Name: **Rakshith Nagaraju**

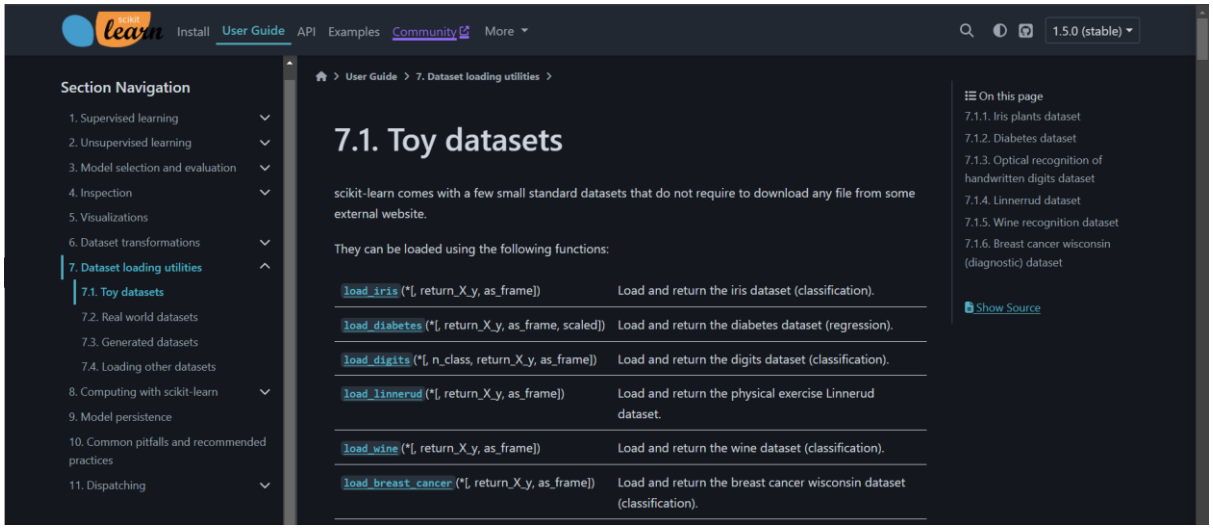
Batch Code: **LISUM33**

Submission date: **28th May 2024**

Submitted to: **Data Glacier Internship April 2024 – July 2024**

Document URL: https://github.com/Rakshith-611/Data-Glacier/tree/Week-4

Step 1: Selecting toy dataset.



Scikit-Learn provides in-built toy datasets as shown above.

For the purposes of this task, I have chosen the 'linnerud' dataset. This dataset contains 20x3 for both data and targets, so it serves us well.

Step 2: Analysing the dataset.

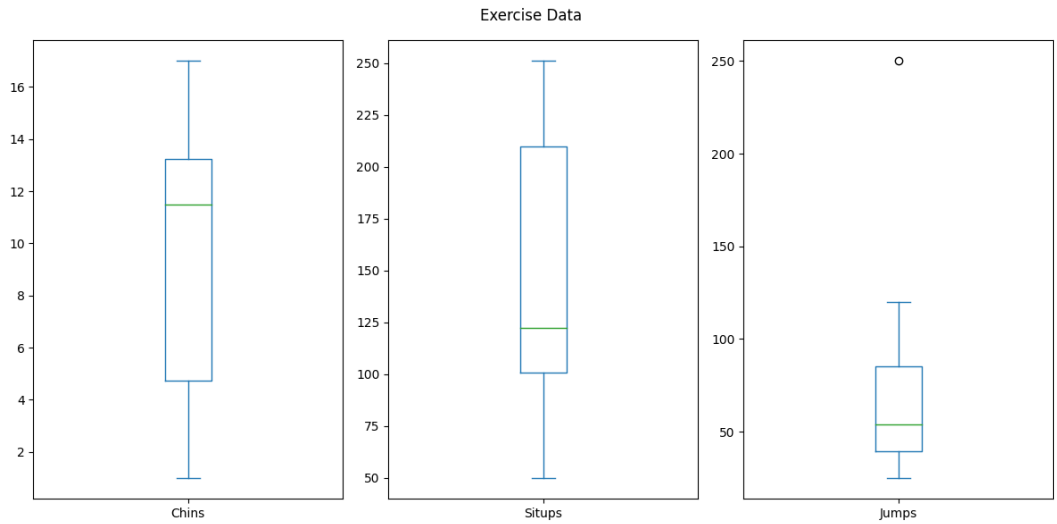


Fig 1: Data visualisation.

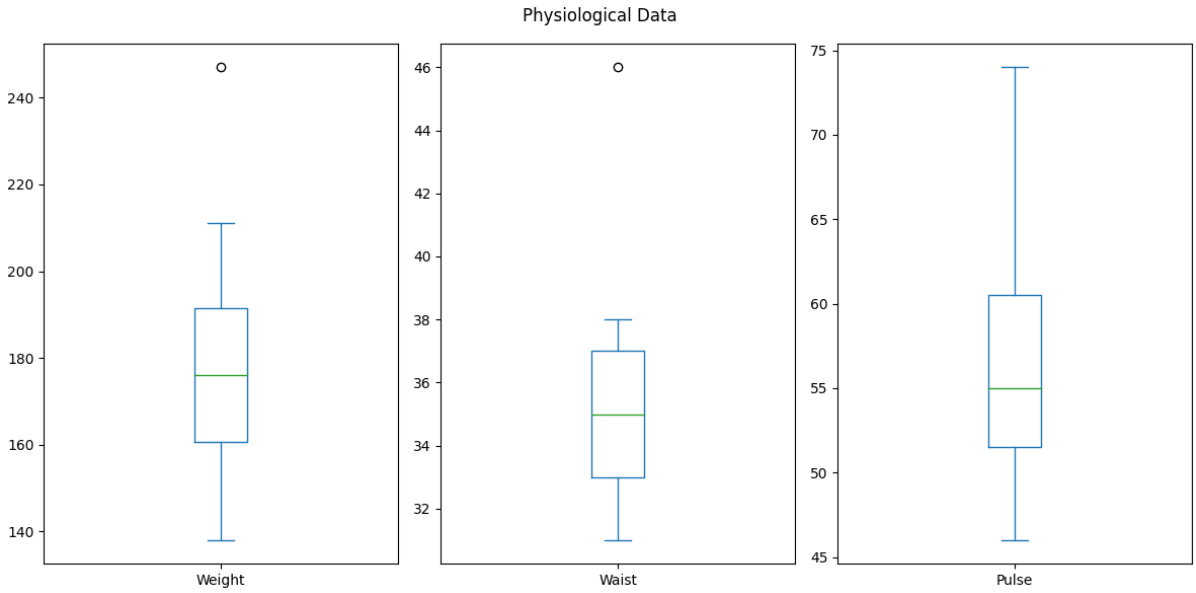


Fig 2: Target Visualisation.

Seems that there is only one outlier in the dataset which will not hamper the regression model that much.

Step 3: Saving the model.

```
from sklearn.datasets import load_linnerud
from sklearn.linear_model import LinearRegression
import pandas as pd
import pickle


dataset = load_linnerud()

# Create DataFrames for the features and targets
# Interchanging data and target because it is easier to know personal phisiological data
X = pd.DataFrame(dataset.target, columns=dataset.target_names)
y = pd.DataFrame(dataset.data, columns=dataset.feature_names)

regressor = LinearRegression()

regressor.fit(X,y)

pickle.dump(regressor, open('model.pkl', 'wb'))

model = pickle.load(open('model.pkl','rb'))
print(model.predict([[160, 30, 60]]))
```

Fig 3: Saving the model from sklearn datasets as pickle file.

Data variables: “Weight (lbs)”, “Waist (inches)”, “Pulse (bpm)”

Target variables: “Chin up count”, “Sit up count”, “Jump count”

On providing test case variables as [160, 30, 60]

The model predicted [15.77114932, 226.44985901, 77.52629495] as the output.

Step 4: Deploying the Flask model.

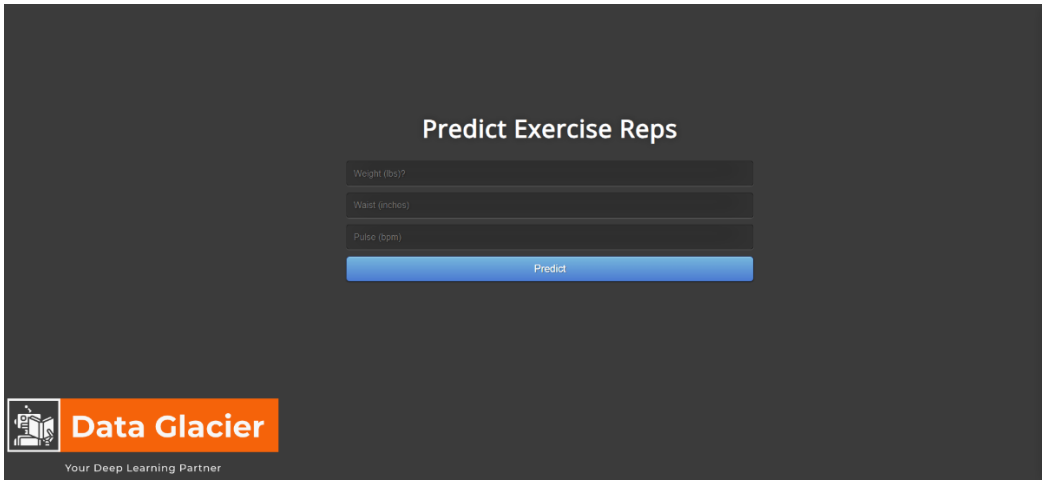
app.py > predict

```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)

    chin_ups = round(prediction[0, 0])
    sit_ups = round(prediction[0, 1])
    jumps = round(prediction[0, 2])

    return render_template('index.html', prediction_text=f"Chin Up's: {chin_ups} | Sit Up's: {sit_ups} | Jumps: {jumps}")


if __name__ == "__main__":
    app.run(port=5000)
```

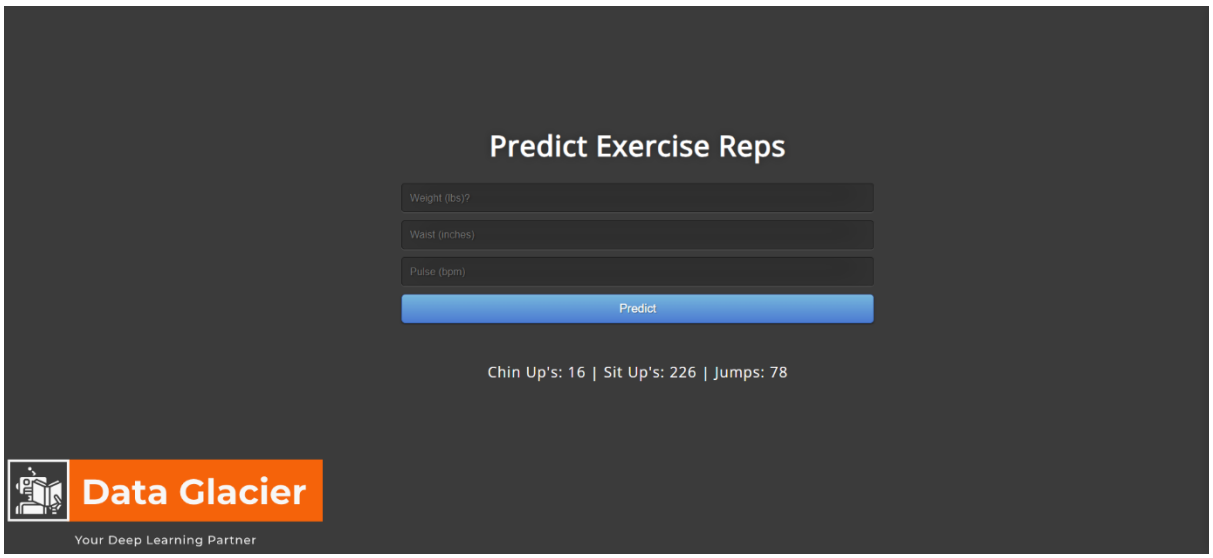
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 * Serving Flask app 'app'
 * Debug mode: off
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
127.0.0.1 - - [28/May/2024 22:41:43] "GET / HTTP/1.1" 200 -
```

On running the flask app on local port ‘5000’, we can see the following page:



And on providing the same input that was provided while training the model ([160, 30, 60]), we obtain the result page:



This is in line with the model prediction when rounded off to the nearest integer value which makes sense for repetitions of an exercise.

The same style sheet and index page as the example provided was used with modifications where necessary to accommodate the different dataset with different data and target variables.