Q.) Explain the linear regression algorithm in detail.

Answer: The goal of the linear regression algorithm is to get the best values for a0 and a1 to find the best fit line. The best fit line should have the least error means the error between predicted values and actual values should be minimized.

Cost function:

The cost function helps to figure out the best possible values for a0 and a1, which provides the best fit line for the data points.

Cost function optimizes the regression coefficients or weights and measures how a linear regression model is performing. The cost function is used to find the accuracy of the **mapping function** that maps the input variable to the output variable. This mapping function is also known as **the Hypothesis function**.

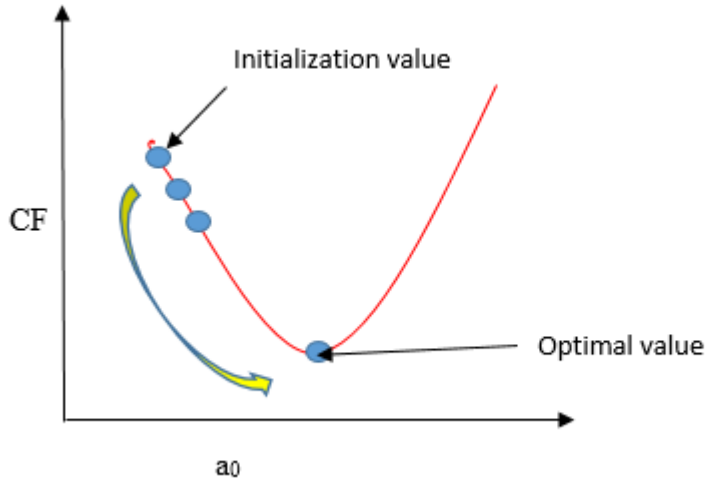
In Linear Regression, **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the predicted values and actual values.

By simple linear equation y=mx+b we can calculate MSE as:

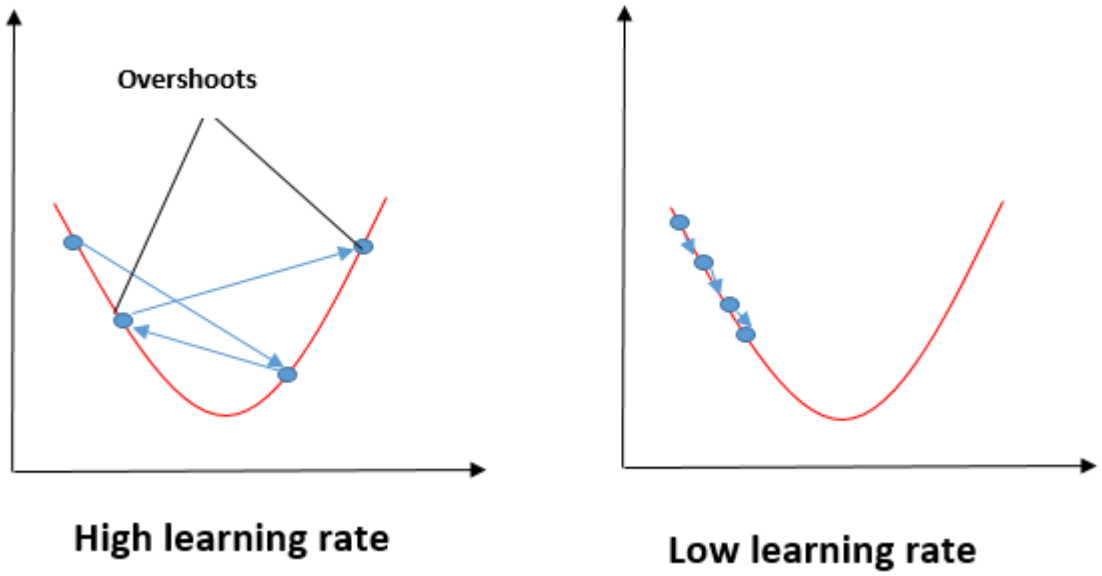
Let's y = actual values, y_i = predicted values

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Using the MSE function, we will change the values of a0 and a1 such that the MSE value settles at the minima. Model parameters ***xi, b (a_0,a_1)*** can be manipulated to minimize the cost function. These parameters can be



Imagine a pit in the shape of U. You are standing at the topmost point in the pit, and your objective is to reach the bottom of the pit. There is a treasure, and you can only take a discrete number of steps to reach the bottom. If you decide to take one footstep at a time, you would eventually get to the bottom of the pit but, this would take a longer time. If you choose to take longer steps each time, you may get to sooner but, there is a chance that you could overshoot the bottom of the pit and not near the bottom. In the gradient descent algorithm, the number of steps you take is the learning rate, and this decides how fast the algorithm converges to the minima.



To update a_0 and a_1, we take gradients from the cost function. To find these gradients, we take partial derivatives for a_0 and a_1.

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

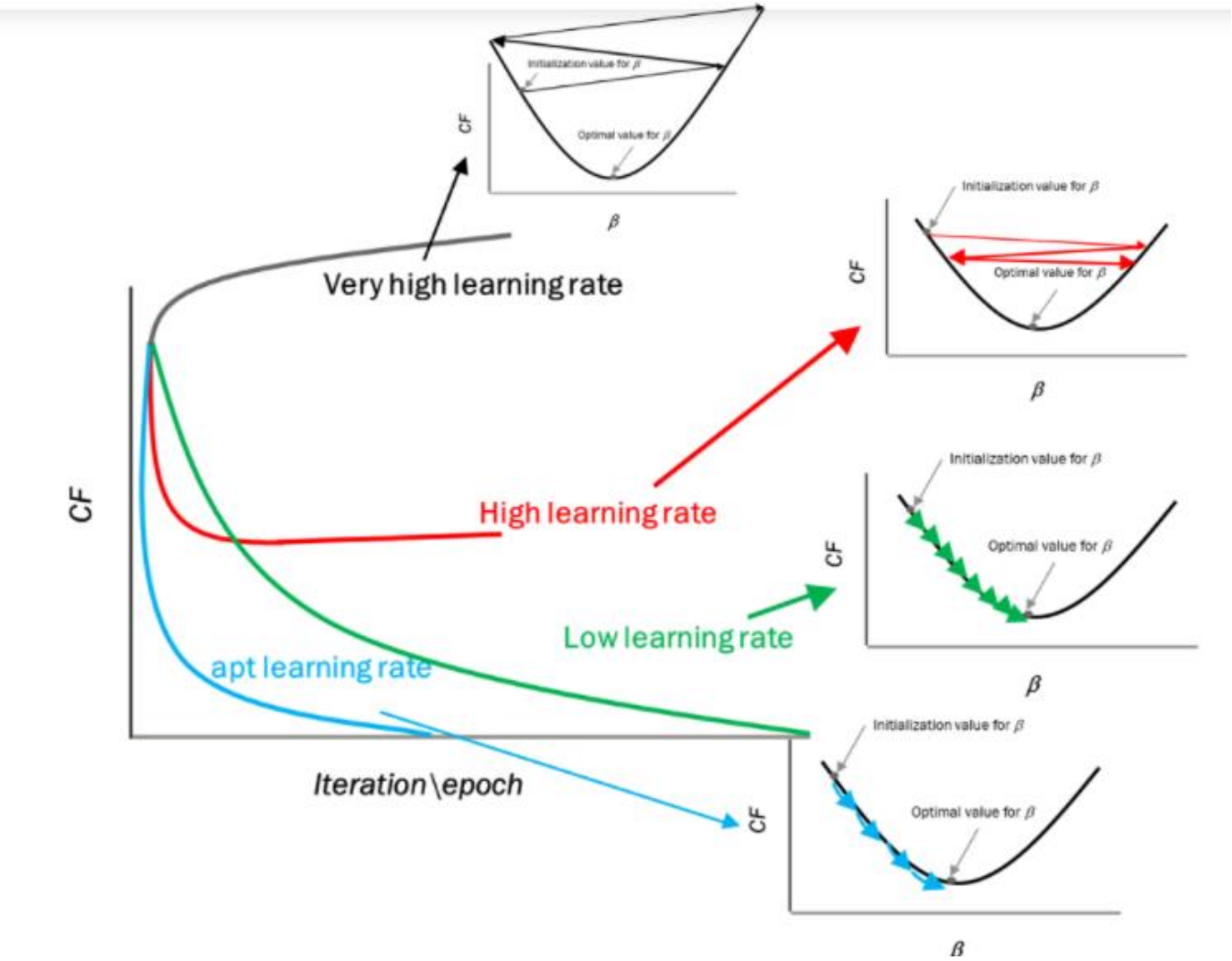
$$a_0 = a_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

The partial derivates are the gradients, and they are used to update the values of a_0 and a_1. Alpha is the learning rate.

Impact of different values for learning rate

The blue line represents the optimal value of the learning rate, and the cost function value is minimized in a few iterations. The green line represents if the learning rate is lower than the optimal value, then the number of



iterations required high to minimize the cost function. If the learning rate selected is very high, the cost function could continue to increase with iterations and saturate at a value higher than the minimum value, that represented by a red and black line.

Use case

In this, I will take random numbers for the dependent variable (salary) and an independent variable (experience) and will predict the impact of a year of experience on salary.

Steps to implement Linear regression model

import some required libraries

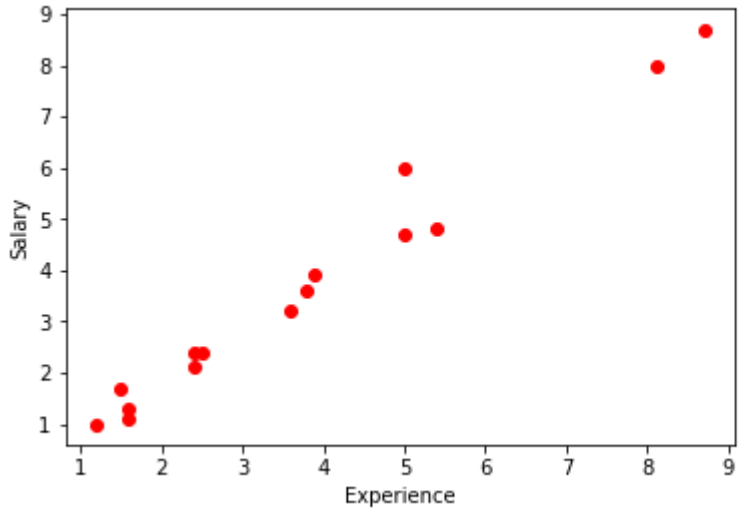
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Define the dataset

```
x= np.array([2.4,5.0,1.5,3.8,8.7,3.6,1.2,8.1,2.5,5,1.6,1.6,2.4,3.9,5.4])
y = np.array([2.1,4.7,1.7,3.6,8.7,3.2,1.0,8.0,2.4,6,1.1,1.3,2.4,3.9,4.8])
n = np.size(x)
```

Plot the data points

```
plt.scatter(experience,salary, color = 'red')
plt.xlabel("Experience")
plt.ylabel("Salary")
plt.show()
```



The main function to calculate values of coefficients

Initialize the parameters.

Predict the value of a dependent variable by given an independent variable.

Calculate the error in prediction for all data points.

Calculate partial derivative w.r.t a0 and a1.

Calculate the cost for each number and add them.

Update the values of a0 and a1.

```
#initialize the parameters
a0 = 0                #intercept
a1 = 0                #Slop
lr = 0.0001           #Learning rate
iterations = 1000     # Number of iterations
error = []            # Error array to calculate cost for each iterations.
for itr in range(iterations):
    error_cost = 0
    cost_a0 = 0
    cost_a1 = 0
    for i in range(len(experience)):
        y_pred = a0+a1*experience[i]   # predict value for given x
        error_cost = error_cost +(salary[i]-y_pred)**2
        for j in range(len(experience)):
            partial_wrt_a0 = -2 *(salary[j] - (a0 + a1*experience[j]))              #partial derivative w.r.t a0
            partial_wrt_a1 = (-2*experience[j])*(salary[j]-(a0 + a1*experience[j]))   #partial derivative w.r.t a1
            cost_a0 = cost_a0 + partial_wrt_a0      #calculate cost for each number and add
            cost_a1 = cost_a1 + partial_wrt_a1      #calculate cost for each number and add
```

```
    a0 = a0 - lr * cost_a0    #update a0
    a1 = a1 - lr * cost_a1    #update a1
    print(itr,a0,a1)         #Check iteration and updated a0 and a1
  error.append(error_cost)      #Append the data in array
```

```
51 -0.2100587036075669 1.0240594725158565
51 -0.210069416639929 1.0240604165874214
51 -0.2100827665464727 1.0240617279107738
51 -0.21009872814788516 1.024063481908892
51 -0.2101172734497008 1.0240657579772252
51 -0.21013837262662904 1.0240686345668573
51 -0.21016199500166557 1.0240721843016172
51 -0.21018810996167744 1.0240764694231033
51 -0.21021668775506228 1.0240815378378745
51 -0.21024770012426341 1.02408742000484
51 -0.21028112073594665 1.0240941268503343
51 -0.21031692538390484 1.0241016488365344
51 -0.21035509195351762 1.0241099562394875
52 -0.21035743358141284 1.024110693217287
52 -0.21036211816964787 1.0241121510338222
52 -0.2103691481960975 1.0241142983610119
52 -0.2103785269213727 1.024117090524692
52 -0.21039025787243382 1.024120472129565
52 -0.2104043442038269 1.0241243803082456
52 -0.2104207879753039? 1.024128748398941
```

At approximate iteration 50- 60, we got the value of a0 and a1.

print(a0)

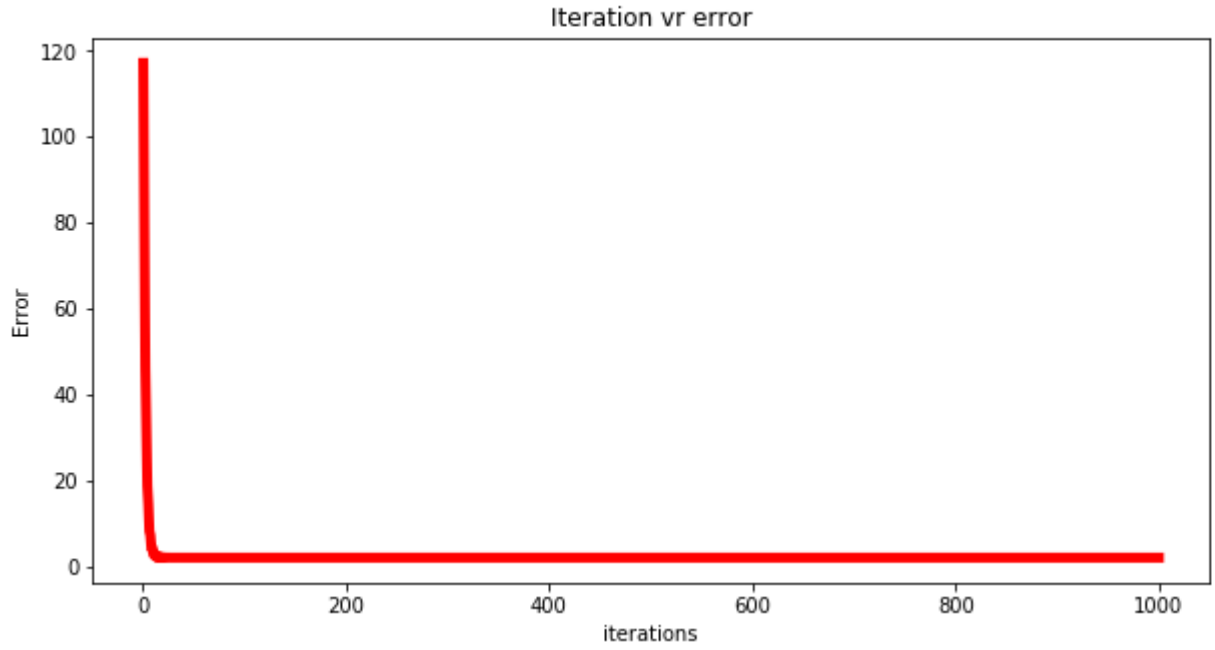
print(a1)

```
-0.21354150071690242
1.0247464287610857
```

Plotting the error for each iteration.

```
plt.figure(figsize=(10,5))
plt.plot(np.arange(1,len(error)+1),error,color='red',linewidth = 5)
plt.title("Iteration vr error")
plt.xlabel("iterations")
plt.ylabel("Error")
```

```
Text(0, 0.5, 'Error')
```



Predicting the values.

pred = a0+a1*experience

print(pred)

```
[2.24584993 4.91019064 1.32357814 3.68049493 8.70175243 3.47554564
 1.01615421 8.08690457 2.34832457 4.91019064 1.42605279 1.42605279
 2.24584993 3.78296957 5.32008921]
```

Plot the regression line.

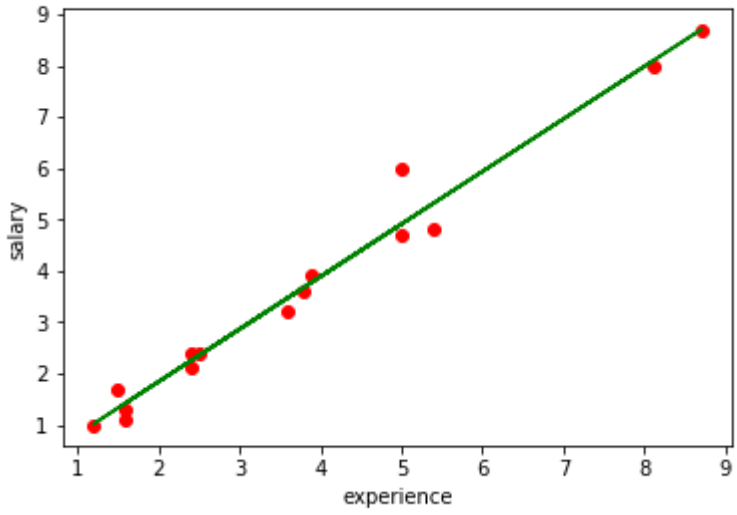
plt.scatter(experience,salary,color = 'red')

plt.plot(experience,pred, color = 'green')

plt.xlabel("experience")

plt.ylabel("salary")

```
Text(0, 0.5, 'salary')
```



Analyze the performance of the model by calculating the mean squared error.

```
error1 = salary - pred
se = np.sum(error1 ** 2)
mse = se/n
print("mean squared error is", mse)
```

```
mean squared error is 0.12785817711928918
```

Use the scikit library to confirm the above steps.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
experience = experience.reshape(-1,1)
model = LinearRegression()
model.fit(experience,salary)
salary_pred = model.predict(experience)
Mse = mean_squared_error(salary, salary_pred)
print('slop', model.coef_)
print("Intercept", model.intercept_)
print("MSE", Mse)
```

```
slop [1.02474643]
Intercept -0.2135415007169037
MSE 0.1278581771192891
```

Q.) Explain the Anscombe's quartet in detail.

Answer: Anscombe's quartet comprises four datasets that have nearly identical simple statistical properties, yet appear very different when graphed. Each dataset consists of eleven (x,y) points. They were constructed in 1973 by the statistician Francis Anscombe to demonstrate both the importance of graphing data before analyzing it and the effect of outliers on statistical properties.

Simple understanding:

Once Francis John "Frank" Anscombe who was a statistician of great repute found 4 sets of 11 data-points in his dream and requested the council as his last wish to plot those points. Those 4 sets of 11 data-points are given below.

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

After that, the council analyzed them using only descriptive statistics and found the mean, standard deviation, and correlation between x and y.

Q.) What is Pearson's R?

Answer: In Statistics, the Pearson's Correlation Coefficient is also referred to as **Pearson's r, the Pearson product-moment correlation coefficient (PPMCC), or bivariate correlation**. It is a statistic that measures the linear correlation between two variables. Like all correlations, it also has a numerical value that lies between -1.0 and +1.0.

Whenever we discuss correlation in statistics, it is generally Pearson's correlation coefficient. However, it cannot capture nonlinear relationships between two variables and cannot differentiate between dependent and independent variables.

Pearson's correlation coefficient is the covariance of the two variables divided by the product of their standard deviations. The form of the definition involves a "product moment", that is, the mean (the first moment about the origin) of the product of the mean-adjusted random variables; hence the modifier product-moment in the name.

Pearson's Correlation Coefficient is named after Karl Pearson. He formulated the correlation coefficient from a related idea by Francis Galton in the 1880s.

How is the Correlation coefficient calculated?

Using the formula proposed by Karl Pearson, we can calculate a **linear relationship** between the two given variables. For example, a child's height increases with his increasing age (different factors affect this biological change). So, we can calculate the relationship between these two variables by obtaining the value of Pearson's Correlation Coefficient r. There are certain requirements for Pearson's Correlation Coefficient:

Scale of measurement should be interval or ratio

Variables should be approximately normally distributed

The association should be linear

There should be no outliers in the data

The formula given is:

$$r = \frac{N\Sigma xy-(\Sigma x)(\Sigma y)}{\sqrt{[N\Sigma x^2-(\Sigma x)^2][N\Sigma y^2-(\Sigma y)^2]}}$$

Where,

N = the number of pairs of scores

Σxy = the sum of the products of paired scores

Σx = the sum of x scores

Σy = the sum of y scores

Σx2 = the sum of squared x scores

Σy2 = the sum of squared y scores

Some steps are needed to be followed:

Step 1: Make a Pearson correlation coefficient table. Make a data chart using the two variables and name them as X and Y. Add three additional columns for the values of XY, X^2, and Y^2. Refer to this table.

Person	Age (X)	Income (Y)	XY	X^2	Y^2
1					
2					
3					
4					

Step 2: Use basic multiplications to complete the table.

Person	Age (X)	Income (Y)	XY	X^2	Y^2
1	20	1500	30000	400	2250000
2	30	3000	90000	900	9000000
3	40	5000	200000	1600	25000000
4	50	7500	375000	2500	56250000

Step 3: Add up all the columns from bottom to top.

Person	Age (X)	Income (Y)	XY	X^2	Y^2
1	20	1500	30000	400	2250000
2	30	3000	90000	900	9000000
3	40	5000	200000	1600	25000000
4	50	7500	375000	2500	56250000
Total	140	17000	695000	5400	92500000

Step 4: Use these values in the formula to obtain the value of r.

r = [4 * 695000 - 140 * 17000] / √{4 * 5400 - (140)^2} {4 * 92500000 - (17000)^2}
= [2780000 - 2380000] / √{21600 - 19600} {370000000 - 289000000}
= 400000 / √{2000} {81000000}
= 400000 / √162000000000
= 400000 / 402492.24
= 0.99

The positive value of Pearson's correlation coefficient implies that if we change either of these variables, there will be a **positive effect** on the other. For example, if we increase the age there will be an increase in the income.

Referred blog: 4 types of Elasticity in Economics

Determining the strength of the Pearson product-moment correlation coefficient

As we have learned from the definition of the Pearson product-moment correlation coefficient, it measures the strength and direction of the linear relationship between two variables.

The more inclined the value of the Pearson correlation coefficient to -1 and 1, the stronger the association between the two variables.

Below, we have shown the guidelines to interpret the Pearson coefficient correlation:

	Coefficient, r	
Strength of Association	Positive	Negative
Small	.1 to .3	-0.1 to -0.3
Medium	.3 to .5	-0.3 to -0.5
Large	.5 to 1.0	-0.5 to 1.0

A notable point is that the strength of association of the variables depend on the sample size and what you measure.

Q.) What is scaling? Why is scaling performed? What is the difference between normalized scaling and standardized scaling?

Answer: It is a step of data Pre-Processing which is applied to independent variables to normalize the data within a particular range. It also helps in speeding up the calculations in an algorithm.

Why?

Most of the times, collected data set contains features highly varying in magnitudes, units and range. If scaling is not done then algorithm only takes magnitude in account and not units hence incorrect modelling. To solve this issue, we have to do scaling to bring all the variables to the same level of magnitude.

It is important to note that **scaling just affects the coefficients** and none of the other parameters like **t-statistic, F-statistic, p-values, R-squared**, etc.

Normalization/Min-Max Scaling:

It brings all of the data in the range of 0 and 1. **sklearn.preprocessing.MinMaxScaler** helps to implement normalization in python.

$$\text{MinMax Scaling: } x = \frac{x - min(x)}{max(x) - min(x)}$$

Standardization Scaling:

Standardization replaces the values by their Z scores. It brings all of the data into a standard normal distribution which has mean (**μ)** zero and standard deviation one (**σ**).

[redacted]

$$\text{Standardisation: } x = \frac{x - mean(x)}{sd(x)}$$

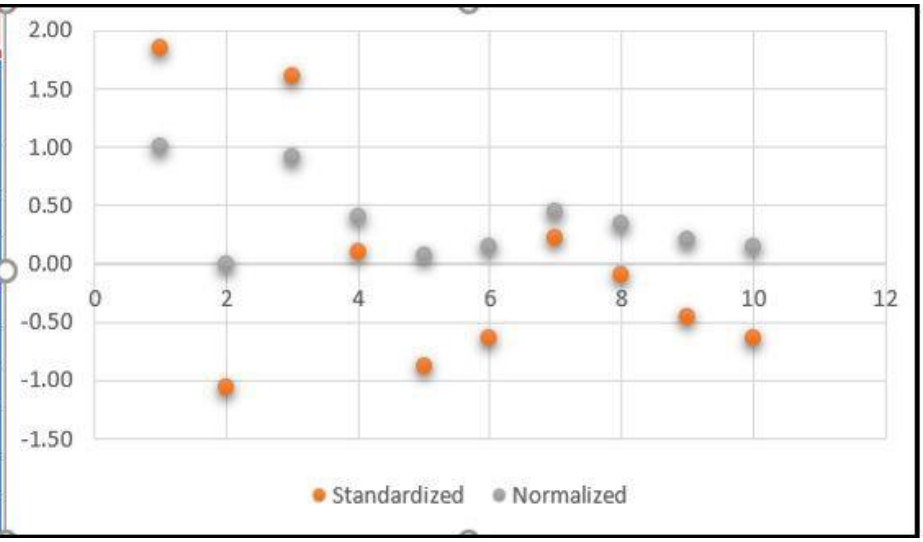
sklearn.preprocessing.scale helps to implement standardization in python.

One disadvantage of normalization over standardization is that it **loses** some information in the data, especially about **outliers**.

Example:

Below shows example of Standardized and Normalized scaling on original values.

Original	Standardized	Normalized
59	1.85	1.000
11	-1.06	0.000
55	1.61	0.917
30	0.09	0.396
14	-0.88	0.063
18	-0.64	0.146
32	0.21	0.438
27	-0.09	0.333
21	-0.46	0.208
18	-0.64	0.146



Q.) You might have observed that sometimes the value of VIF is infinite. Why does this happen?

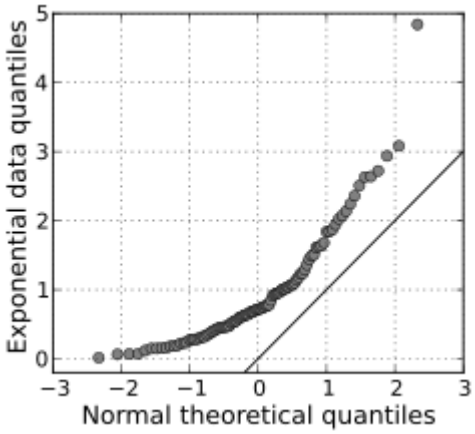
Answer: If there is perfect correlation, then VIF = infinity. This shows a perfect correlation between two independent variables. In the case of perfect correlation, we get R2 =1, which lead to 1/(1-R2) infinity. To solve this problem we need to drop one of the variables from the dataset which is causing this perfect multicollinearity.

An infinite VIF value indicates that the corresponding variable may be expressed exactly by a linear combination of other variables (which show an infinite VIF as well).

Q.) What is a Q-Q plot? Explain the use and importance of a Q-Q plot in linear regression.

Answer: Q-Q Plots (Quantile-Quantile plots) are plots of two quantiles against each other. A quantile is a fraction where certain values fall below that quantile. For example, the median is a quantile where 50% of the data fall below that point and 50% lie above it. The purpose of Q Q plots is to find out if two sets of data come from the same distribution. A 45 degree angle is plotted on the Q Q plot; if the two data sets come from a common distribution, the points will fall on that reference line.

A Q Q plot showing the 45 degree reference line:



If the two distributions being compared are similar, the points in the Q–Q plot will approximately lie on the line y = x. If the distributions are linearly related, the points in the Q–Q plot will approximately lie on a line, but not necessarily on the line y = x. Q–Q plots can also be used as a graphical means of estimating parameters in a location-scale family of distributions.

A Q–Q plot is used to compare the shapes of distributions, providing a graphical view of how properties such as location, scale, and skewness are similar or different in the two distributions.