



Data Glacier

Your Deep Learning Partner

VIRTUAL INTERNSHIP

DATA ANALYSIS

LISUM01

CLOUD AND API DEPLOYMENT

Payal Upadhyay

2021/07/08

Contents

Introduction	3
API based deployment (on Postman)	3
api.py	3
Postman	4
Cloud based deployment (Heroku)	6
app.py	6
Model.py	6
index.html	7
requirements.txt	8
Procfile	9
Heroku	9
Conclusion	11

Name: Payal Upadhyay

Batch Code: LISUM101

Submission Date: 2021/07/08

Submitted to: Data Glacier

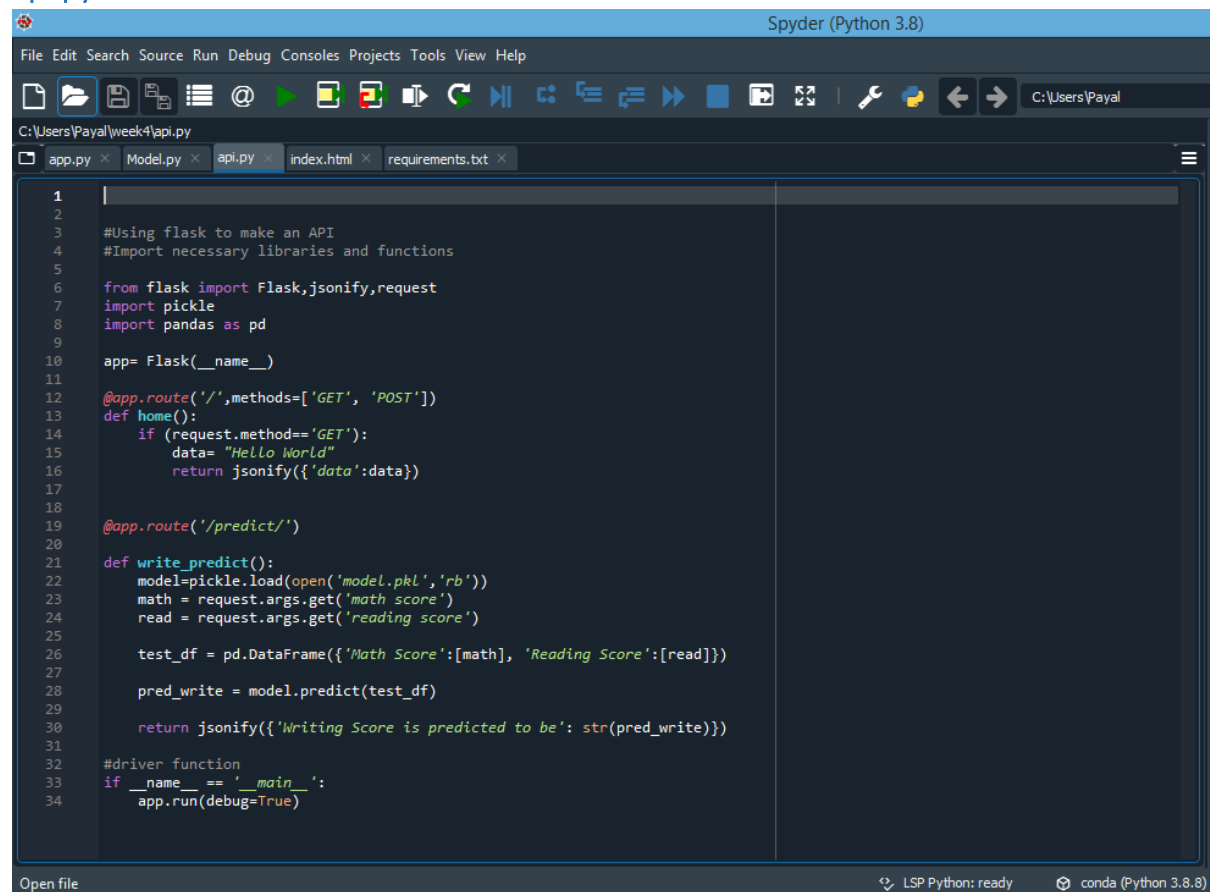
Introduction

I have used StudentsPerformance.csv (available on www.kaggle.com) to predict a student's writing score based on their math score as well as reading score. The data was split into a train set (70 %) and test set (30%). Linear Regression was used to make the prediction. The model was deployed on Postman and Heroku. Snapshots of files created and instructions on the process is provided below.

API based deployment (on Postman)

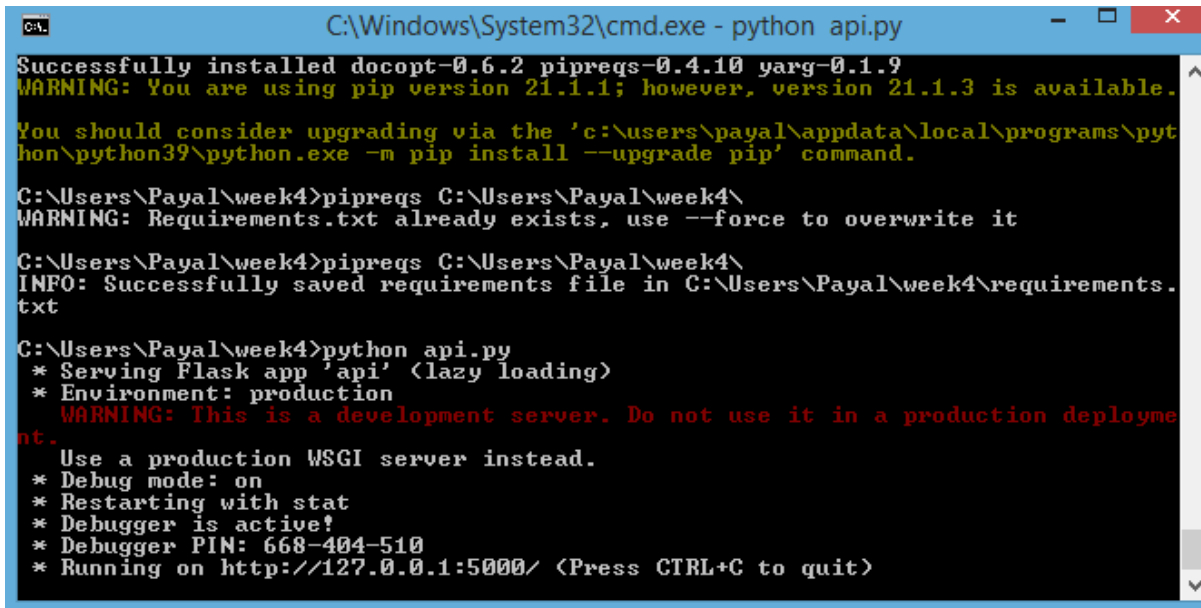
Aim: Predict the writing score based on math score and reading score

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 @app.route('/predict/')
19
20 def write_predict():
21     model = pickle.load(open('model.pkl', 'rb'))
22     math = request.args.get('math score')
23     read = request.args.get('reading score')
24
25     test_df = pd.DataFrame({'Math Score': [math], 'Reading Score': [read]})
26
27     pred_write = model.predict(test_df)
28
29     return jsonify({'Writing Score is predicted to be': str(pred_write)})
30
31 #driver function
32 if __name__ == '__main__':
33     app.run(debug=True)
```

- 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.

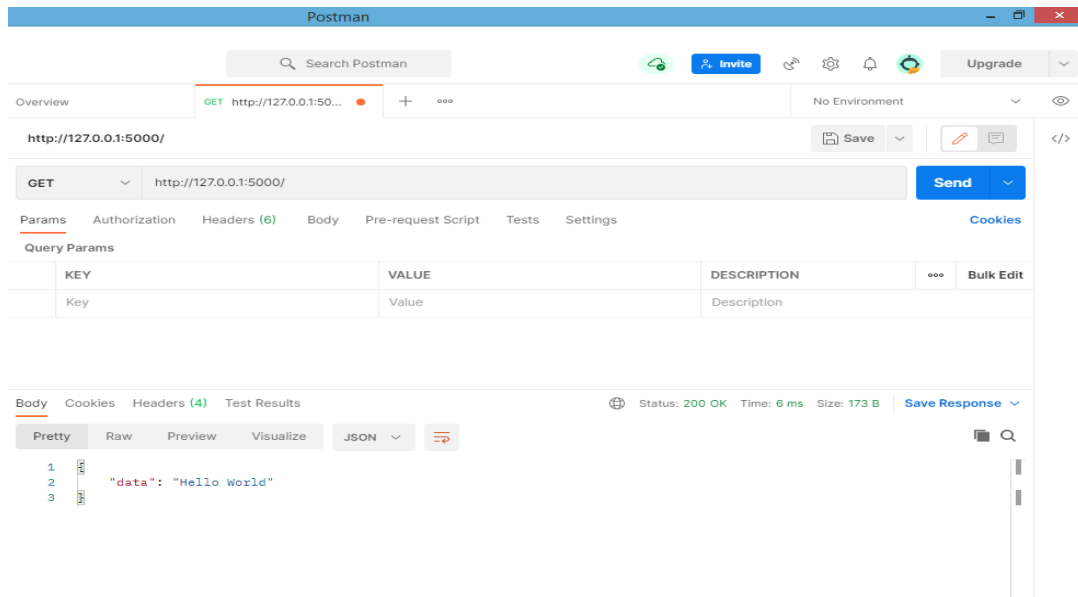
C:\Users\Payal\week4>pipreqs C:\Users\Payal\week4\
WARNING: Requirements.txt already exists, use --force to overwrite it

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

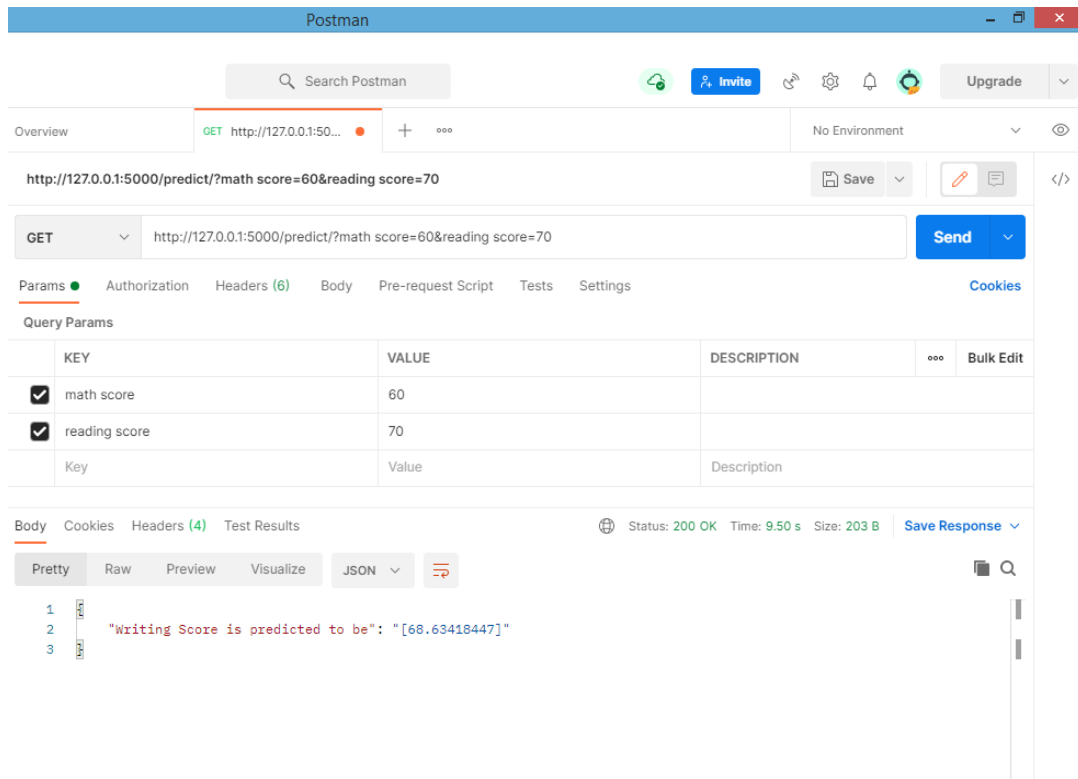
C:\Users\Payal\week4>python api.py
* Serving Flask app 'api' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: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

Postman

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

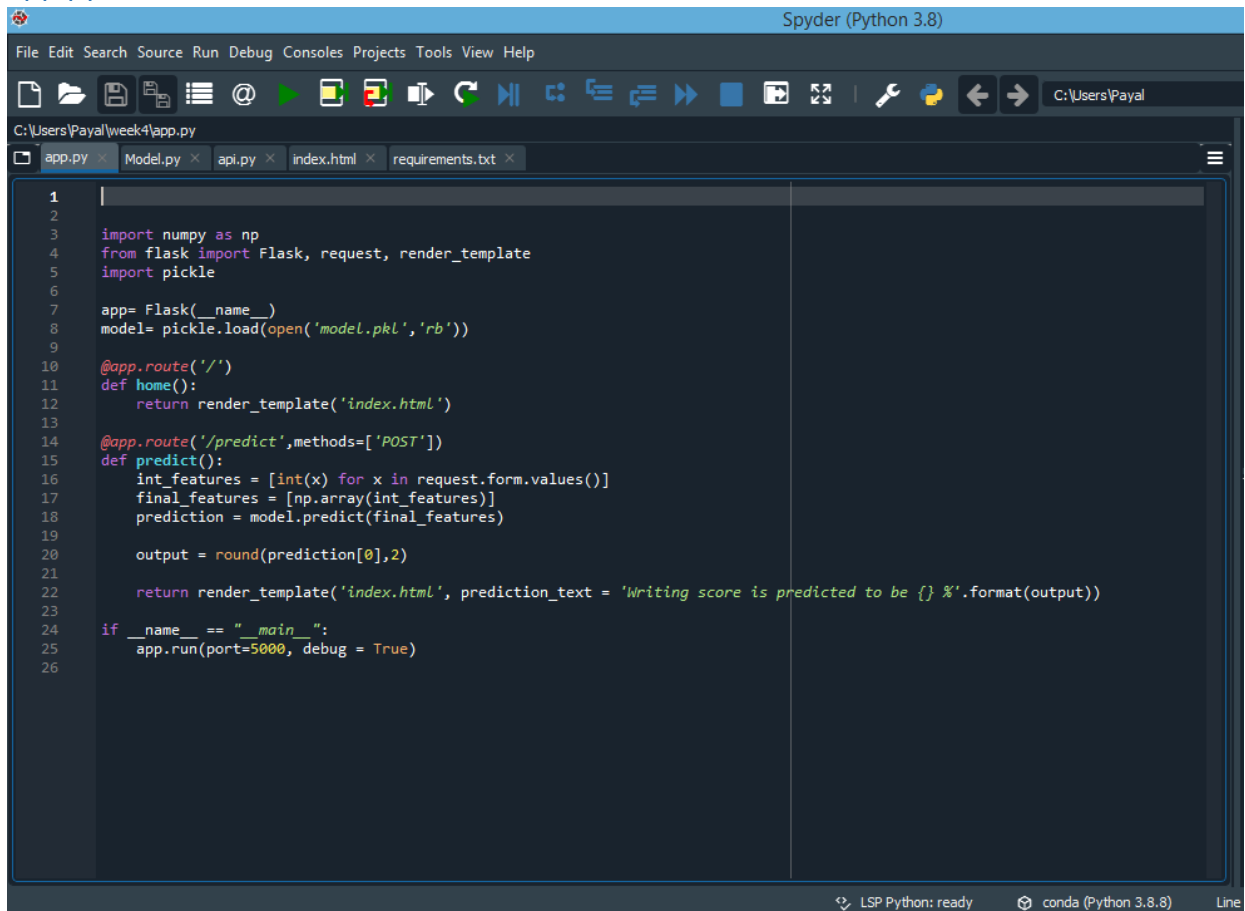


- Enter 'predict\' to the url
- Enter parameters
- Click send to see output



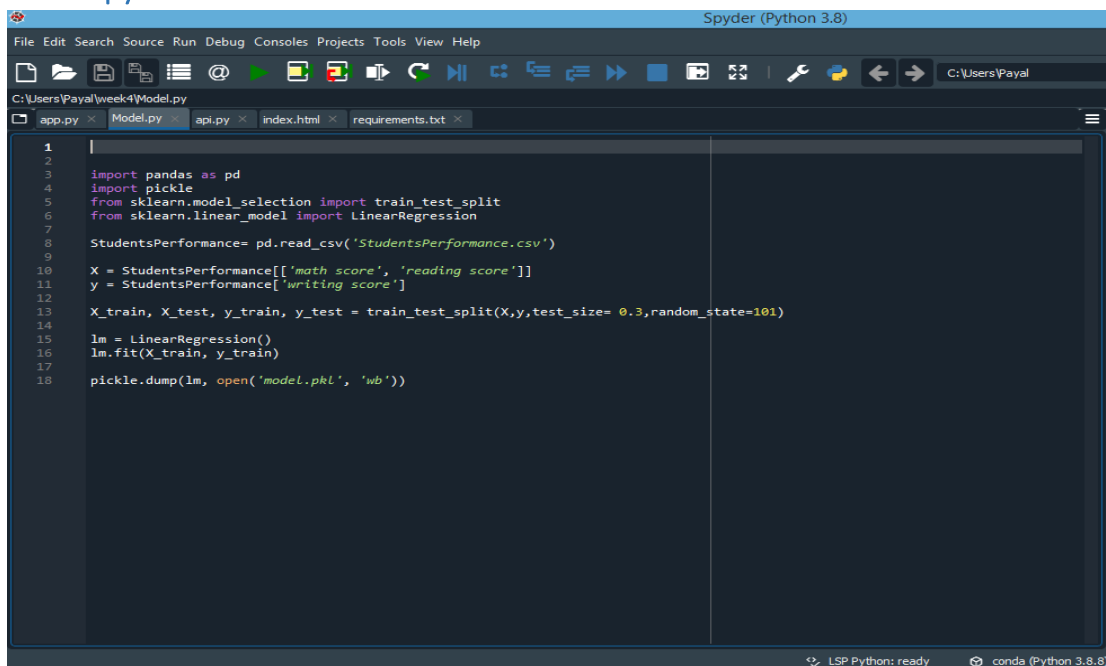
Cloud based deployment (Heroku)

app.py

A screenshot of the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations and execution. The main editor window displays the code for app.py, which is a Flask web application. The code includes imports for numpy, flask, request, render_template, and pickle. It defines a Flask app, a home route, and a predict route that takes POST requests, processes them, and returns a prediction. The app is run on port 5000 with debug mode enabled. The status bar at the bottom indicates 'LSP Python: ready' and 'conda (Python 3.8.8)'.

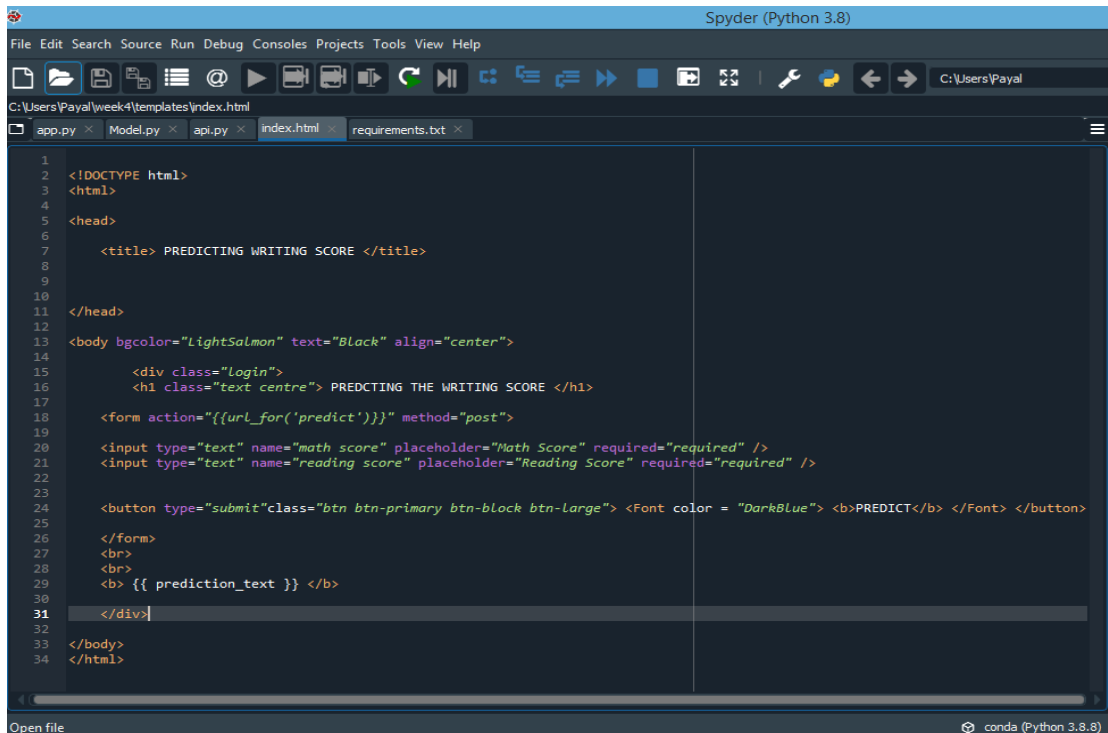
```
1  
2  
3 import numpy as np  
4 from flask import Flask, request, render_template  
5 import pickle  
6  
7 app= Flask(__name__)  
8 model= pickle.load(open('model.pkl','rb'))  
9  
10 @app.route('/')  
11 def home():  
12     return render_template('index.html')  
13  
14 @app.route('/predict',methods=['POST'])  
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

A screenshot of the Spyder Python IDE interface, showing the Model.py file. The code in Model.py is used for training a linear regression model. It imports pandas, pickle, and sklearn modules. It reads a CSV file named 'StudentsPerformance.csv', extracts 'math score' and 'reading score' as features (X) and 'writing score' as the target variable (y). It then splits the data into training and testing sets, trains a LinearRegression model, and saves the trained model as 'model.pkl'. The status bar at the bottom shows 'LSP Python: ready' and 'conda (Python 3.8.8)'.

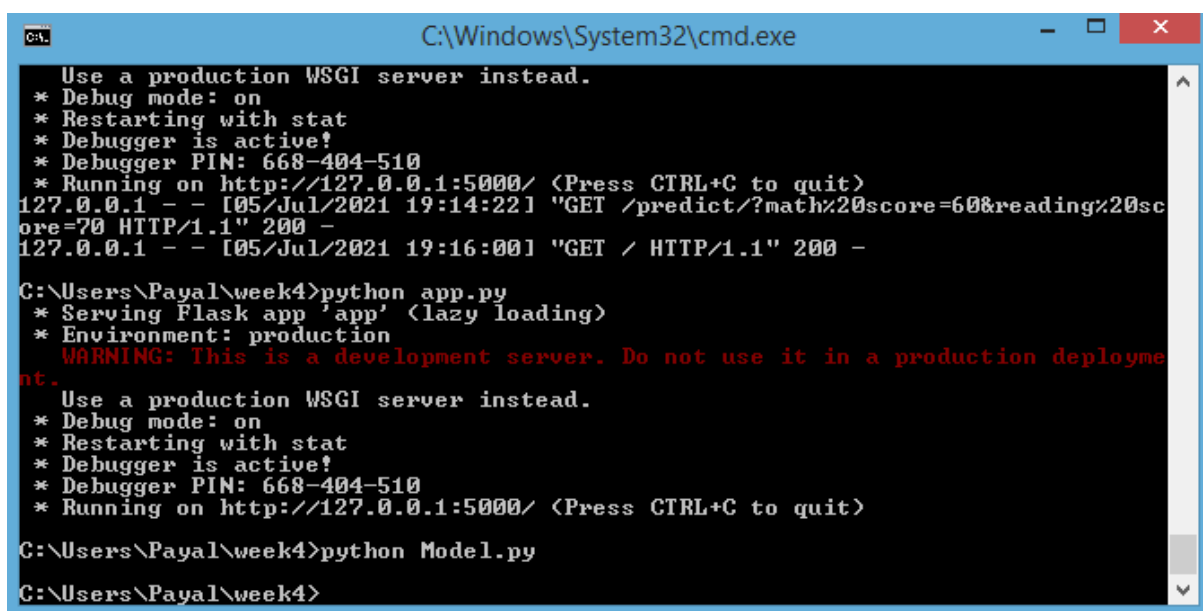
```
1  
2  
3 import pandas as pd  
4 import pickle  
5 from sklearn.model_selection import train_test_split  
6 from sklearn.linear_model import LinearRegression  
7  
8 StudentsPerformance= pd.read_csv('StudentsPerformance.csv')  
9  
10 X = StudentsPerformance[['math score', 'reading score']]  
11 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'))
```

index.html



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

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



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

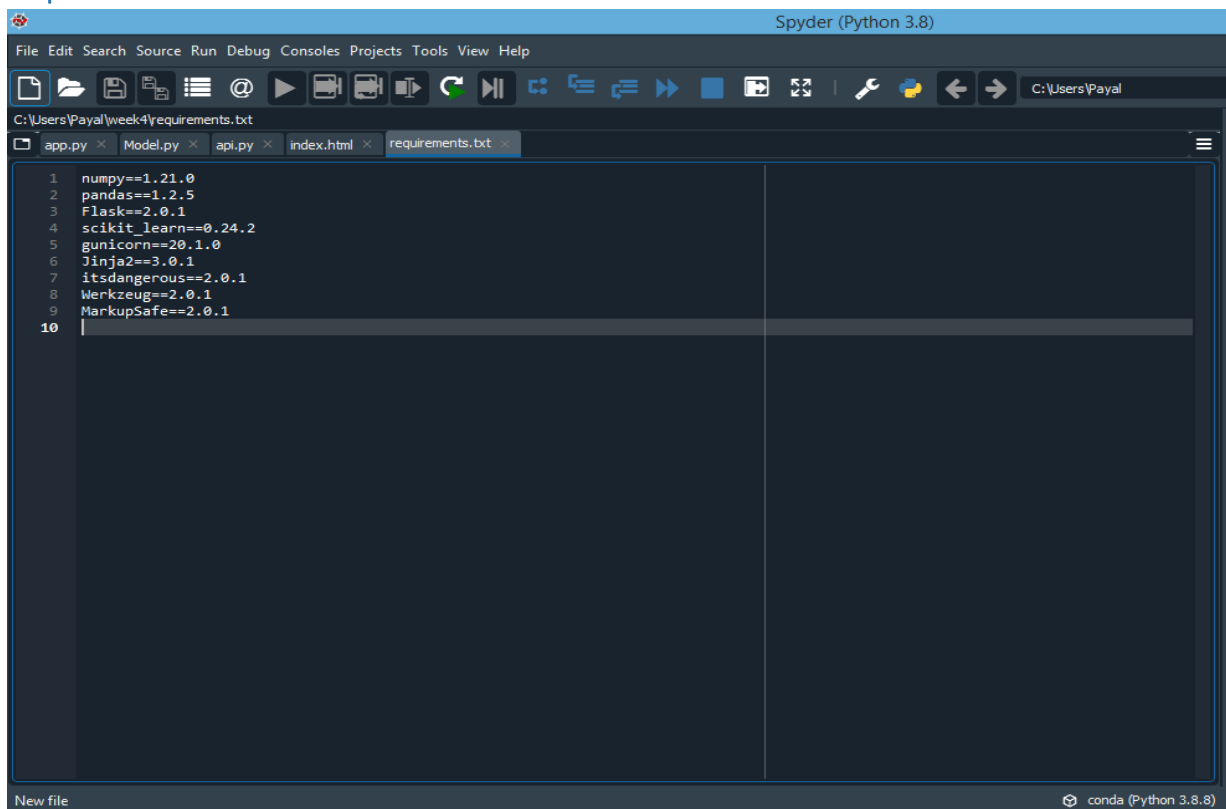
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 668-404-510
* Running on http://127.0.0.1:5000/ <Press CTRL+C to quit>
127.0.0.1 - - [05/Jul/2021 19:14:22] "GET /predict/?math%20score=60&reading%20score=70 HTTP/1.1" 200 -
127.0.0.1 -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



The screenshot shows the Spyder Python IDE interface. The title bar indicates 'Spyder (Python 3.8)'. The menu bar includes 'File', 'Edit', 'Search', 'Source', 'Run', 'Debug', 'Consoles', 'Projects', 'Tools', 'View', and 'Help'. The toolbar contains various icons for file operations and execution. The file explorer on the left shows the project structure with files 'app.py', 'Model.py', 'api.py', 'index.html', and 'requirements.txt'. The 'requirements.txt' file is open in the editor, displaying the following content:

```
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
```

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

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

The screenshot shows a GitHub repository page for 'payal-upadhyay/Week5'. The repository has 1 branch and 0 tags. The commit history is as follows:

File	Commit Message	Time Ago
templates	Create index.html	38 minutes ago
.gitattributes	Initial commit	2 days ago
Model.py	Add files via upload	2 days ago
Procfile	Update Procfile	2 hours ago
StudentsPerformance.csv	Add files via upload	5 minutes ago
app.py	Add files via upload	3 hours ago
model.pkl	Add files via upload	2 days ago
requirements.txt	Add files via upload	1 hour ago

At the bottom, there is a button to 'Add a README with an overview of your project.'

Procfile

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

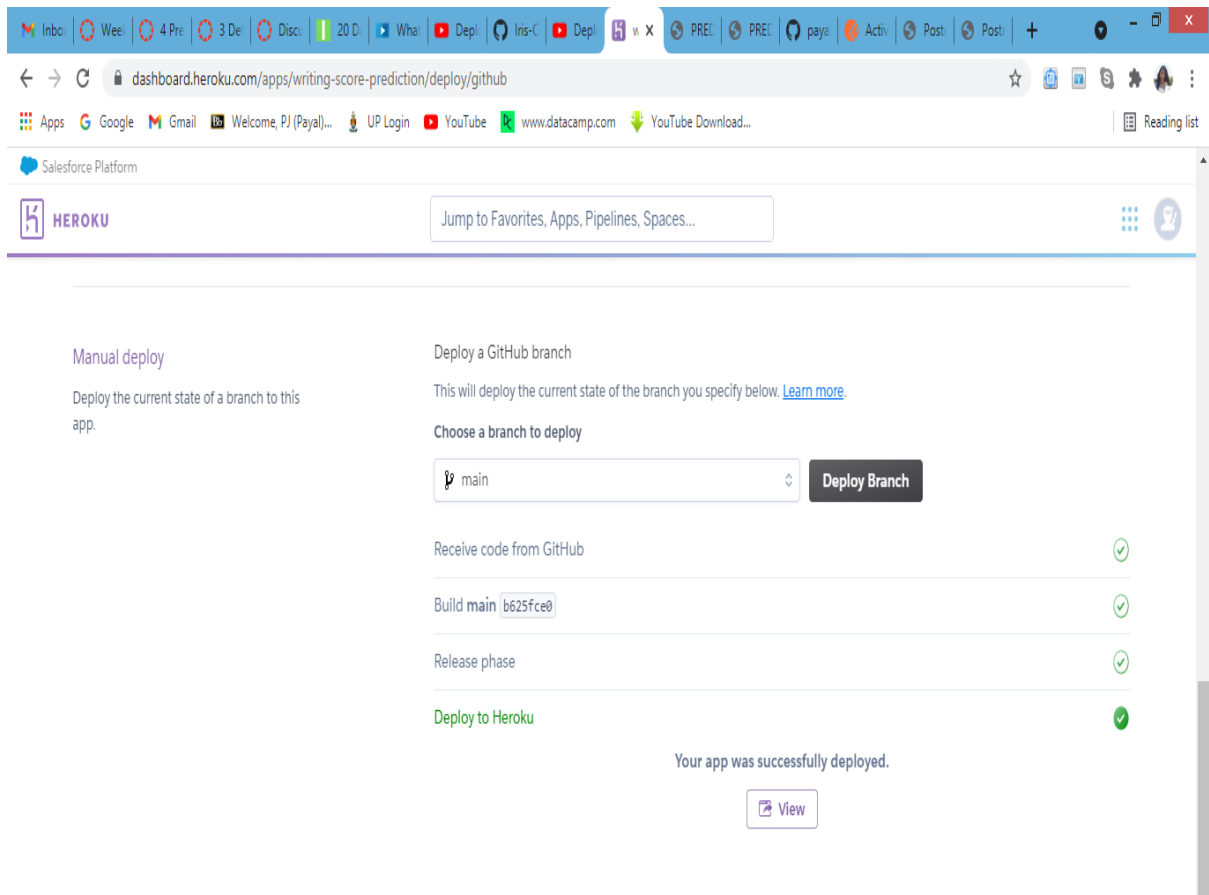
The screenshot shows the 'Procfile' file in the 'Week5' repository. The file content is:

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

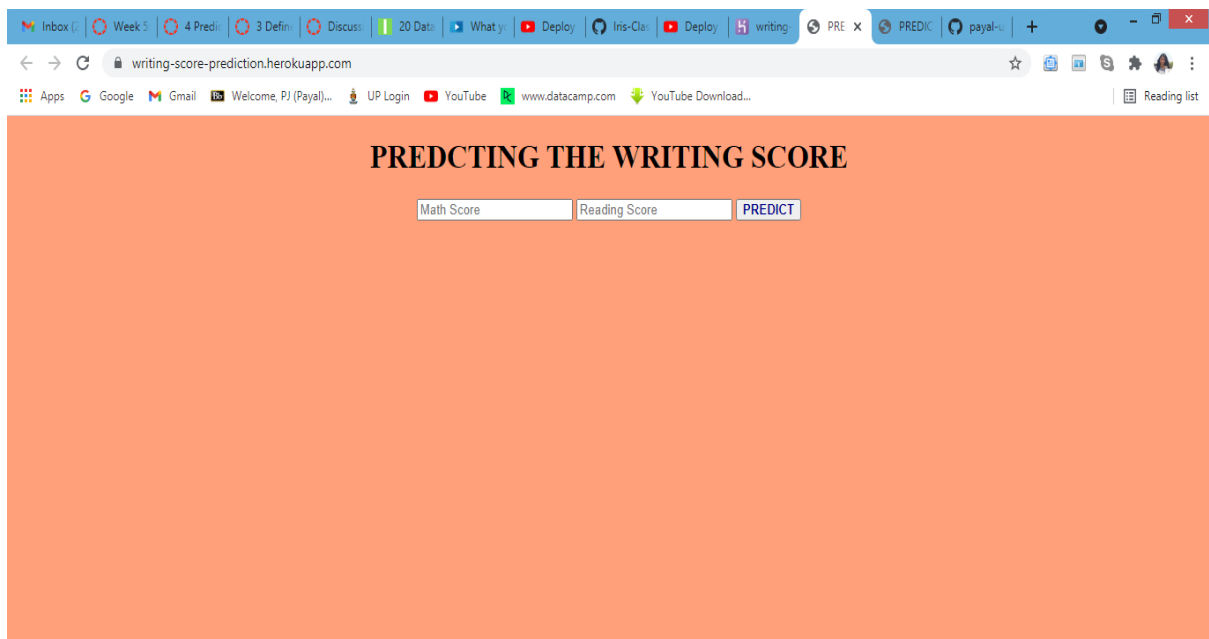
The file is 1 line (1 sloc) and 22 Bytes. It was updated 4 hours ago by the latest commit 0e9cfbc.

Heroku

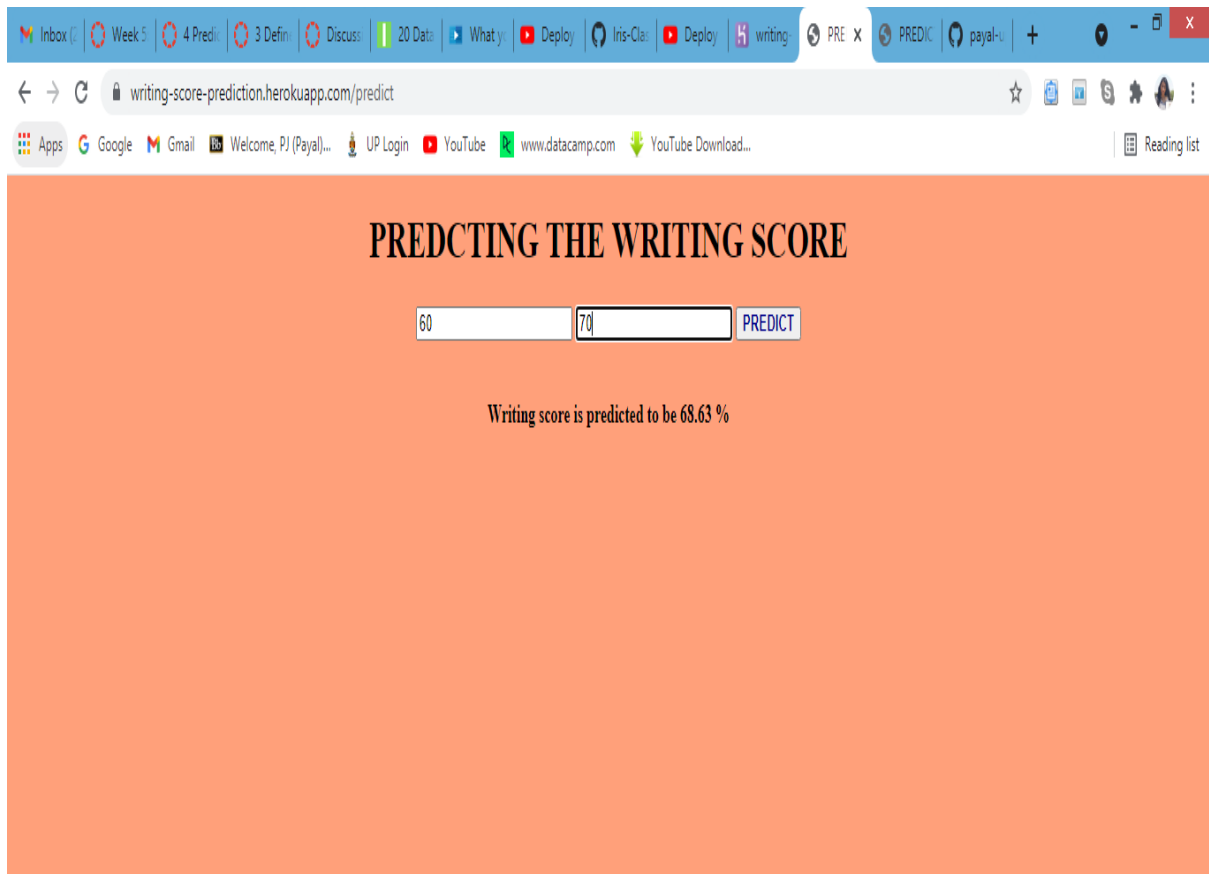
- Create and log into Heroku account
- Select 'create new app' and name your app
- Enter repository name and connect GitHub repository to Heroku
- Click 'Deploy Branch' in order to deploy



- After successful deployment click on view to view the webpage



- Test the webpage by inputting parameters



The screenshot shows a web browser window with the address bar displaying `writing-score-prediction.herokuapp.com/predict`. The page has an orange background and the title "PREDICTING THE WRITING SCORE" in bold black text. Below the title, there are two input fields: the first contains the number "60" and the second contains "70". To the right of these fields is a blue button labeled "PREDICT". Below the input fields, the text "Writing score is predicted to be 68.63 %" is displayed.

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

Conclusion

After inputting a few scores I realized that the writing score that is predicted is more correlated with the reading score and there is very little correlation between writing score and math score.