

Name: Payal Upadhyay

Batch Code: LISUM101

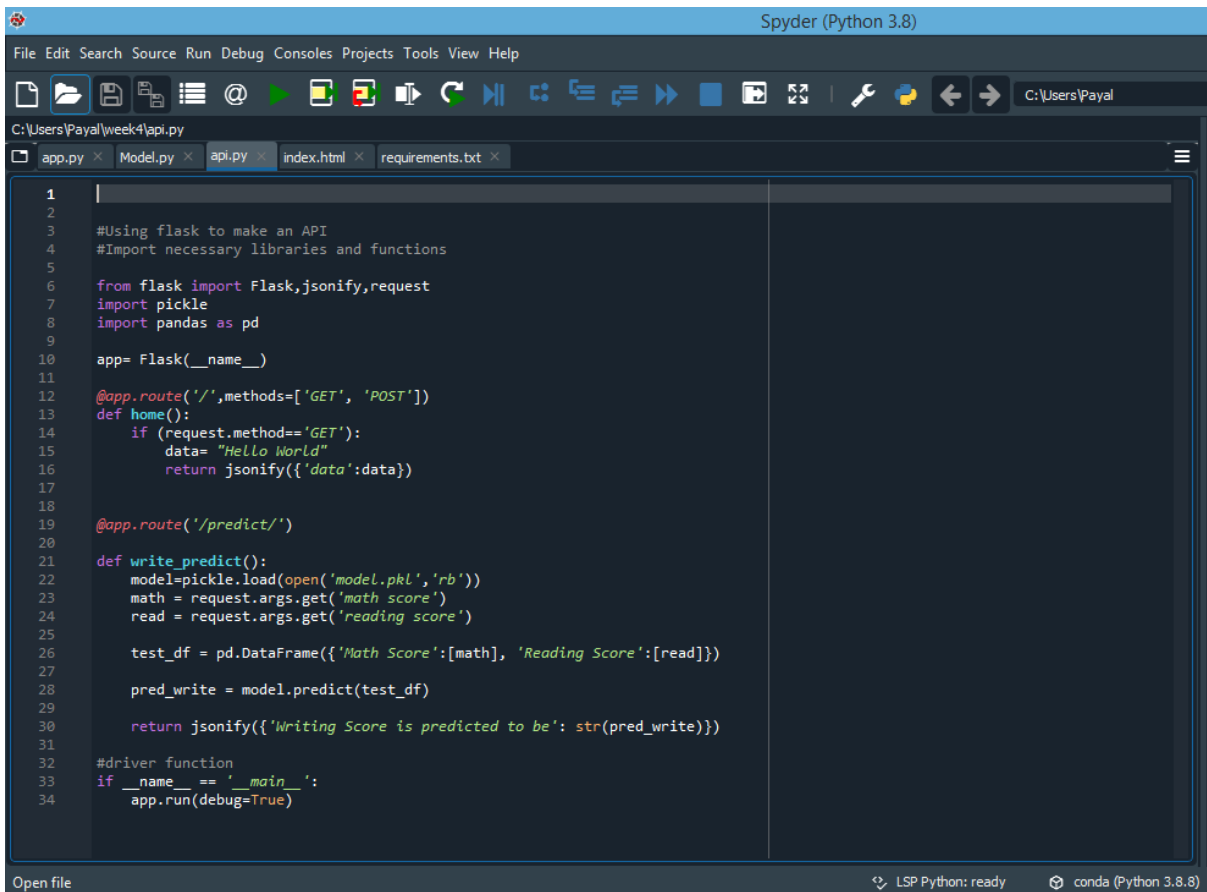
Submission Date: 2021/07/08

Submitted to: Data Glacier

API based deployment (on Postman)

Aim: Predict the writing score based on math score and reading score

api.py



```
1 |
2 |
3 | #Using flask to make an API
4 | #Import necessary libraries and functions
5 |
6 | from flask import Flask, jsonify, request
7 | import pickle
8 | import pandas as pd
9 |
10 | app = Flask(__name__)
11 |
12 | @app.route('/', methods=['GET', 'POST'])
13 | def home():
14 |     if (request.method == 'GET'):
15 |         data = "Hello World"
16 |         return jsonify({'data': data})
17 |
18 |
19 | @app.route('/predict/')
20 |
21 | def write_predict():
22 |     model = pickle.load(open('model.pkl', 'rb'))
23 |     math = request.args.get('math score')
24 |     read = request.args.get('reading score')
25 |
26 |     test_df = pd.DataFrame({'Math Score': [math], 'Reading Score': [read]})
27 |
28 |     pred_write = model.predict(test_df)
29 |
30 |     return jsonify({'Writing Score is predicted to be': str(pred_write)})
31 |
32 | #driver function
33 | if __name__ == '__main__':
34 |     app.run(debug=True)
```

Open file LSP Python: ready conda (Python 3.8.8)

- Run the api.py file to ensure no errors are encountered

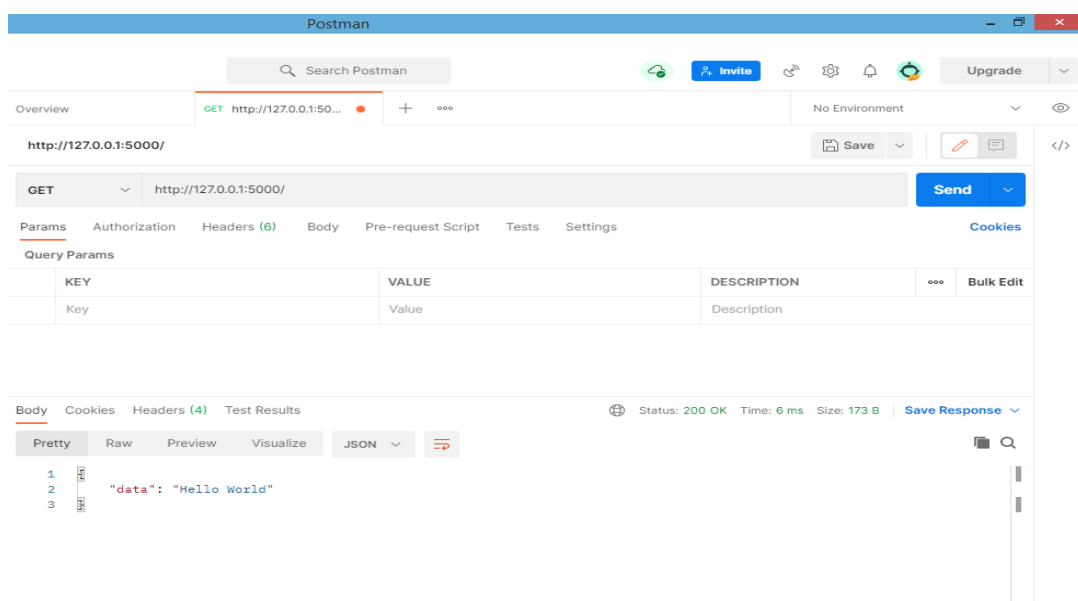
```
C:\Windows\System32\cmd.exe - python api.py
Successfully installed docopt-0.6.2 pipreqs-0.4.10 yarg-0.1.9
WARNING: You are using pip version 21.1.1; however, version 21.1.3 is available.
You should consider upgrading via the 'c:\users\payal\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\Payal\week4>pipreqs C:\Users\Payal\week4\
WARNING: Requirements.txt already exists, use --force to overwrite it

C:\Users\Payal\week4>pipreqs C:\Users\Payal\week4\
INFO: Successfully saved requirements file in C:\Users\Payal\week4\requirements.txt

C:\Users\Payal\week4>python api.py
* Serving Flask app 'api' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 668-404-510
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Install Postman
- Create an account and Log in
- Click 'Create Request'
- Enter the URL from the command prompt and test



- Enter 'predict\' to the url
- Enter parameters
- Click send to see output

The screenshot shows the Postman interface with a GET request configured. The URL is `http://127.0.0.1:5000/predict?math score=60&reading score=70`. The request is sent, and the response is displayed in the 'Body' tab, showing a JSON object with a predicted writing score.

Request Details:

- Method: GET
- URL: `http://127.0.0.1:5000/predict?math score=60&reading score=70`
- Query Params:

KEY	VALUE	DESCRIPTION
math score	60	
reading score	70	

Response Details:

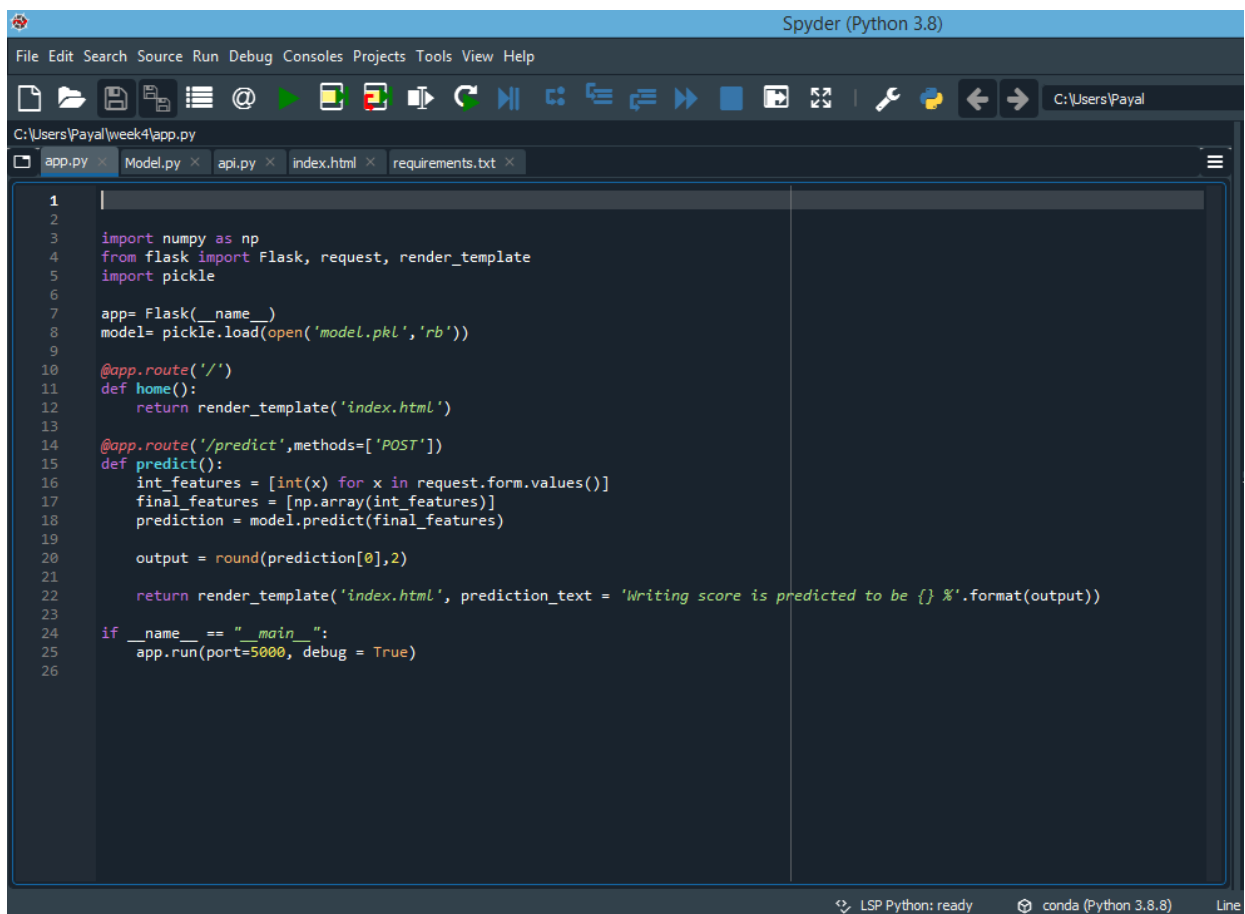
- Status: 200 OK
- Time: 9.50 s
- Size: 203 B
- Body (Pretty):


```

1 {
2   "Writing Score is predicted to be": "[68.63418447]"
3 }
```

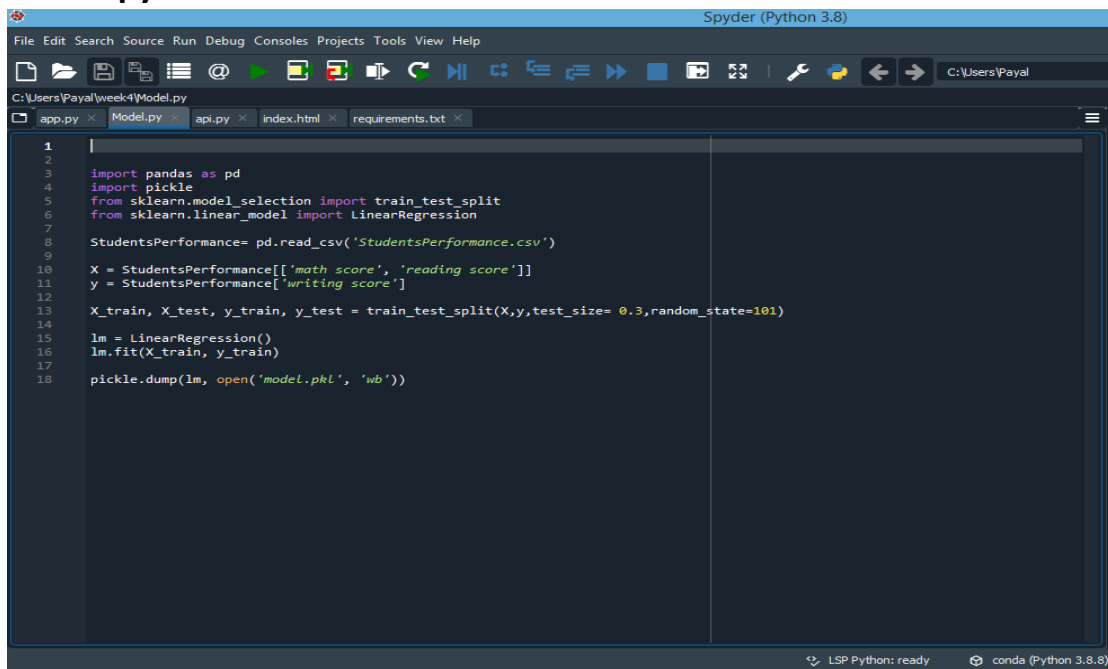
Cloud based deployment (Heroku)

app.py

The image shows a screenshot of the Spyder Python IDE interface. The title bar at the top reads "Spyder (Python 3.8)". Below the title bar is a menu bar with options: File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Underneath the menu bar is a toolbar with various icons for file operations, running, and debugging. The main window displays a file explorer on the left with a tree view showing the project structure: C:\Users\Payal\week4\app.py, app.py, Model.py, api.py, index.html, and requirements.txt. The app.py file is selected and its code is displayed in the main editor. The code is a Flask application that loads a pickle model and has two routes: a home route and a predict route. The predict route takes a POST request, processes the data, and returns a prediction. The status bar at the bottom indicates "LSP Python: ready" and "conda (Python 3.8.8)".

```
1  
2  
3 import numpy as np  
4 from flask import Flask, request, render_template  
5 import pickle  
6  
7 app= Flask(__name__)  
8 model= pickle.load(open('model.pkl','rb'))  
9  
10 @app.route('/')  
11 def home():  
12     return render_template('index.html')  
13  
14 @app.route('/predict',methods=['POST'])  
15 def predict():  
16     int_features = [int(x) for x in request.form.values()]  
17     final_features = [np.array(int_features)]  
18     prediction = model.predict(final_features)  
19  
20     output = round(prediction[0],2)  
21  
22     return render_template('index.html', prediction_text = 'Writing score is predicted to be {}'.format(output))  
23  
24 if __name__ == "__main__":  
25     app.run(port=5000, debug = True)  
26
```

Model.py

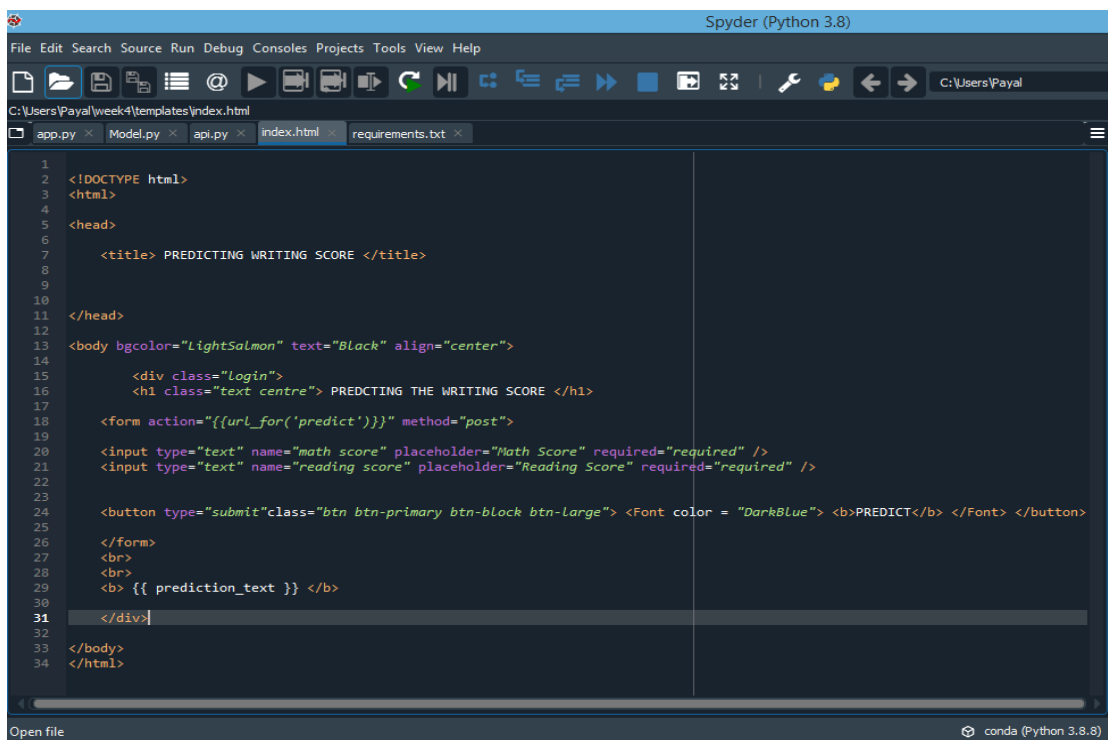


The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations, running, and debugging. The file explorer on the left shows the project structure with files: app.py, Model.py, api.py, index.html, and requirements.txt. The main editor displays the code for Model.py, which imports pandas, pickle, sklearn, and LinearRegression. It reads a CSV file, splits the data, trains a LinearRegression model, and saves it as model.pkl.

```
1 |
2 |
3 | import pandas as pd
4 | import pickle
5 | from sklearn.model_selection import train_test_split
6 | from sklearn.linear_model import LinearRegression
7 |
8 | StudentsPerformance= pd.read_csv('StudentsPerformance.csv')
9 |
10 | X = StudentsPerformance[['math score', 'reading score']]
11 | y = StudentsPerformance['writing score']
12 |
13 | X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.3,random_state=101)
14 |
15 | lm = LinearRegression()
16 | lm.fit(X_train, y_train)
17 |
18 | pickle.dump(lm, open('model.pkl', 'wb'))
```

The status bar at the bottom indicates 'LSP Python: ready' and 'conda (Python 3.8.8)'.

index.html

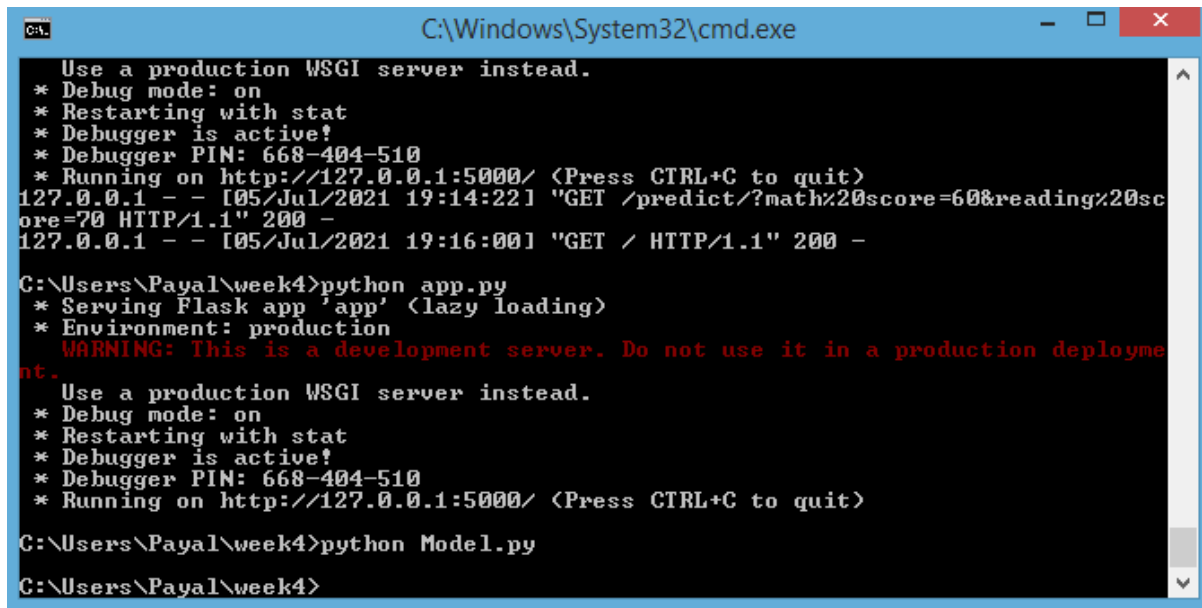


The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations, running, and debugging. The file explorer on the left shows the project structure with files: app.py, Model.py, api.py, index.html, and requirements.txt. The main editor displays the code for index.html, which is an HTML page for predicting writing scores. It includes a title, a form with input fields for math and reading scores, and a submit button.

```
1 |
2 | <!DOCTYPE html>
3 | <html>
4 |
5 | <head>
6 |
7 |     <title> PREDICTING WRITING SCORE </title>
8 |
9 |
10 |
11 | </head>
12 |
13 | <body bgcolor="LightSalmon" text="Black" align="center">
14 |
15 |     <div class="login">
16 |         <h1 class="text centre"> PREDICTING THE WRITING SCORE </h1>
17 |
18 |         <form action="{{url_for('predict')}}" method="post">
19 |
20 |             <input type="text" name="math score" placeholder="Math Score" required="required" />
21 |             <input type="text" name="reading score" placeholder="Reading Score" required="required" />
22 |
23 |             <button type="submit" class="btn btn-primary btn-block btn-large"> <Font color = "DarkBlue"> <b>PREDICT</b> </Font> </button>
24 |
25 |         </form>
26 |         <br>
27 |         <br>
28 |         <b> {{ prediction_text }} </b>
29 |
30 |     </div>
31 |
32 |
33 | </body>
34 | </html>
```

The status bar at the bottom indicates 'conda (Python 3.8.8)'.

- Run app.py and Model.py to make sure no errors are encountered



```
C:\Windows\System32\cmd.exe

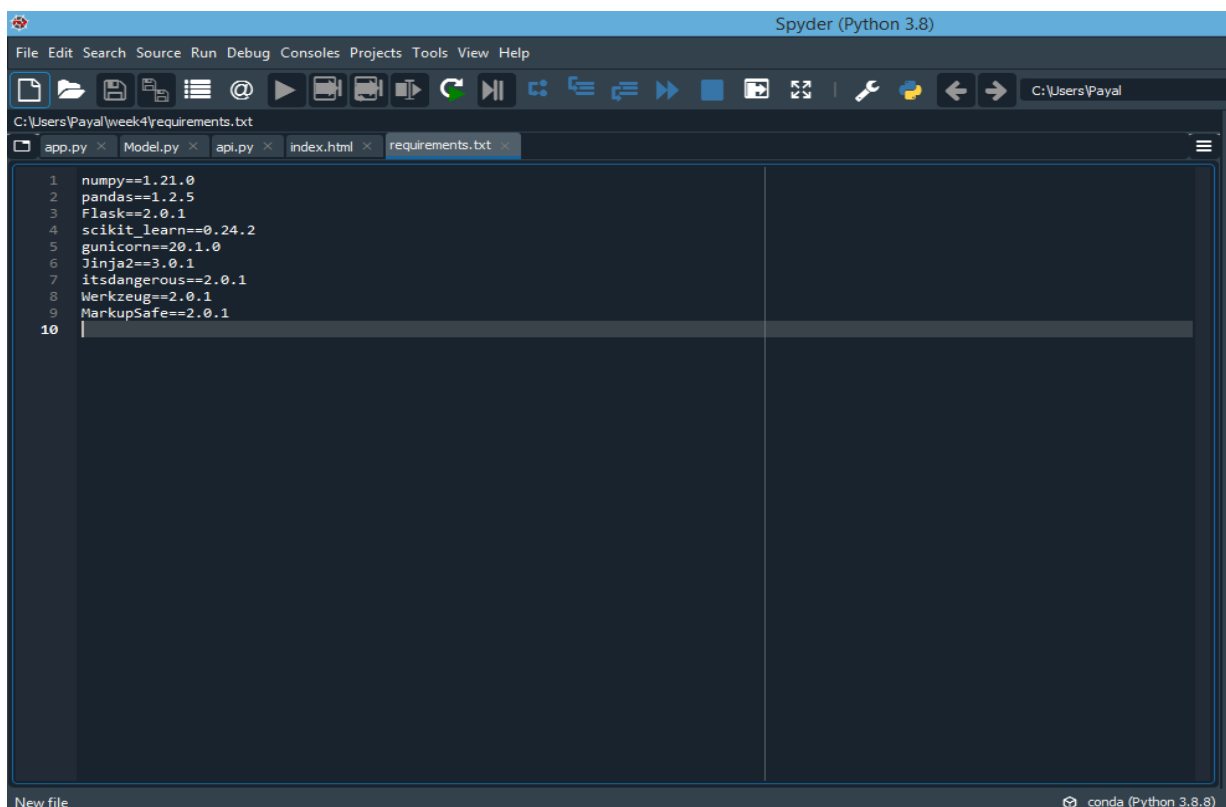
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 668-404-510
* Running on http://127.0.0.1:5000/ <Press CTRL+C to quit>
127.0.0.1 - - [05/Jul/2021 19:14:22] "GET /predict/?math%20score=60&reading%20score=70 HTTP/1.1" 200 -
127.0.0.1 - - [05/Jul/2021 19:16:00] "GET / HTTP/1.1" 200 -

C:\Users\Payal\week4>python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 668-404-510
* Running on http://127.0.0.1:5000/ <Press CTRL+C to quit>

C:\Users\Payal\week4>python Model.py

C:\Users\Payal\week4>
```

requirements.txt



```
Spyder (Python 3.8)

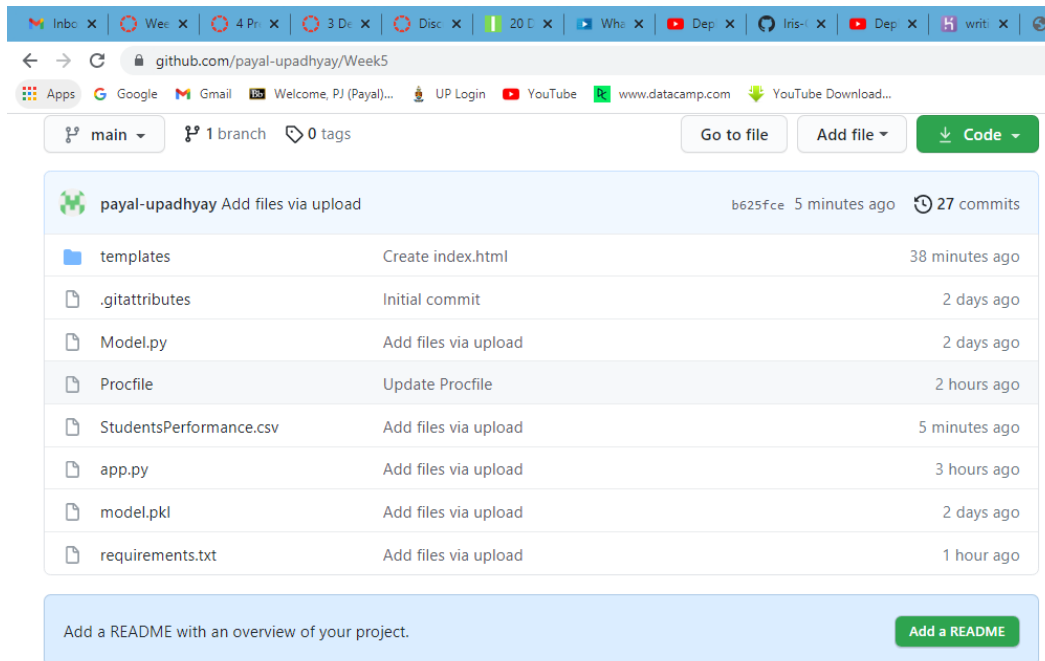
File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Payal\week4\requirements.txt

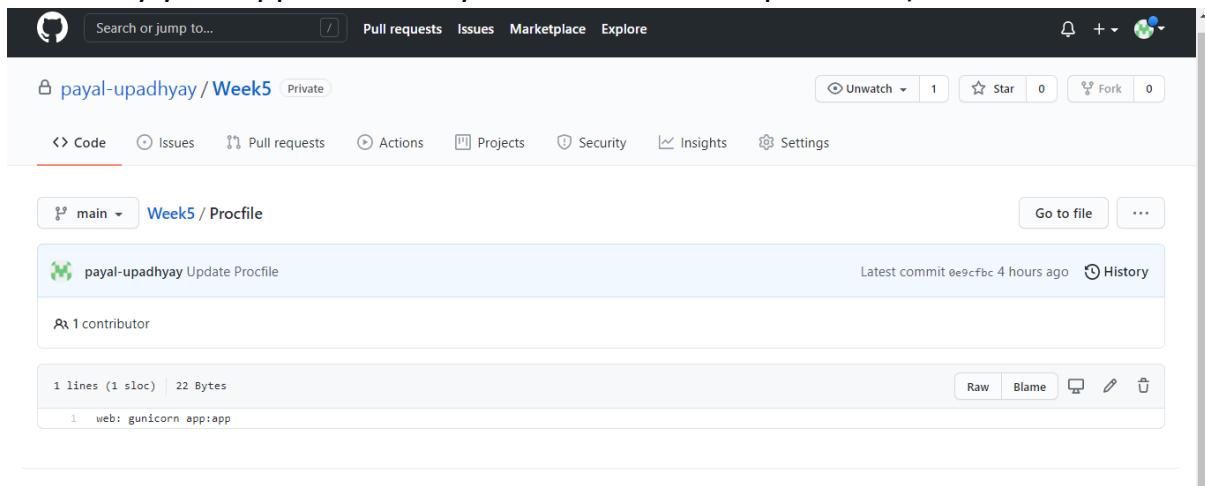
app.py x Model.py x api.py x index.html x requirements.txt x

1 numpy==1.21.0
2 pandas==1.2.5
3 Flask==2.0.1
4 scikit_learn==0.24.2
5 gunicorn==20.1.0
6 Jinja2==3.0.1
7 itsdangerous==2.0.1
8 Werkzeug==2.0.1
9 MarkupSafe==2.0.1
10
```

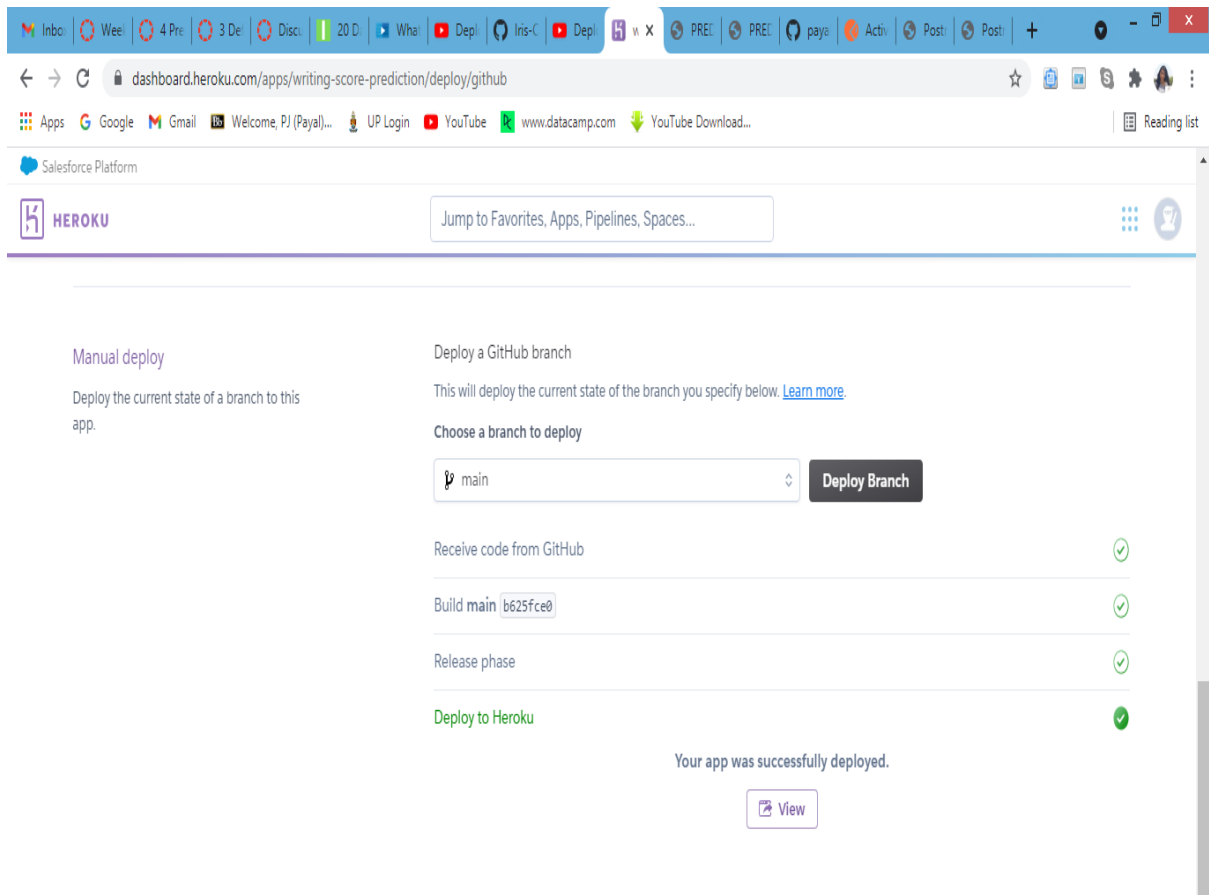
- Upload all the files to GitHub repository
- index.html file must be contained in the templates folder



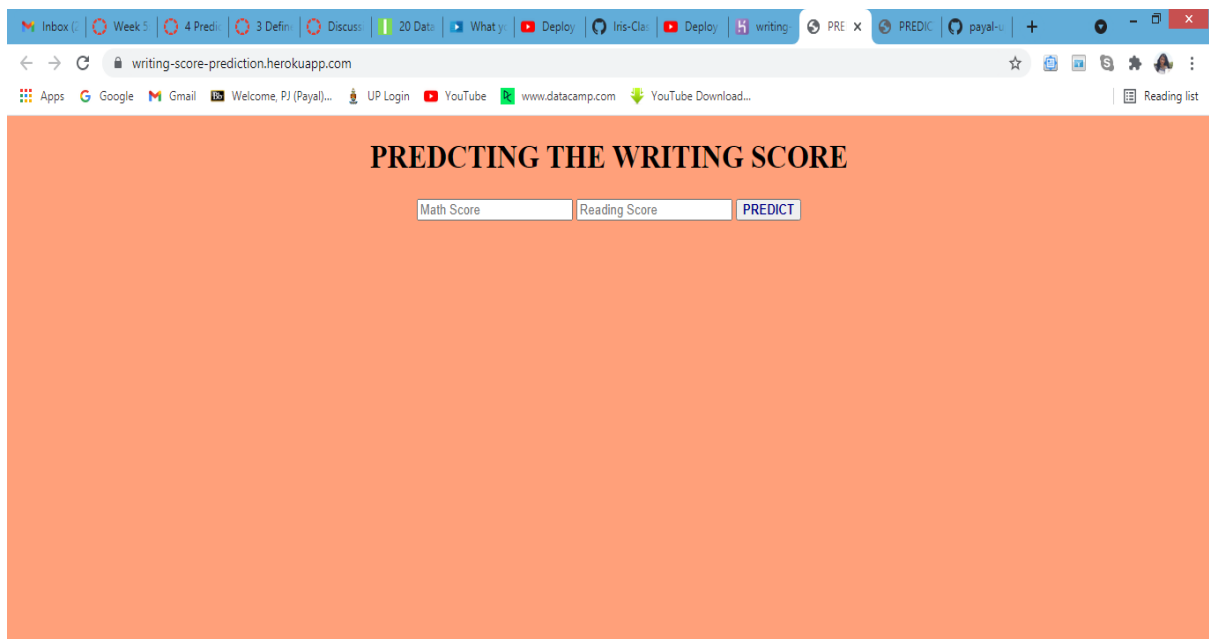
- Create a **Procfile** (Procfile is a mechanism for declaring what commands are run by your application's dynos on the Heroku platform.)



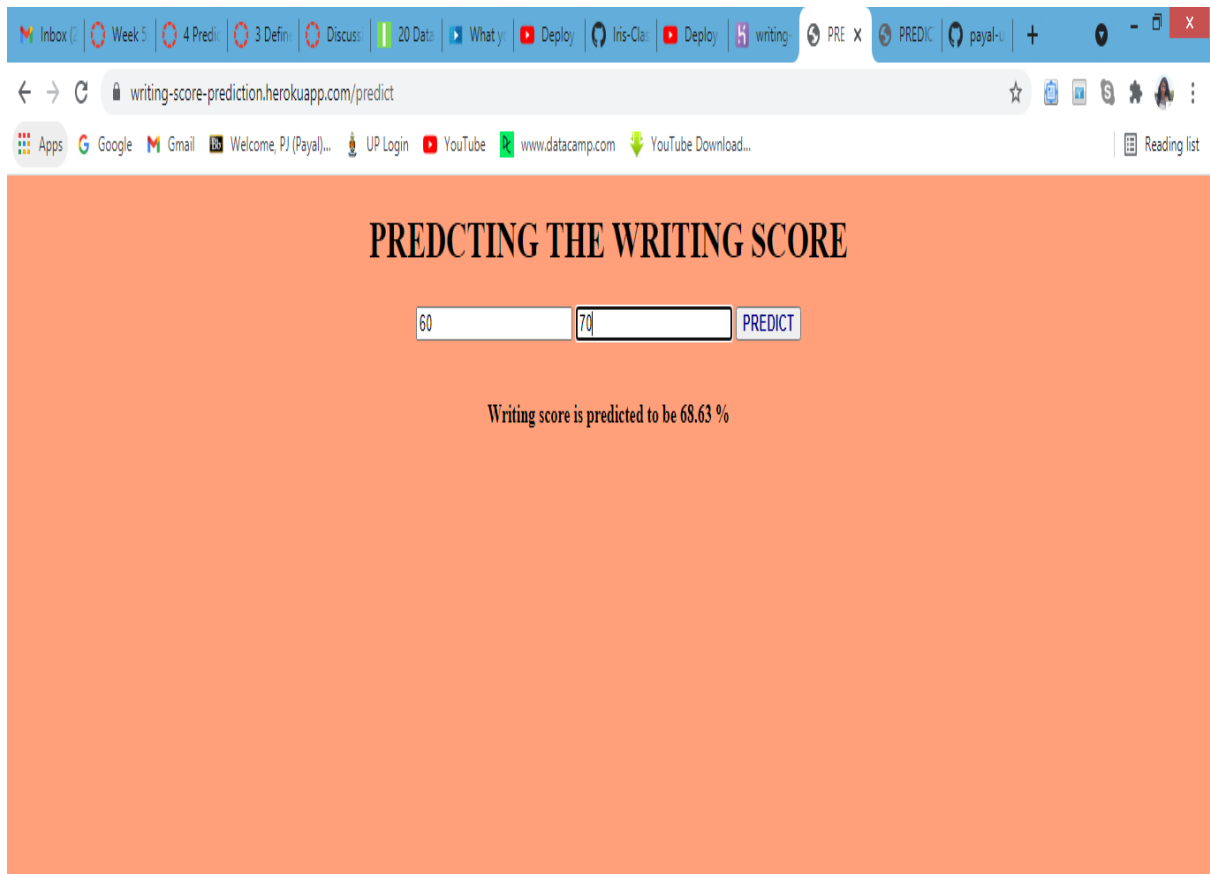
- Create and log into Heroku account
- Select 'create new app' and name your app
- Enter repository name and connect GitHub repository to Heroku
- Click 'Deploy Branch' in order to deploy



- After successful deployment click on view to view the webpage



- Test the webpage by inputting parameters



View on : <https://writing-score-prediction.herokuapp.com/>