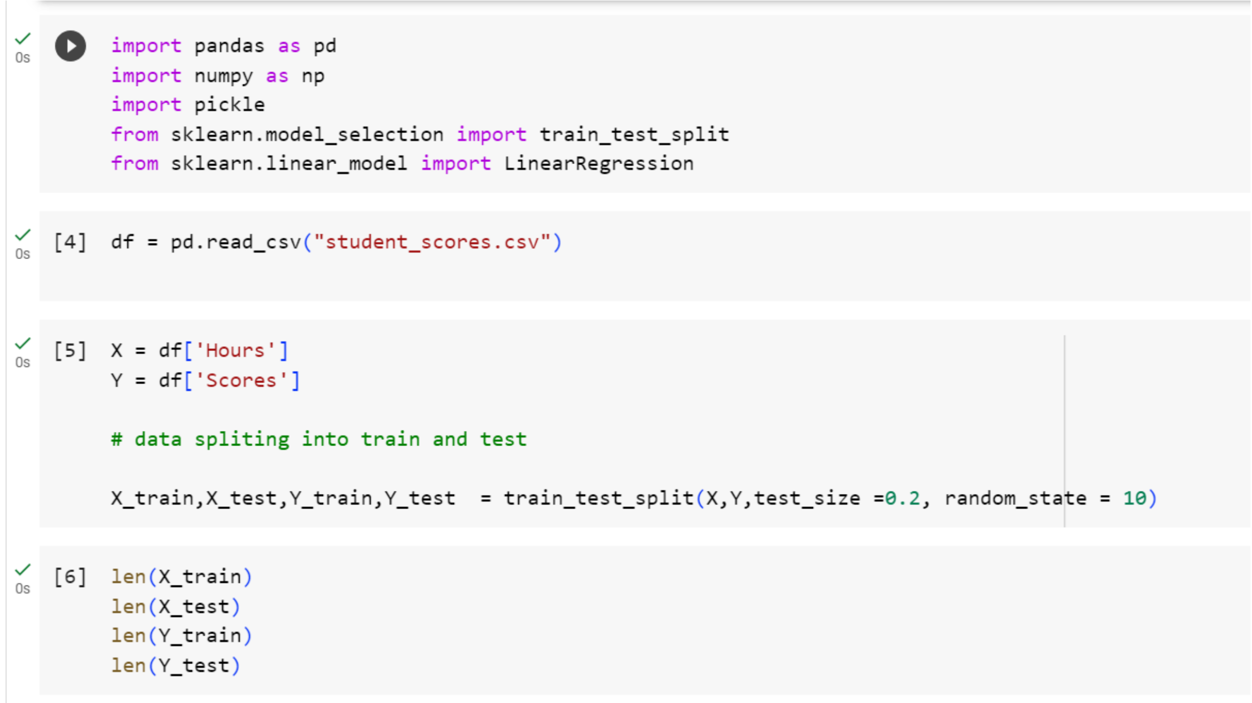
Week 5: Cloud and API deployment

Name: Anusha Asim
Batch Code: LISUM24
Submitted on: 10th September, 2023

Step 1: Training and Downloading ML Model

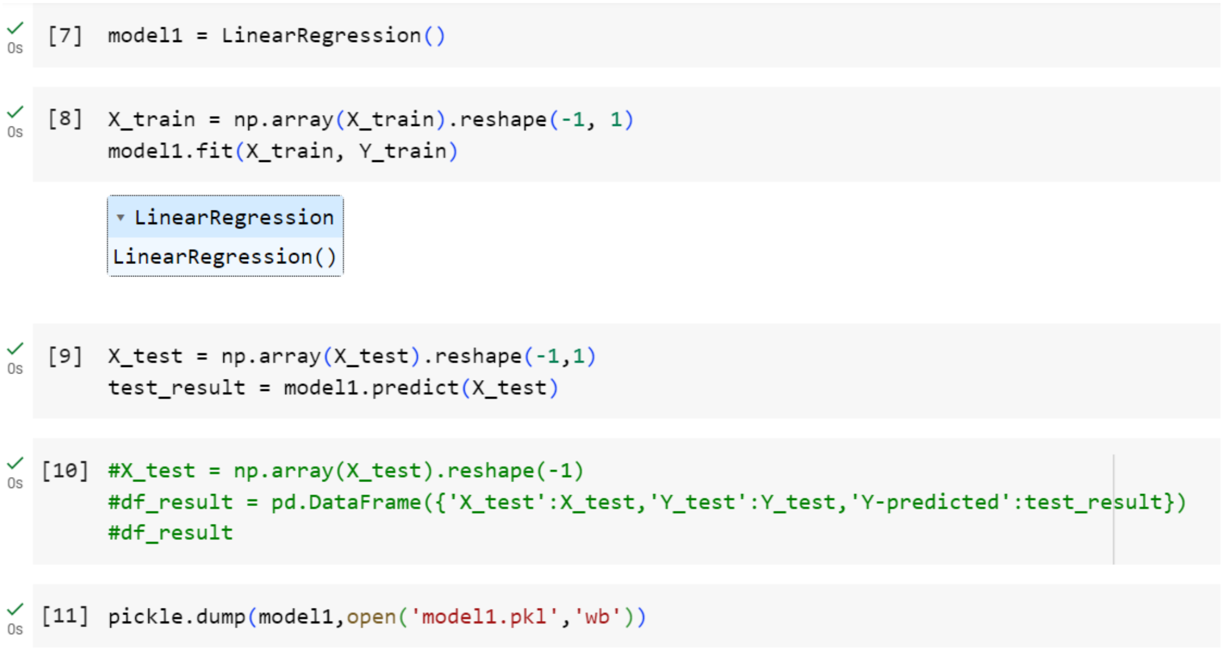


```
import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
[4] df = pd.read_csv("student_scores.csv")
```

```
[5] X = df['Hours']
    Y = df['Scores']

    # data spliting into train and test

    X_train,X_test,Y_train,Y_test  = train_test_split(X,Y,test_size =0.2, random_state = 10)
```

```
[6] len(X_train)
    len(X_test)
    len(Y_train)
    len(Y_test)
```



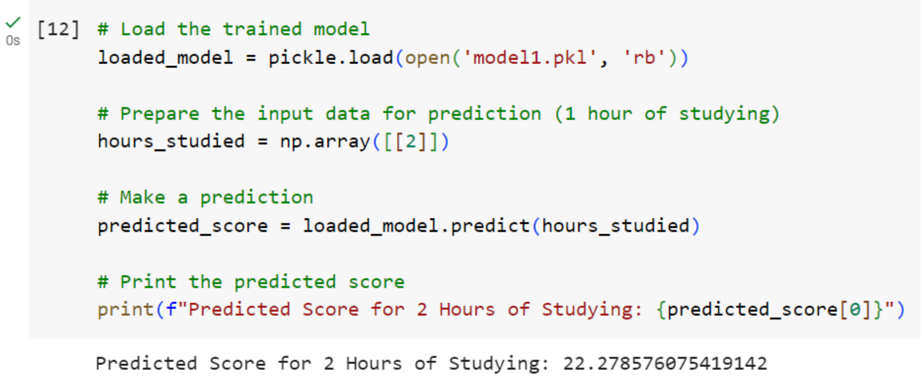
```
[7] model1 = LinearRegression()
```

```
[8] X_train = np.array(X_train).reshape(-1, 1)
    model1.fit(X_train, Y_train)

    ▾ LinearRegression
    LinearRegression()
```

```
[9] X_test = np.array(X_test).reshape(-1,1)
    test_result = model1.predict(X_test)
```

```
[10] #X_test = np.array(X_test).reshape(-1)
     #df_result = pd.DataFrame({'X_test':X_test,'Y_test':Y_test,'Y-predicted':test_result})
     #df_result
```

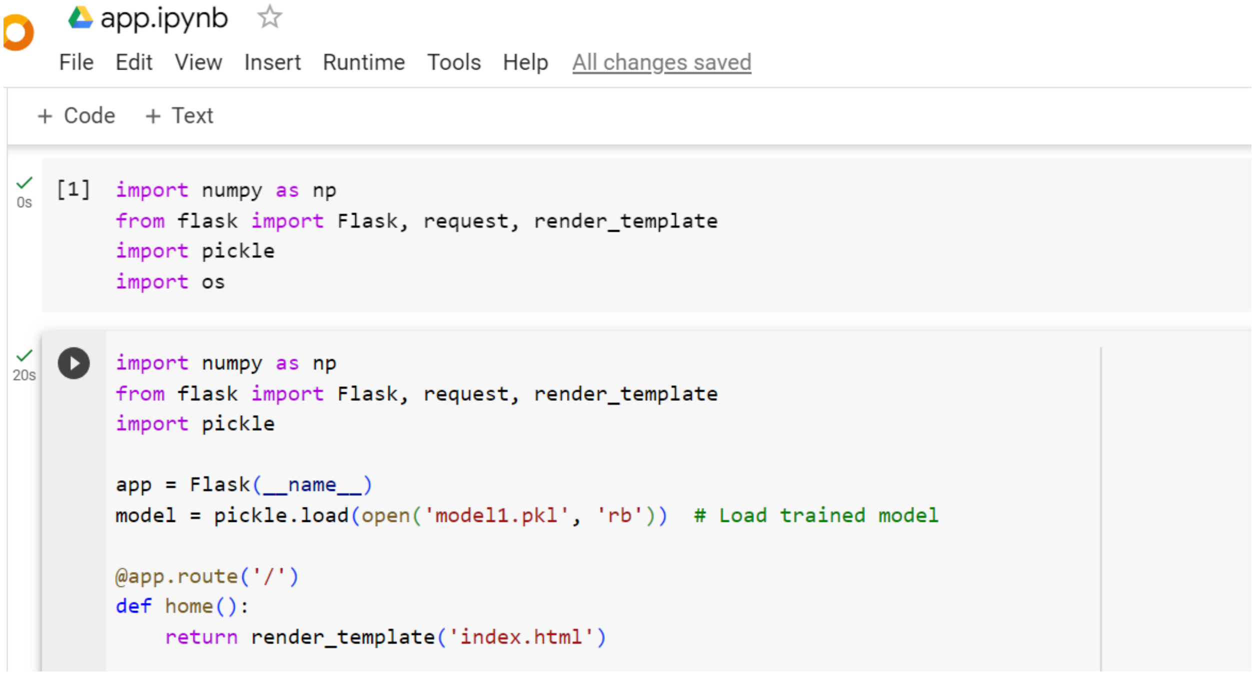
```
[11] pickle.dump(model1,open('model1.pkl','wb'))
```



```
# Load the trained model
loaded_model = pickle.load(open('model1.pkl', 'rb'))

# Prepare the input data for prediction (1 hour of studying)
hours_studied = np.array([[2]])

# Make a prediction
predicted_score = loaded_model.predict(hours_studied)

# Print the predicted score
print(f"Predicted Score for 2 Hours of Studying: {predicted_score[0]}")
```

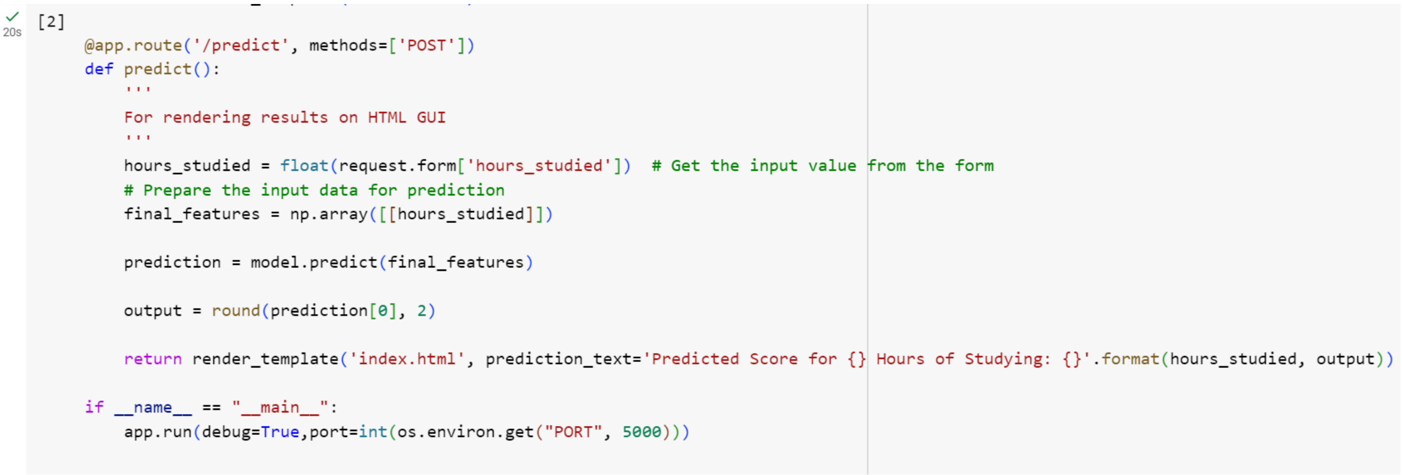
```
Predicted Score for 2 Hours of Studying: 22.278576075419142
```

Step 2: Creating Web App Using Flask and HTML

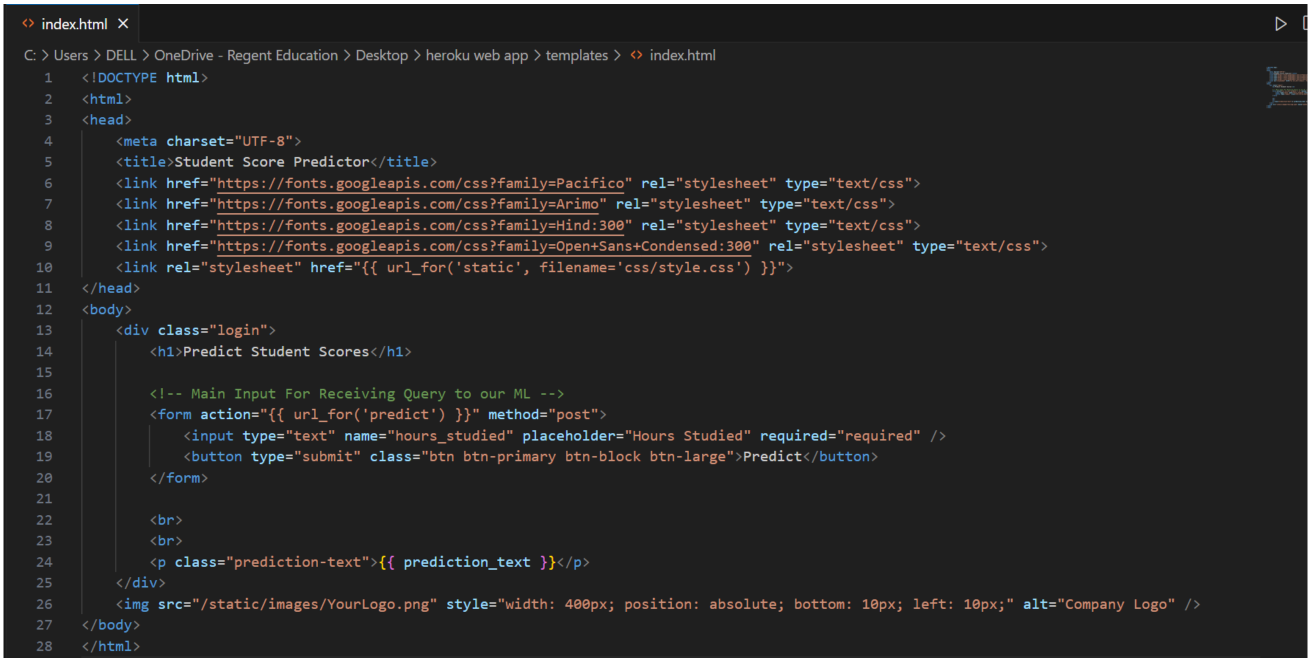


```
import numpy as np
from flask import Flask, request, render_template
import pickle
import os
```

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

app = Flask(__name__)
model = pickle.load(open('model1.pkl', 'rb'))  # Load trained model

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

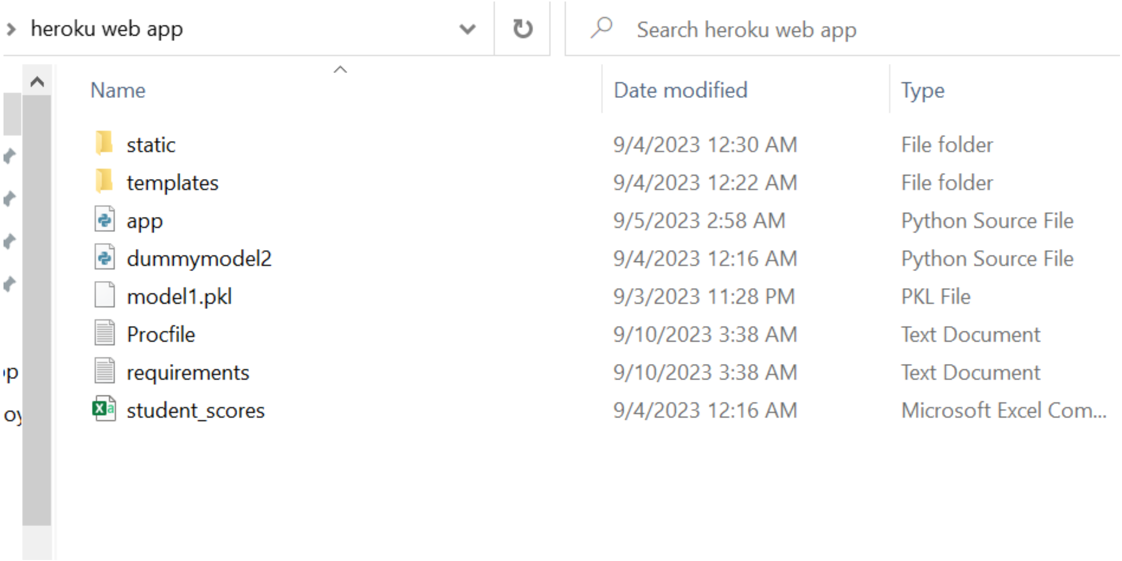


```
@app.route('/predict', methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    hours_studied = float(request.form['hours_studied'])  # Get the input value from the form
    # Prepare the input data for prediction
    final_features = np.array([[hours_studied]])

    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Predicted Score for {} Hours of Studying: {}'.format(hours_studied, output))

if __name__ == "__main__":
    app.run(debug=True,port=int(os.environ.get("PORT", 5000)))
```

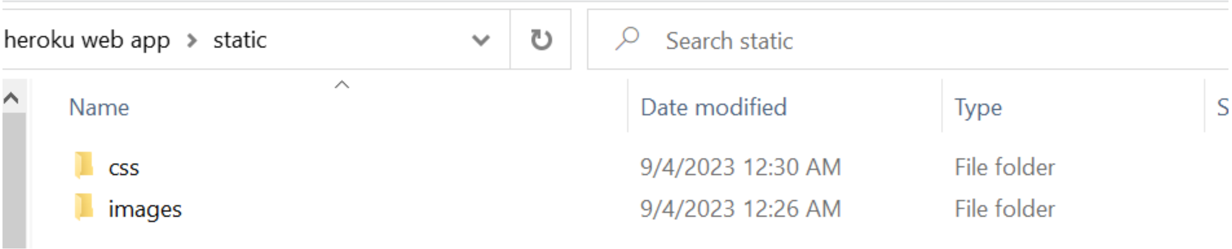

index.html
C: > Users > DELL > OneDrive - Regent Education > Desktop > heroku web app > templates > index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Student Score Predictor</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 Scores</h1>

        <!-- Main Input For Receiving Query to our ML -->
        <form action="{{ url_for('predict') }}" method="post">
            <input type="text" name="hours_studied" placeholder="Hours Studied" required="required" />
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
        </form>

        <br>
        <br>
        <p class="prediction-text">{{ prediction_text }}</p>
    </div>
    <img src="/static/images/YourLogo.png" style="width: 400px; position: absolute; bottom: 10px; left: 10px;" alt="Company Logo" />
</body>
</html>
```

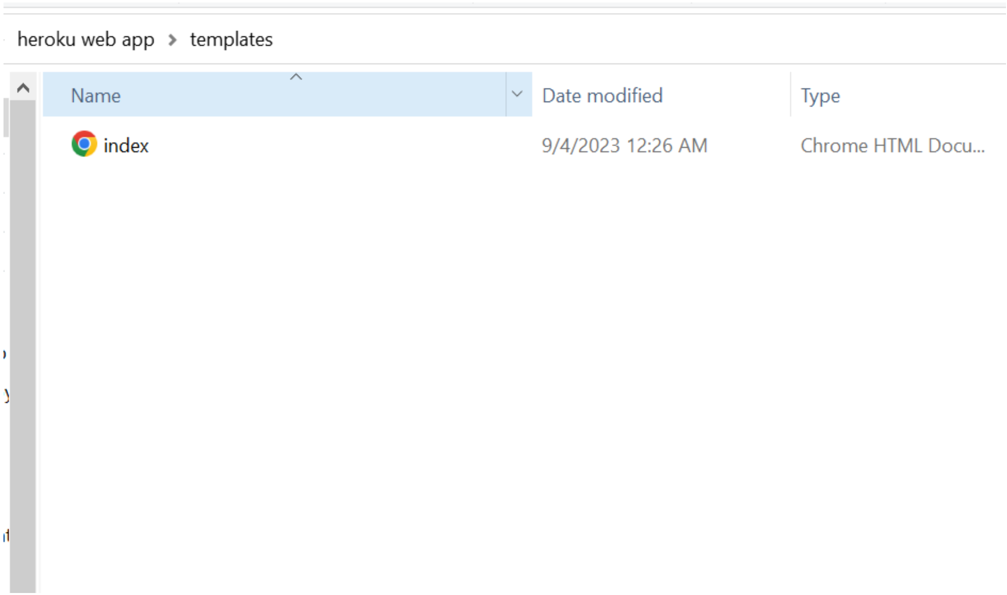
Saving Files in Folder:



Name	Date modified	Type
static	9/4/2023 12:30 AM	File folder
templates	9/4/2023 12:22 AM	File folder
app	9/5/2023 2:58 AM	Python Source File
dummymodel2	9/4/2023 12:16 AM	Python Source File
model1.pkl	9/3/2023 11:28 PM	PKL File
Procfile	9/10/2023 3:38 AM	Text Document
requirements	9/10/2023 3:38 AM	Text Document
student_scores	9/4/2023 12:16 AM	Microsoft Excel Com...



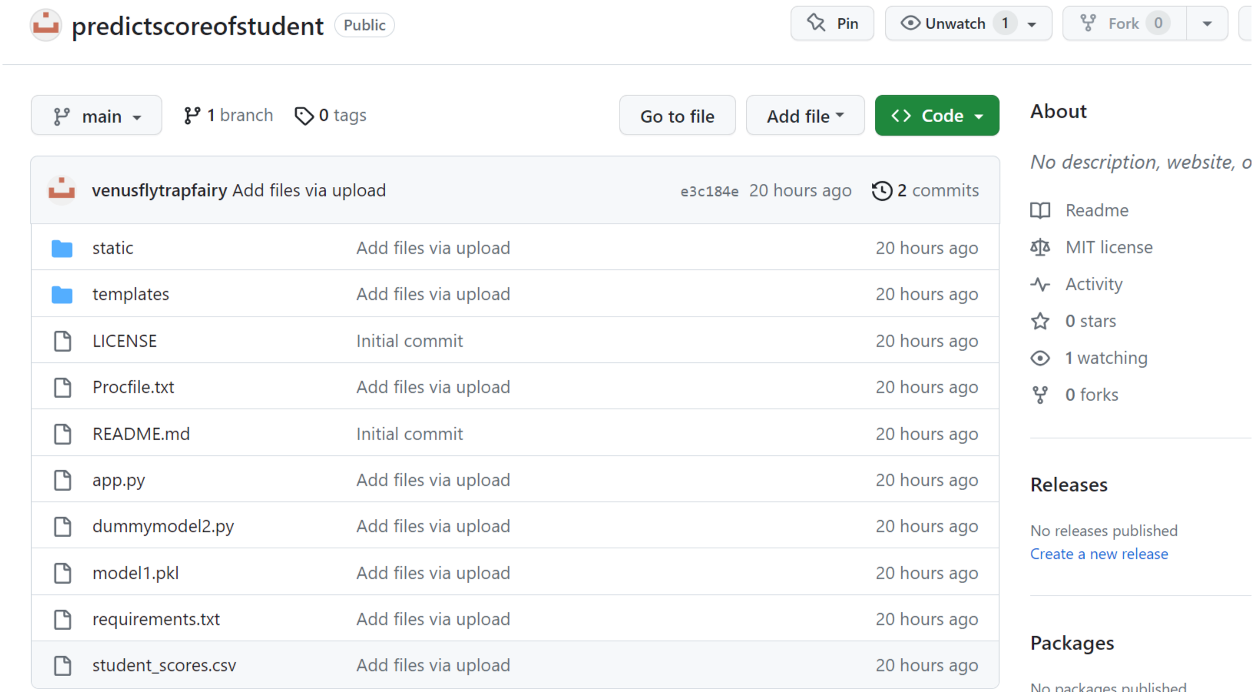
Name	Date modified	Type	S
css	9/4/2023 12:30 AM	File folder	
images	9/4/2023 12:26 AM	File folder	



heroku web app > templates

Name	Date modified	Type
index	9/4/2023 12:26 AM	Chrome HTML Docu...

Step 3: Committing Code in GitHub Online Repository



predictscoreofstudent Public

venusflytrapfairy Add files via upload e3c184e 20 hours ago 2 commits

static	Add files via upload	20 hours ago
templates	Add files via upload	20 hours ago
LICENSE	Initial commit	20 hours ago
Procfile.txt	Add files via upload	20 hours ago
README.md	Initial commit	20 hours ago
app.py	Add files via upload	20 hours ago
dummymodel2.py	Add files via upload	20 hours ago
model1.pkl	Add files via upload	20 hours ago
requirements.txt	Add files via upload	20 hours ago
student_scores.csv	Add files via upload	20 hours ago

About

No description, website, o

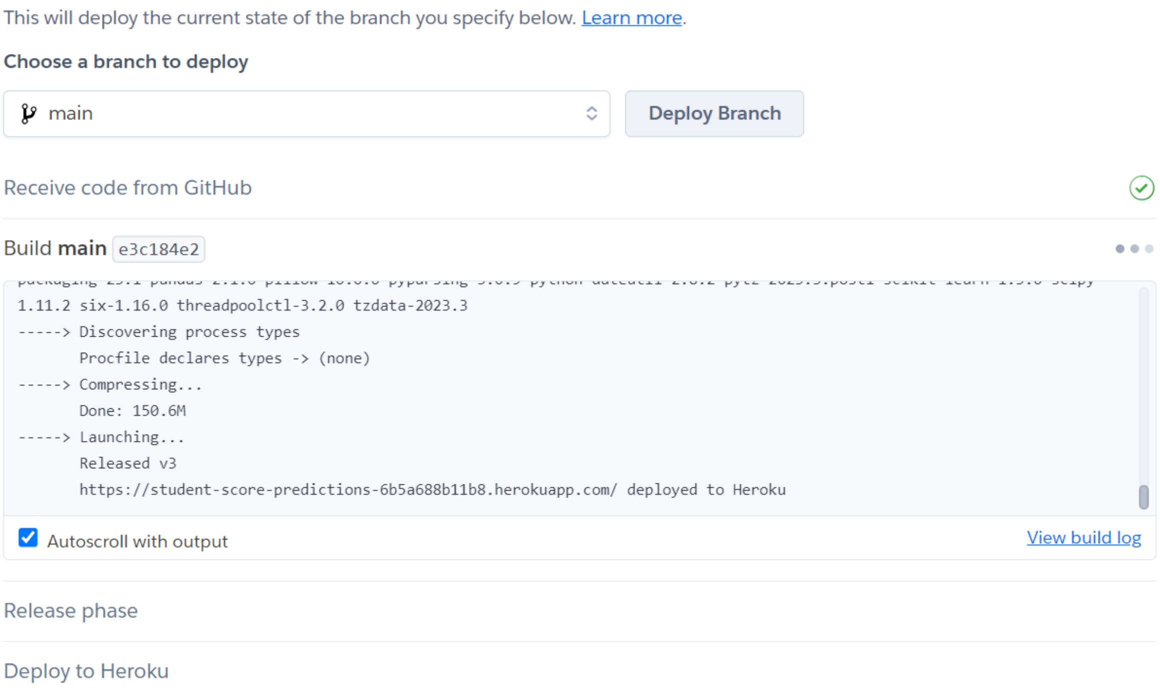
Releases

No releases published
Create a new release

Packages

No packages published

Step 4: Deployment of ML Model



This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

Receive code from GitHub ✓

Build **main** `e3c184e2`

```
packaging-23.1 pandas-2.1.0 pillow-10.0.0 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2023.3.post1 scikit-learn-1.3.0 scipy
1.11.2 six-1.16.0 threadpoolctl-3.2.0 tzdata-2023.3
-----> Discovering process types
       Procfile declares types -> (none)
-----> Compressing...
       Done: 150.6M
-----> Launching...
       Released v3
       https://student-score-predictions-6b5a688b11b8.herokuapp.com/ deployed to Heroku
```

☑ Autoscroll with output

Release phase

Deploy to Heroku