Name: Aniruddh Kulkarni

Roll No: I081

Subject: Machine Learning

Practical: Practical 3

Date: 07-01-22

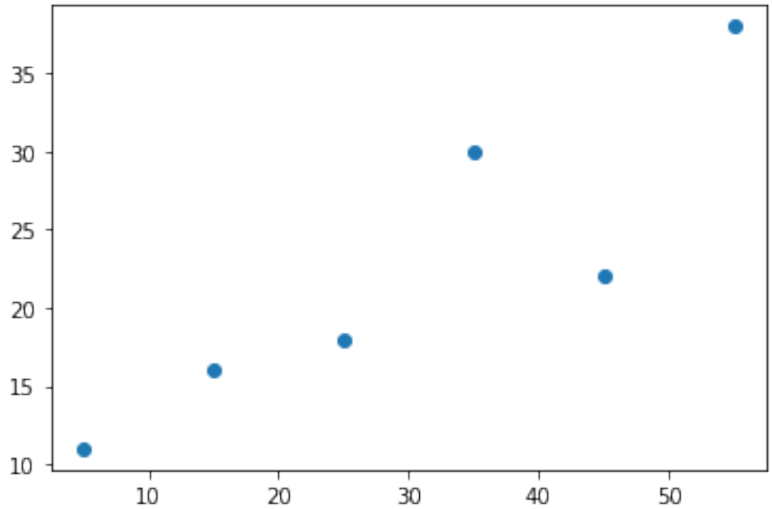
In [4]:
```
# Lab assignment 3
# To write a code for a simple linear regression using SKLearn
# To verify the results using hard coding using Numpy
```

In [2]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [3]:
```
x = np.array([5,15,25,35,45,55])
y = np.array([11,16,18,30,22,38])
```

In [4]:
```
plt.scatter(x,y)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x7fc299d30e80>
```



In [5]:
```
# CREATE THE MODEL
model =LinearRegression()
```

In [6]:
```
x.shape
```

Out[6]: (6,)

In [7]:
```
# do convert the independent variable to n2 dimensional array. This is requ
x = x.reshape((-1,1))
```

In [8]:
```
# fit the model on the data
model.fit(x,y)
```

Out[8]: LinearRegression()

In [9]:
```
r_sq = model.score(x,y)
```

In [10]:
```
print(r_sq)
```

```
0.7913094027030955
```

In [11]:
```
print("Intercept:",model.intercept_)
```

```
Intercept: 8.357142857142856
```

In [12]:
```
print("Slope:", model.coef_)
```

```
Slope: [0.47142857]
```

In [13]:
```
# prediction usng the model
x_predict = np.array([20])
x_predict = x_predict.reshape((-1,1))
y_predict = model.predict(x_predict)
```

In [14]:
```
y_predict
```

Out[14]: array([17.78571429])

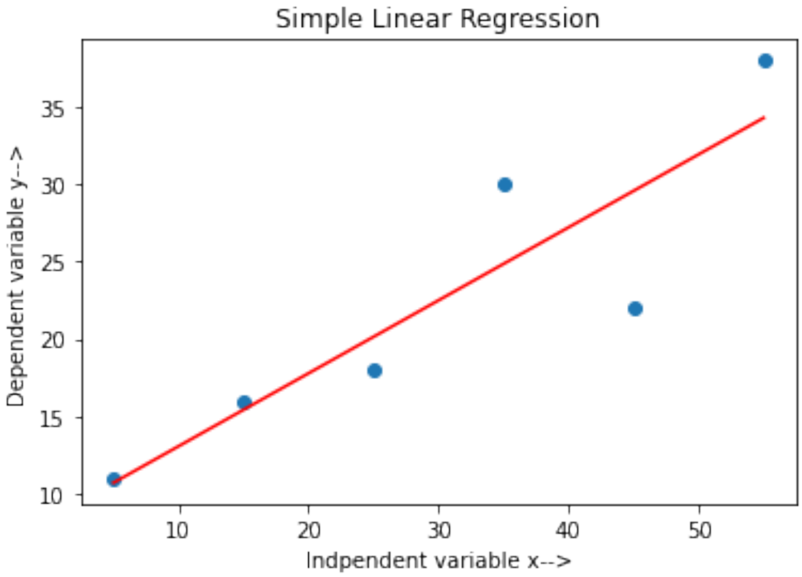
In [15]:
```
# take x_predict as original array x and predict the output
x_predict = x
y_predict = model.predict(x_predict)
```

In [16]:
```
print("Original y: ",y)
print("Predicted y: ", y_predict)
```

```
Original y:  [11 16 18 30 22 38]
Predicted y:  [10.71428571 15.42857143 20.14285714 24.85714286 29.57142857
 34.28571429]
```

In [29]:
```
# visualiusing the regression line on the data
plt.scatter(x,y)
plt.plot(x_predict, y_predict, "r-")
plt.xlabel("Indpendent variable x-->")
plt.ylabel("Dependent variable y-->")
plt.title("Simple Linear Regression")
```

Out[29]: Text(0.5, 1.0, 'Simple Linear Regression')



In [18]:
```
# compute the value of b1, b0 and rsquare
```

In [19]:
```
x = np.array([5,15,25,35,45,55])
y = np.array([11,16,18,30,22,38])
```

In [20]:
```
# computing mean values
xmean = np.average(x)
ymean = np.average(y)
```

In [21]:
```
xymean = np.average(np.multiply(x,y))
xmeansquare = xmean*xmean
xsquarebar = np.average(np.multiply(x,x))
```

In [22]:
```
# computing b1 and b0
b1 = ((xmean*ymean)-xymean)/(xmeansquare - xsquarebar)
b0 = ymean - (b1*xmean)
```

In [23]:
```
b1
```

Out[23]: 0.4714285714285713

In [24]:
```
b0
```

Out[24]: 8.357142857142861

In [25]:
```
# computing predicted value of y
y_predict = b0 + b1*x
```

In [27]:
```
# computing ssr
ssr = np.sum(np.square(y_predict - ymean))
ssr
```

Out[27]: 388.9285714285712

In [28]:
```
# compute sse
sse = np.sum(np.square(y - y_predict))
sse
```

Out[28]: 102.57142857142854

In [30]:
```
print((ssr)/(ssr+sse))
```

0.7913094027030955

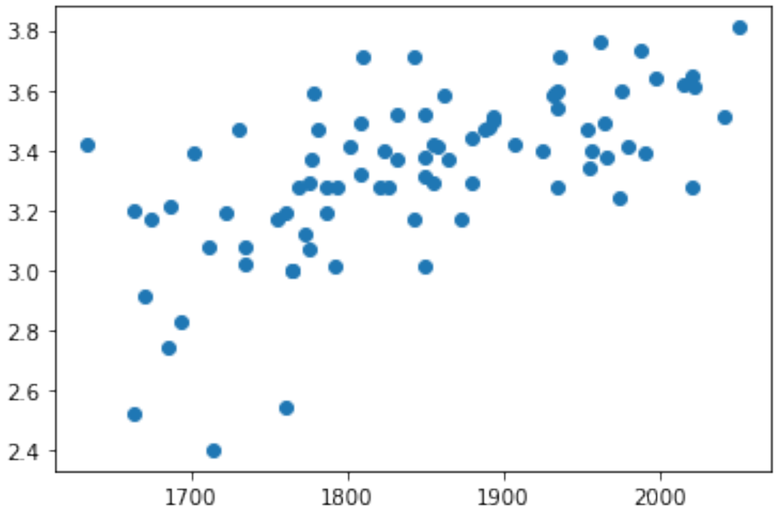
In [31]:
```
import pandas as pd
import statsmodels.api as sm
```

In [32]:
```
data = pd.read_csv('sat_cgpa.csv')
```

In [33]:
```
y = data['GPA']
x = data['SAT']
```

In [34]:
```
plt.scatter(x,y)
```

Out[34]: <matplotlib.collections.PathCollection at 0x7fc2a97f4d90>



In [36]:
```
# Adds a constant coloumn to the dataset inorder to obtain a two-dimension
x = sm.add_constant(x)
x
```

Out[36]:

	const	SAT
0	1.0	1714
1	1.0	1664
2	1.0	1760
3	1.0	1685
4	1.0	1693
...	...	...
79	1.0	1936
80	1.0	1810
81	1.0	1987
82	1.0	1962
83	1.0	2050

84 rows × 2 columns

In [37]:

```
# OLS stands for ordinary least squares and is used as OLS regression here
results = sm.OLS(y,x).fit()
results.summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	GPA	**R-squared:**	0.406
Model:	OLS	**Adj. R-squared:**	0.399
Method:	Least Squares	**F-statistic:**	56.05
Date:	Fri, 07 Jan 2022	**Prob (F-statistic):**	7.20e-11
Time:	21:08:49	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.2750	0.409	0.673	0.503	-0.538	1.088
SAT	0.0017	0.000	7.487	0.000	0.001	0.002

Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+04. This might indicate that there are strong multicollinearity or other numerical problems.