Exercise 2.2

Linear Regression using sklearn

In [15]:
```
import pandas as pd
import numpy as np

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
mount("/content/drive", force_remount=True).
```

In [16]:
```
datafpath = '/content/drive/MyDrive/DATASETS/tempppresdata.csv'

data = pd.read_csv(datafpath)
data.drop(data.columns[2:11], axis=1, inplace=True)
data.drop(data.index[7:12], inplace = True)
data
```

Out[16]:

	Pressure	Temperature
0	40.55	271.8
1	36.19	264.0
2	37.31	238.8
3	32.52	230.7
4	33.71	251.6
5	34.14	257.9
6	34.85	263.9

In [17]:
```
data.shape
```

Out[17]:
```
(7, 2)
```

In [20]:
```
X = data['Pressure']
y = data['Temperature']
```

In [21]:
```
X
```

Out[21]:

	Pressure
0	40.55
1	36.19
2	37.31
3	32.52
4	33.71
5	34.14
6	34.85

dtype: float64

In [22]:
```
y
```

Out[22]:

	Temperature
0	271.8
1	264.0
2	238.8
3	230.7
4	251.6
5	257.9
6	263.9

dtype: float64

In [23]:
```
# gets the correlation of the data set
data.corr()
```

Out[23]:

	Pressure	Temperature
Pressure	1.000000	0.549474
Temperature	0.549474	1.000000

In [45]:
```
y.values.reshape(-1, 1)
```

Out[45]:
```
array([[271.8],
       [264. ],
       [238.8],
       [230.7],
       [251.6],
       [257.9],
       [263.9]])
```

In [52]:
```
#gets makes a linear regression model for my dataset
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X.values.reshape(-1, 1), y.values.reshape(-1, 1))
```

Out[52]:
```
▾ LinearRegression

LinearRegression()
```

In [56]:
```
# get the a or the coefficient

a = model.coef_[0][0]
print(a)
```

```
3.0140649625746136
```

In [57]:
```
b = model.intercept_[0]
print(b)
```

```
146.76914668271797
```

In [28]:
```
# getting the r squared of the pearson r
rsqrd = np.corrcoef(X, y)**2
rsqrd
```

Out[28]:
```
array([[1.      , 0.301922],
       [0.301922, 1.      ]])
```

In [59]:
```
# trying to predict the model using the data from dataset of the independent variab
y_pred = model.predict(X.values.reshape(-1, 1))
```

In [60]:
```
# getting the mean squared error
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y, y_pred)
mse
```

Out[60]:
```
130.6083984899873
```

In [61]:
```
# getting the root mean squared error
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y, y_pred))
rmse
```

Out[61]:
```
11.428403146983717
```