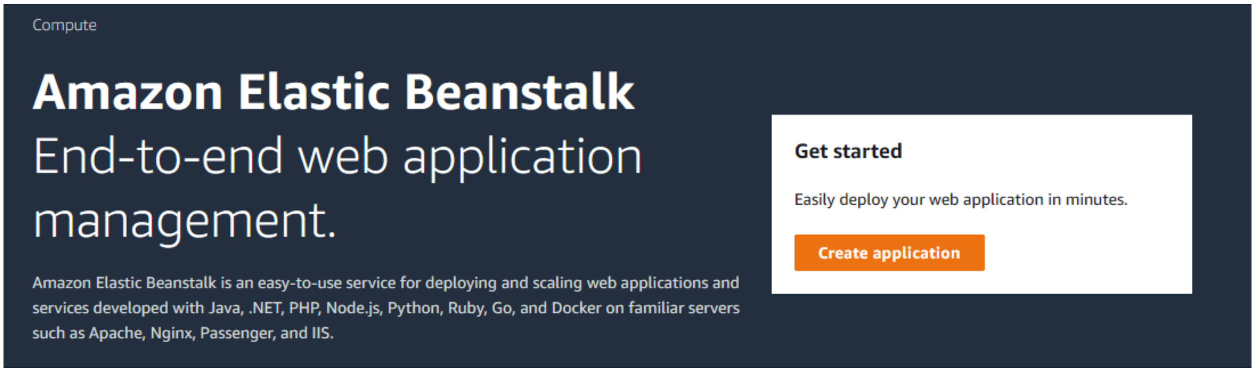
```
web: gunicorn app:app
```

Once the required files were created, all the required files were then compressed into a ZIP file.



static	File folder	
templates	File folder	
app	Python Source File	2 KB
dataset	Microsoft Excel C...	460 KB
model	Jupyter Source File	174 KB
model.pkl	PKL File	10,636 KB
model	Python Source File	1 KB
Procfile	File	1 KB
requirements	Text Document	1 KB

After creating an AWS account and then installing AWS Elastic Beanstalk CLI by following detailed instructions from here, we proceed to Amazon Elastic Beanstalk and ready to deploy the Flask app. **US East (Ohio) us-east-2** server was selected for this project.



Compute

Amazon Elastic Beanstalk

End-to-end web application management.

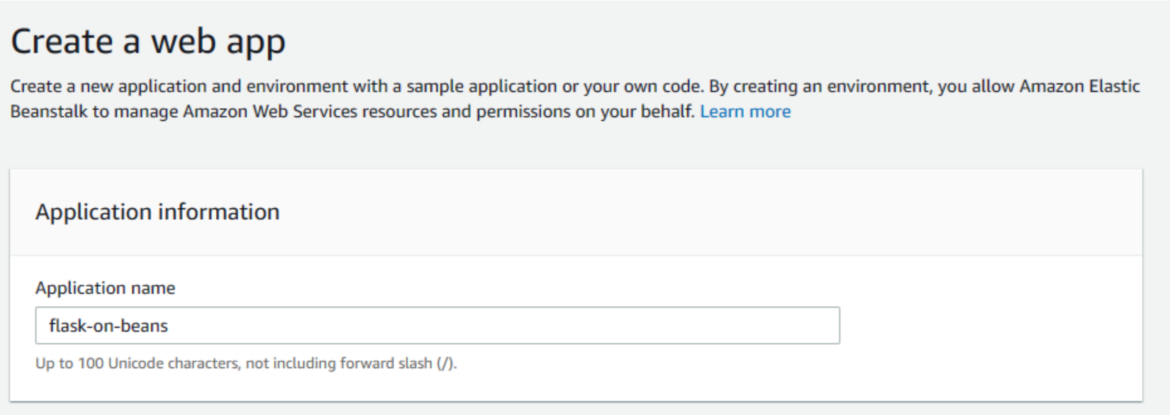
Amazon Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

Get started

Easily deploy your web application in minutes.

By selecting “Create application” on the page, it redirects us to the configure environment page.

The following settings were chosen for this deployment project:



Create a web app

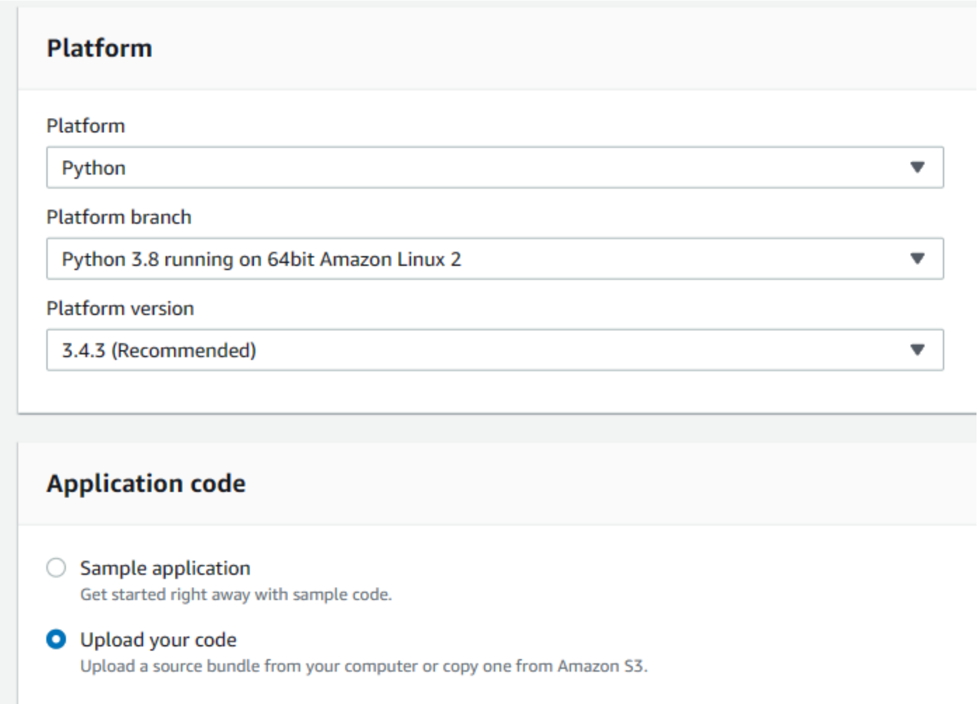
Create a new application and environment with a sample application or your own code. By creating an environment, you allow Amazon Elastic Beanstalk to manage Amazon Web Services resources and permissions on your behalf. Learn more

Application information

Application name

flask-on-beans

Up to 100 Unicode characters, not including forward slash (/).



Platform

Platform

Python ▼

Platform branch

Python 3.8 running on 64bit Amazon Linux 2 ▼

Platform version

3.4.3 (Recommended) ▼

Application code

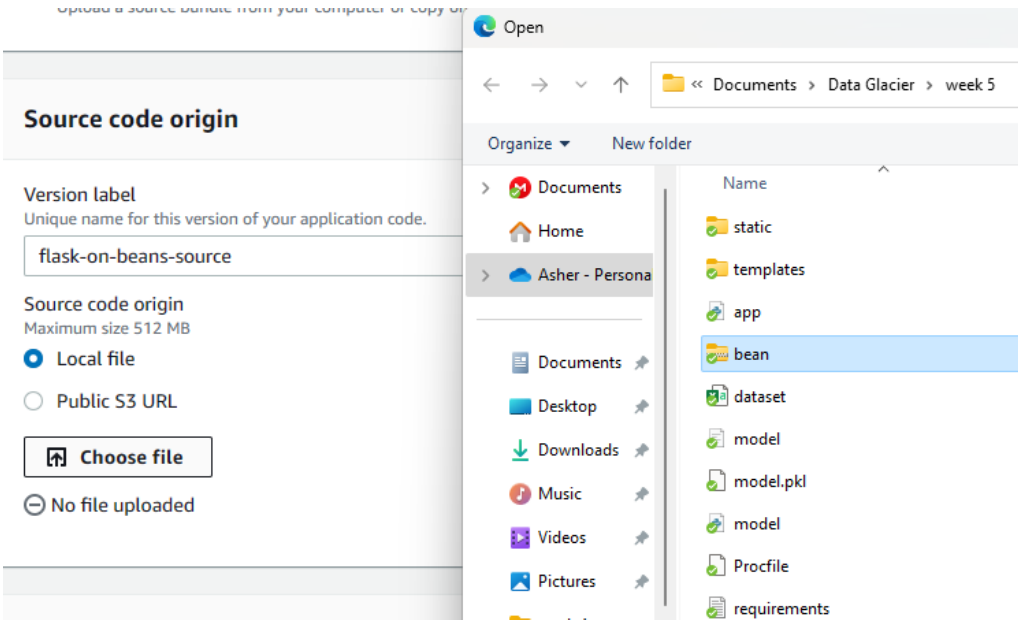
○ Sample application

Get started right away with sample code.

● Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

We then upload the local ZIP file as the source code origin. After this, we click on the “create application” button.



Source code origin

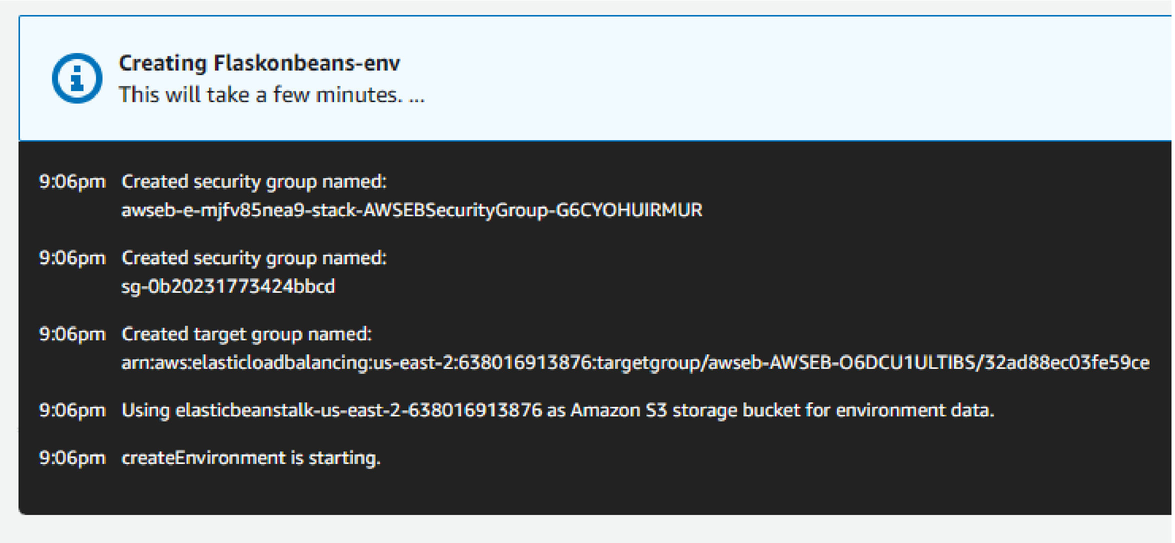
Version label
Unique name for this version of your application code.
flask-on-beans-source

Source code origin
Maximum size 512 MB
● Local file
○ Public S3 URL

Choose file

No file uploaded

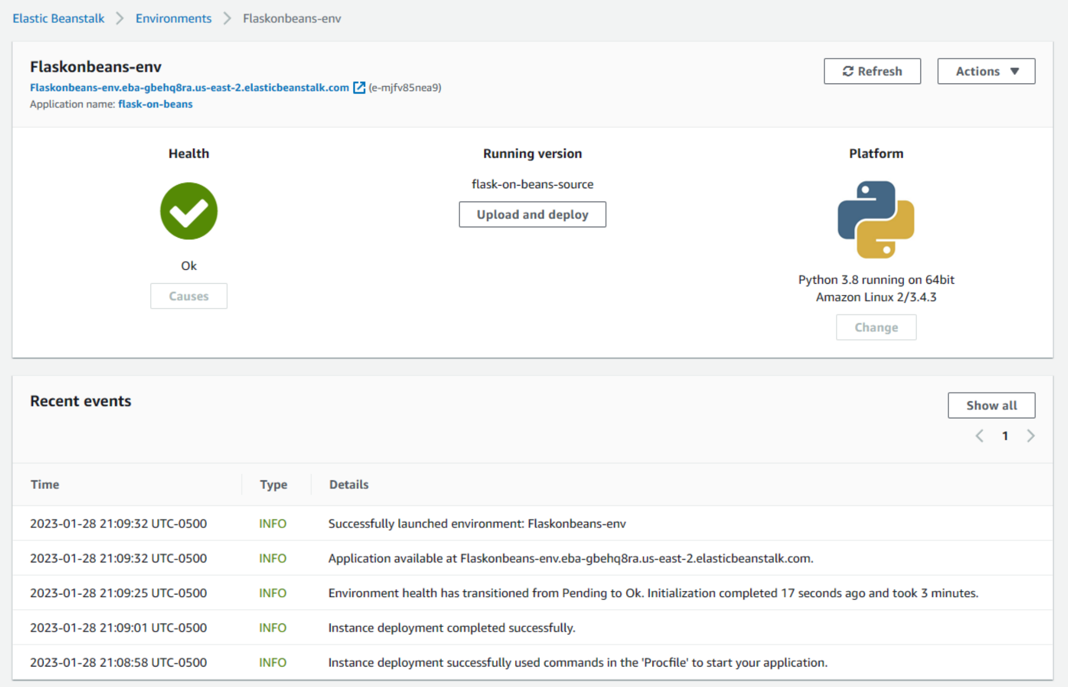
Elastic Beanstalk then creates the environment for deploying the flask app.



Creating Flaskonbeans-env
This will take a few minutes. ...

```
9:06pm  Created security group named:
        awseb-e-mjfv85nea9-stack-AWSEBSecurityGroup-G6CYOHUIRMUR

9:06pm  Created security group named:
        sg-0b20231773424bbcd

9:06pm  Created target group named:
        arn:aws:elasticloadbalancing:us-east-2:638016913876:targetgroup/awseb-AWSEB-O6DCU1ULTIBS/32ad88ec03fe59ce

9:06pm  Using elasticbeanstalk-us-east-2-638016913876 as Amazon S3 storage bucket for environment data.

9:06pm  createEnvironment is starting.
```

Finally, we can access the deployed web app with the given URL.



Flaskonbeans-env

Flaskonbeans-env.eba-gbehq8ra.us-east-2.elasticbeanstalk.com (e-mjfv85nea9)

Application name: flask-on-beans

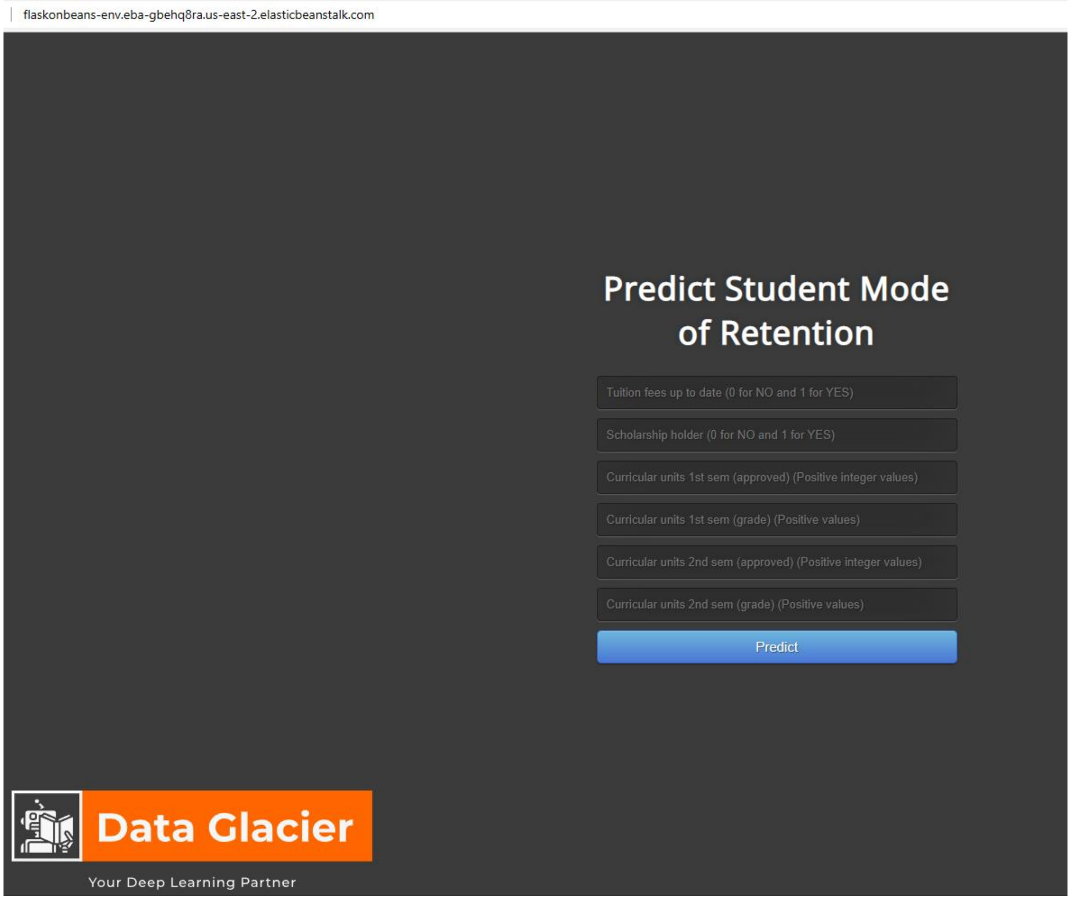
Health: Ok

Running version: flask-on-beans-source

Platform: Python 3.8 running on 64bit Amazon Linux 2/3.4.3

Recent events

Time	Type	Details
2023-01-28 21:09:32 UTC-0500	INFO	Successfully launched environment: Flaskonbeans-env
2023-01-28 21:09:32 UTC-0500	INFO	Application available at Flaskonbeans-env.eba-gbehq8ra.us-east-2.elasticbeanstalk.com.
2023-01-28 21:09:25 UTC-0500	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 17 seconds ago and took 3 minutes.
2023-01-28 21:09:01 UTC-0500	INFO	Instance deployment completed successfully.
2023-01-28 21:08:58 UTC-0500	INFO	Instance deployment successfully used commands in the 'Procfile' to start your application.



flaskonbeans-env.eba-gbehq8ra.us-east-2.elasticbeanstalk.com

Predict Student Mode of Retention

Tuition fees up to date (0 for NO and 1 for YES)

Scholarship holder (0 for NO and 1 for YES)

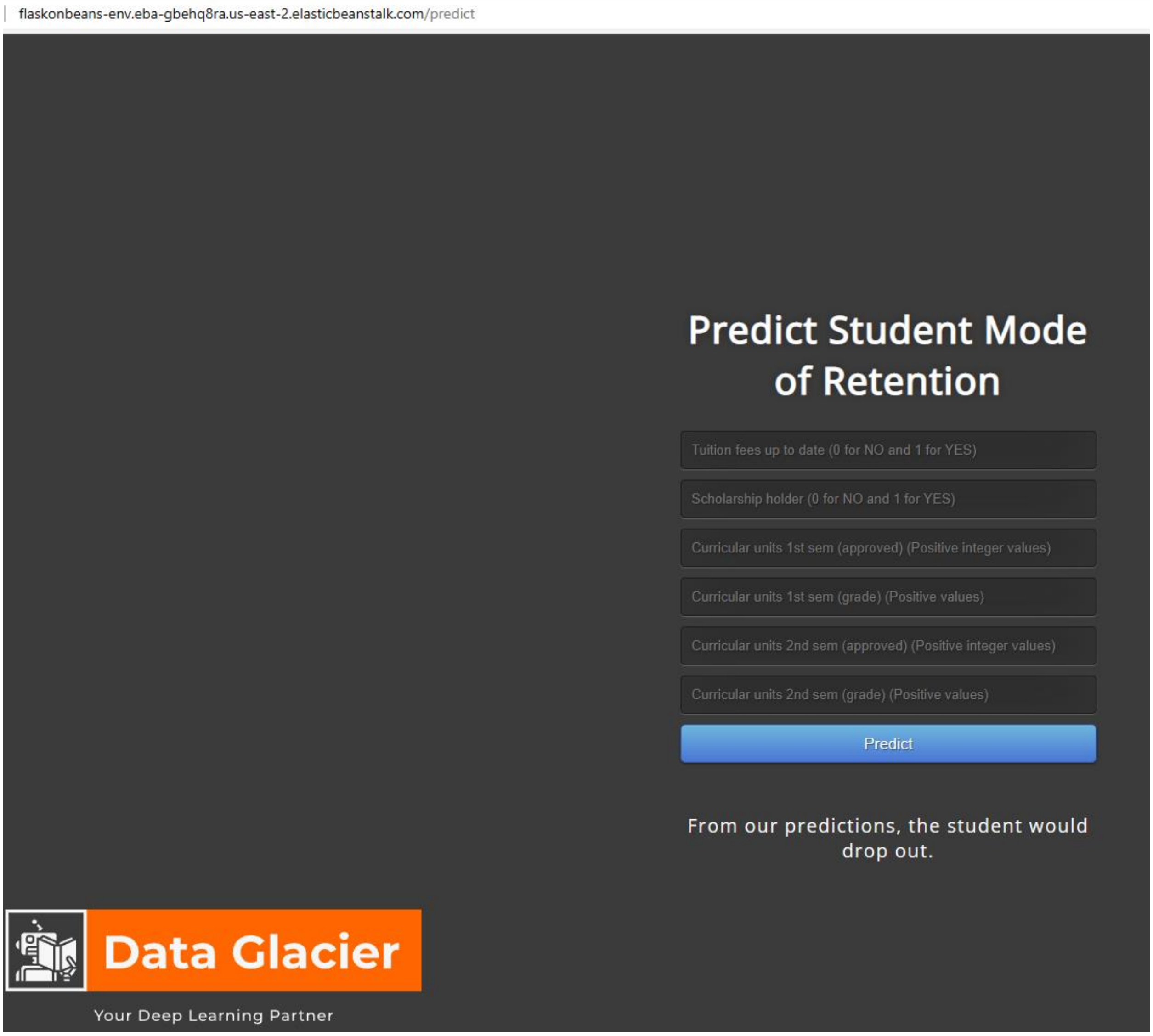
Curricular units 1st sem (approved) (Positive integer values)

Curricular units 1st sem (grade) (Positive values)

Curricular units 2nd sem (approved) (Positive integer values)

Curricular units 2nd sem (grade) (Positive values)

After several input value testing to check the web app, the deployed AWS Elastic Beanstalk app works the same way like the locally deployed Flask app does.



flaskonbeans-env.eba-gbehq8ra.us-east-2.elasticbeanstalk.com/predict

Predict Student Mode of Retention

Tuition fees up to date (0 for NO and 1 for YES)
Scholarship holder (0 for NO and 1 for YES)
Curricular units 1st sem (approved) (Positive integer values)
Curricular units 1st sem (grade) (Positive values)
Curricular units 2nd sem (approved) (Positive integer values)
Curricular units 2nd sem (grade) (Positive values)
Predict
Data Glacier
Your Deep Learning Partner

From our predictions, the student would drop out.