Simple Linear Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
```

Importing the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
dataset = pd.read_csv("/content/drive/MyDrive/Datasets/weight-h
print(dataset)
```

```
      Gender     Height      Weight
0       Male  73.847017  241.893563
1       Male  68.781904  162.310473
2       Male  74.110105  212.740856
3       Male  71.730978  220.042470
4       Male  69.881796  206.349801
...      ...        ...         ...
9995  Female  66.172652  136.777454
9996  Female  67.067155  170.867906
9997  Female  63.867992  128.475319
9998  Female  69.034243  163.852461
9999  Female  61.944246  113.649103

[10000 rows x 3 columns]
```

```
X = dataset.iloc[:,1:2]
y = dataset.iloc[:,-1]
```

```
print(X)
```

```
         Height
0     73.847017
```

```
1     68.781904
2     74.110105
3     71.730978
4     69.881796
...         ...
9995  66.172652
9996  67.067155
9997  63.867992
9998  69.034243
9999  61.944246

[10000 rows x 1 columns]
```

```
print(y)
```

```
0       241.893563
1       162.310473
2       212.740856
3       220.042470
4       206.349801
           ...
9995    136.777454
9996    170.867906
9997    128.475319
9998    163.852461
9999    113.649103
Name: Weight, Length: 10000, dtype: float64
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_
```

Training the Simple Linear Regression model on the Training set

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
# Upto here was the builing part, now we have to train the mode


# To connect our model we use the fit method

# X_train contains the features of the dataset
# y_train contins the dependant varaibles
regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Predicting the Test set results

```
regressor.predict(X_test)
# This returns a vector containg the predicted data
```

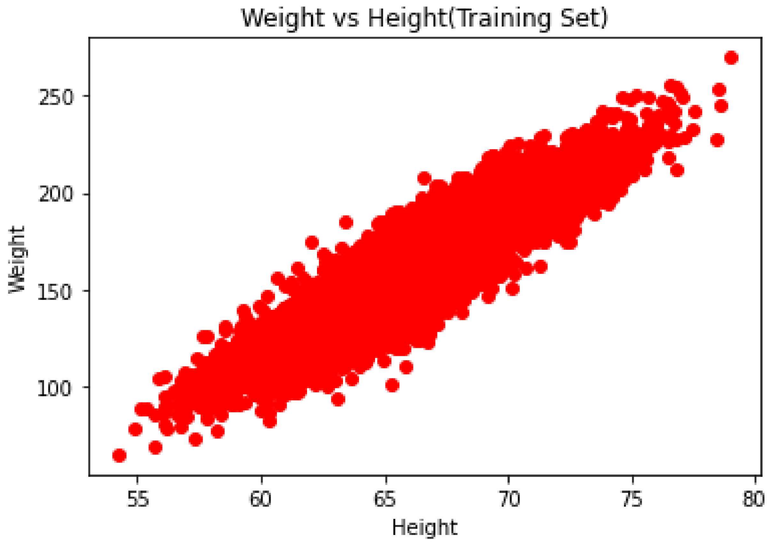
```
array([148.7894772 , 168.43520123, 224.31884497, ..., 159.17576427,
       155.86404539, 144.83449257])
```

```
y_pred = regressor.predict(X_test)
```

Visualising the Training set results

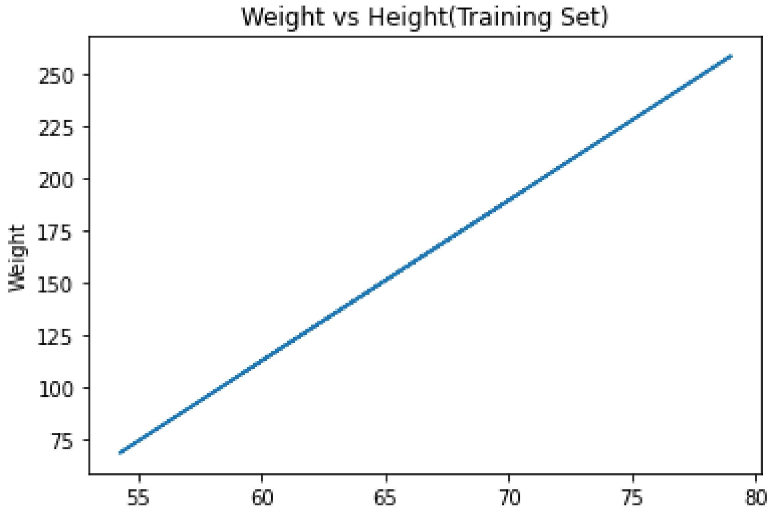
```
plt.title("Weight vs Height(Training Set)")
plt.xlabel("Height")
plt.ylabel("Weight")
# scatter method allows us to put points/coordinates
plt.scatter(X_train, y_train, color="red") # Real vlaues
```

```
<matplotlib.collections.PathCollection at 0x7fc9687762d0>
```



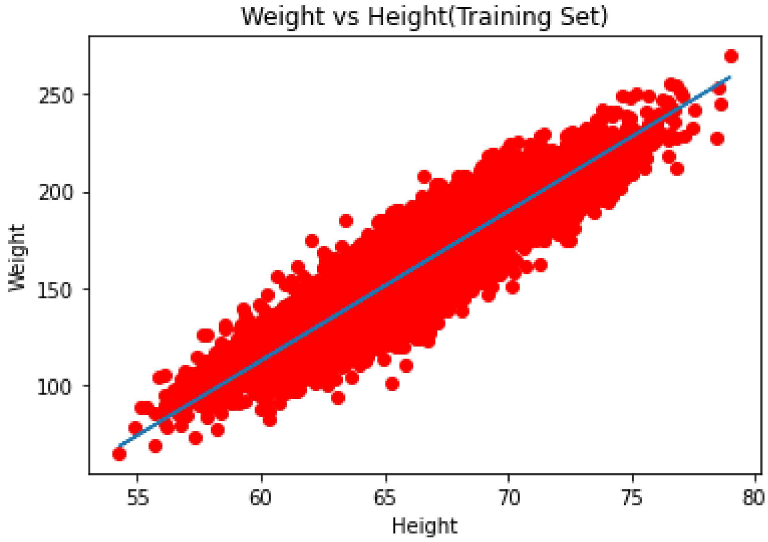
```
# plot method is used to plot the curve of a function(y=b0+b1*x
plt.title("Weight vs Height(Training Set)")
plt.xlabel("Height")
plt.ylabel("Weight")
plt.plot(X_train, regressor.predict(X_train)) # Best fit line o
```

```
[<matplotlib.lines.Line2D at 0x7fc96aa54490>]
```



```
plt.title("Weight vs Height(Training Set)")
plt.xlabel("Height")
plt.ylabel("Weight")
plt.scatter(X_train, y_train, color="red")
plt.plot(X_train, regressor.predict(X_train))
# show is used to display the graphic in the output
plt.show()

# Red dots are real values of salary (x_train, y_train)
# Blue line is the best fit line on training X_train and predic
```

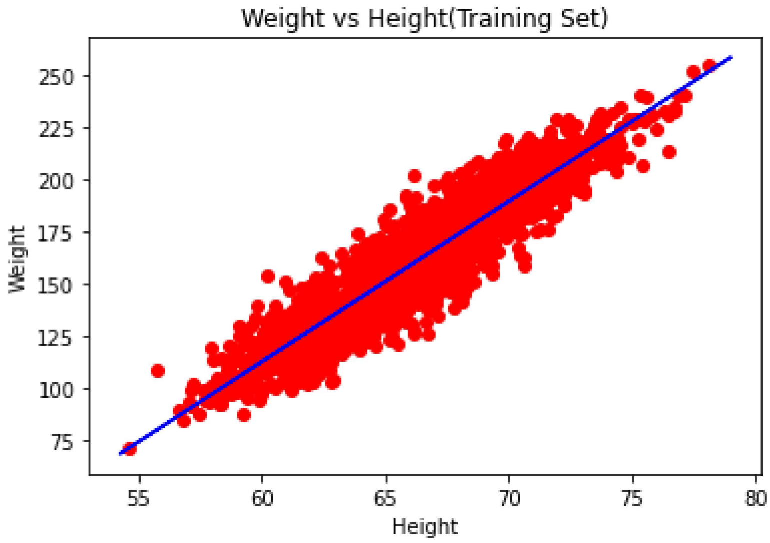


Visualising the Test set results

```
plt.title("Weight vs Height(Training Set)")
plt.xlabel("Height")
plt.ylabel("Weight")
plt.scatter(X_test, y_test, color="red")

# Predicted salries of test set will on the same rgression line
plt.plot(X_train, regressor.predict(X_train), color="blue")
plt.show()

# Red are new observations(test set)
# Blue is our best fit line after traning on the available trai
```



Making a single prediction (for example the salary of an employee with 12 years of experience)

```
input_value = int(input("Enter Height: "))
print(regressor.predict([[input_value]]))
```

```
Enter Height: 185
[1076.04180898]
```

Therefore, our model predicts that the salary of an employee with 12 years of experience is $ 138967,5.

Important note: Notice that the value of the feature (12 years) was input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting 12 into a double pair of square brackets makes the input exactly a 2D array. Simply put:

$12 \rightarrow \text{scalar}$

$[12] \rightarrow 1\text{D array}$

$[[12]] \rightarrow 2\text{D array}$

Getting the final linear regression equation with the values of the coefficients

```
print(regressor.coef_)
print(regressor.intercept_)
```

```
[7.70936331]
-350.1904028560757
```

Therefore, the equation of our simple linear regression model is:

$$\text{Weight} = 7.70936331 \times \text{Height} + -350.190402856075719.$$

Important Note: To get these coefficients we called the "coef_" and "intercept_" attributes from our regressor object. Attributes in Python are different than methods and usually return a simple value or an array of values.

```
from sklearn.metrics import mean_squared_error
```

```
mse = mean_squared_error(y_test, y_pred)
print("MSE: ", mse)
```

```
MSE:  146.53677213957428
```

```
rmse = math.sqrt(mse)
print(rmse)
```

```
12.105237384684957
```