

Model Deployment using Flask

Name: Joseph Pang

Batch code: LISUM26

Submission date: 11/04/2023

Submitted to: Data Glacier

Abstract

This project has been written for the beginners of model deployment. With a simple linear regression example, a model was created on IntelliJ using Flask.

Table of Contents:

- Installing Flask on your Machine
- Setting up the Project WorkFlow
- Build Machine Learning Model
- IntelliJ usage
- Save the Model
- Connect the Webpage with the Model
- Working of the Deployed Model

Installing Flask on your Device

```
# If you are using pip
$ pip install flask

# For Linux
$ sudo apt-get install python3-flask
```

Setting up the Project WorkFlow

- Model Building
- Save the model and setup app
- Webpage Template
- Predict weight and send results

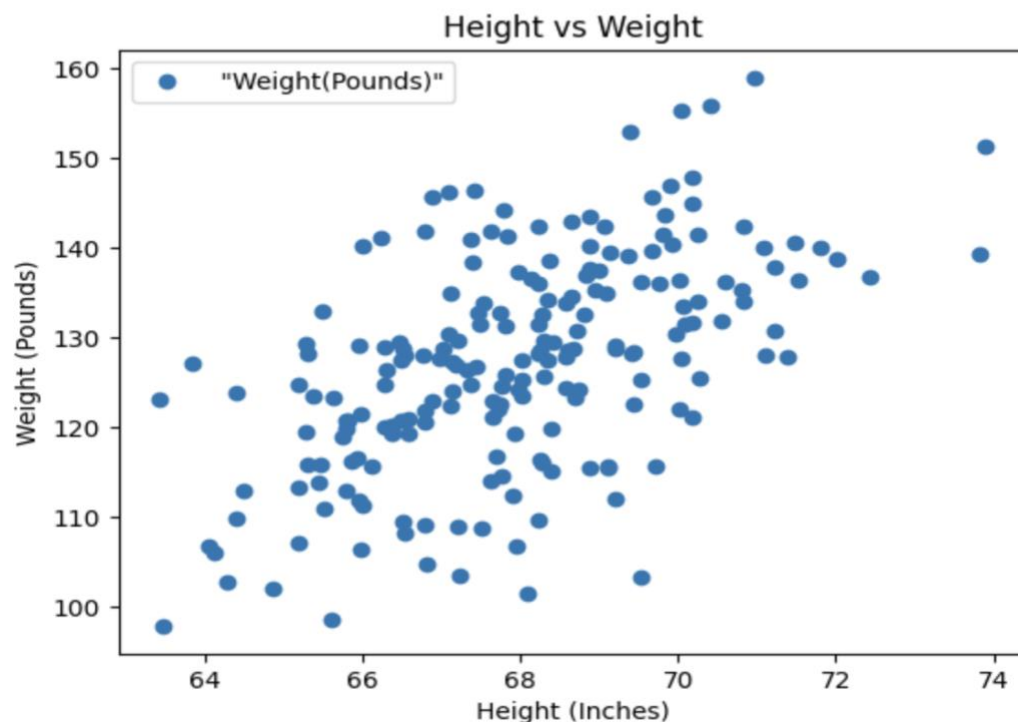
Build Machine Learning Model

I worked from jupyter notebook to create our learning model. Our dataset has 200 rows and 2 columns: Height(Inches), Weight(Pounds). Here is our dataset using the head() function:

	Height(Inches)"	"Weight(Pounds)"
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

Let's plot our data points on a scatter plot to visualize our dataset and see if we can find a correlation between the two features, height and weight.

```
height_weight.plot(x = 1, y = 2, style = "o")  
plt.title("Height vs Weight")  
plt.xlabel("Height (Inches)")  
plt.ylabel("Weight (Pounds)")  
plt.show()
```



Now that we have our attributes and labels, the next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in `train_test_split()` method:

```
[6]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

[7]: from sklearn.linear_model import LinearRegression
      regressor = LinearRegression()
      regressor.fit(X_train, y_train)
```

Let's retrieve the slope (coefficient of x):

```
[9]: regressor.coef_
```

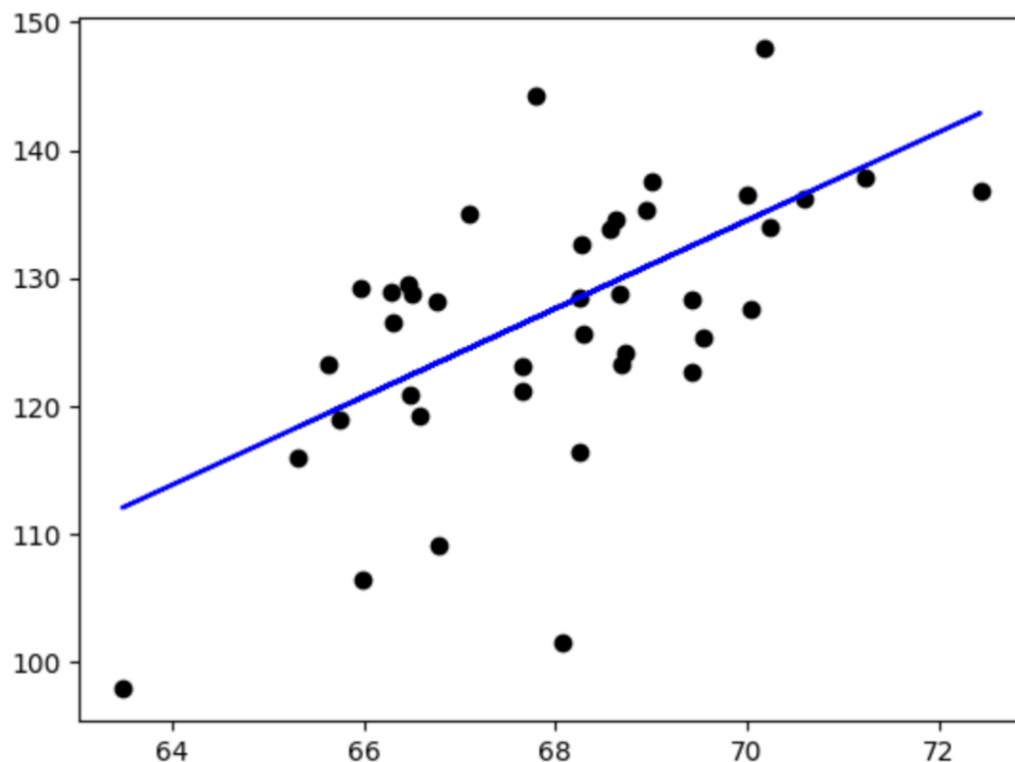
```
[9]: array([[3.44134897]])
```

Making Predictions: Now that we have trained our algorithm, it's time to make some predictions and plot our test sets with our linear regression line.

```
[10]: y_pred = regressor.predict(X_test)

[11]: plt.scatter(X_test, y_test, color = "black")
      plt.plot(X_test, y_pred, color = "blue")

[11]: [<matplotlib.lines.Line2D at 0x1523c51b0>]
```



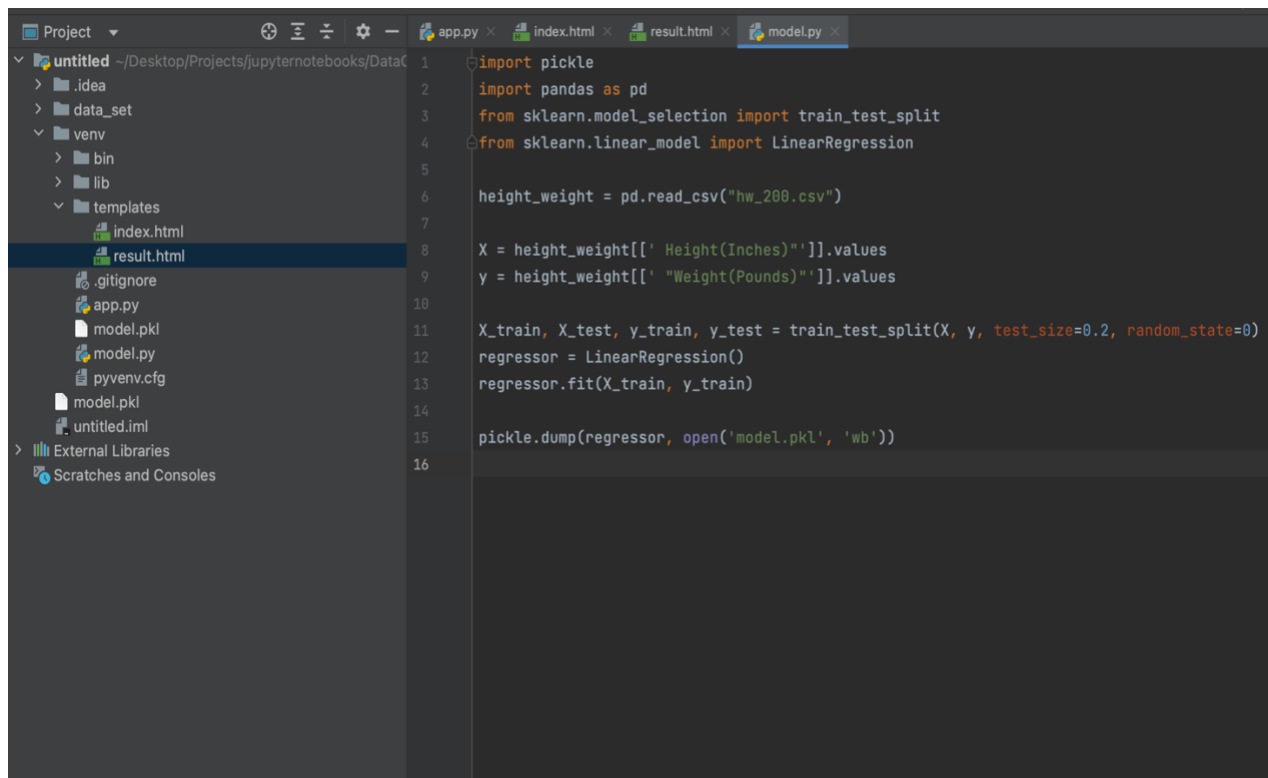
Save the Model: Save regression model using pickle.

```
[12]: import pickle
```

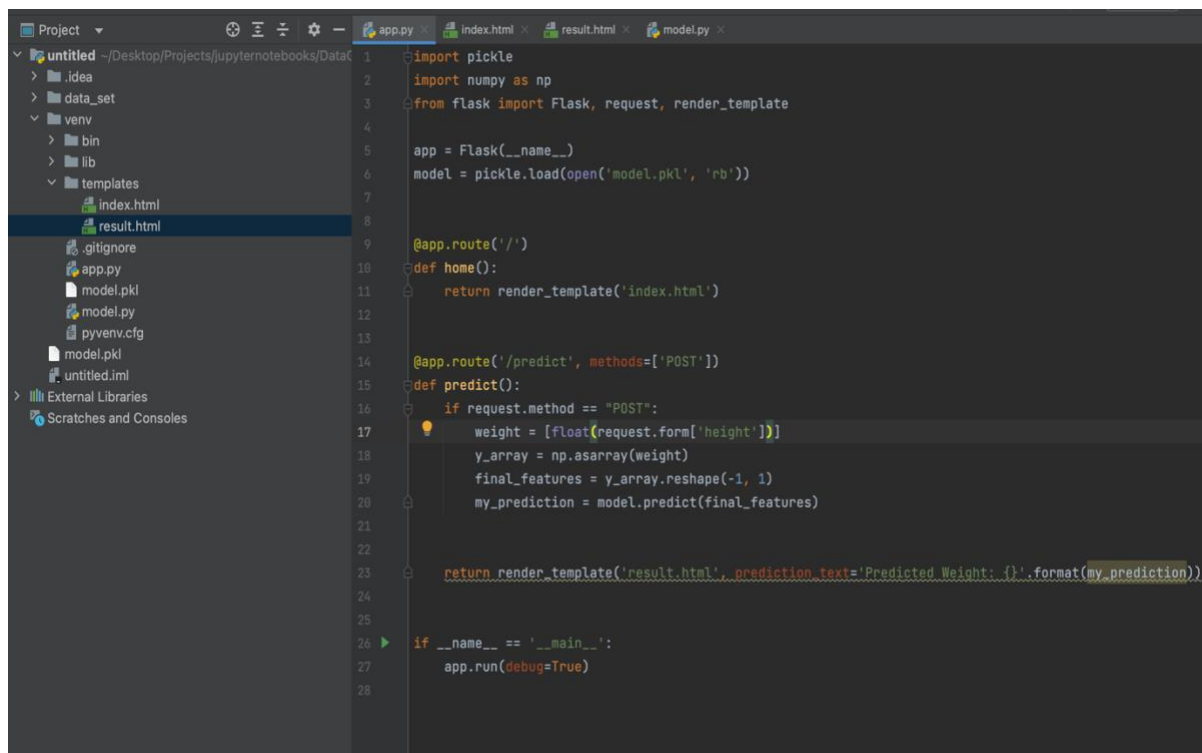
```
[13]: Regressor_Machine = open("model.pkl", "wb")
pickle.dump(regressor, Regressor_Machine)
Regressor_Machine.close()
```

Project Snapshots IntelliJ

Save the Model (model.py)



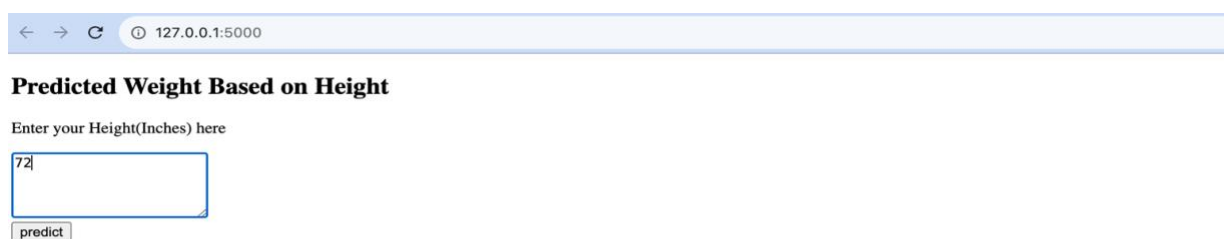
Connect the Webpage with the Model (app.py)



```
1 import pickle
2 import numpy as np
3 from flask import Flask, request, render_template
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8
9 @app.route('/')
10 def home():
11     return render_template('index.html')
12
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     if request.method == "POST":
17         weight = [float(request.form['height'])]
18         y_array = np.asarray(weight)
19         final_features = y_array.reshape(-1, 1)
20         my_prediction = model.predict(final_features)
21
22
23     return render_template('result.html', prediction_text='Predicted Weight: {}'.format(my_prediction))
24
25
26 if __name__ == '__main__':
27     app.run(debug=True)
28
```

Deployed Model on Flask server

We have successfully started the Flask server! Open your browser and go to this address: <http://127.0.0.1:5000>. You will see that the Flask server has rendered our index.html template as our home page. Then after you enter your height in inches as a float, you click on our predict URL which will take us to the result.html page with our regressor model's prediction of your weight given the height entered.



← → ↻ 127.0.0.1:5000

Predicted Weight Based on Height

Enter your Height(Inches) here

Result Page: I entered 72 inches as float to our home page which is 6 feet tall, then we click on predict to give us our result page:



Predicted Weight Based on Height

Predicted Weight: [[141.37507585]]