# Student Detials

> Title= "Mr"\ Name= "Ali Nawaz"\ email = "nawazktk99@gmail.com"\ whatsapp = "03358043653"

## In this Notebook I am are going to Explore Least Squre Error

# What Is the Least Squares Method?

The least-squares method is a form of mathematical regression analysis used to determine the line of best fit for a set of data, providing a visual demonstration of the relationship between the data points. Each point of data represents the relationship between a known independent variable and an unknown dependent variable.

# What is the Least Squares Regression Method?

The least-squares regression method is a technique commonly used in Regression Analysis. It is a mathematical method used to find the best fit line that represents the relationship between an independent and dependent variable.

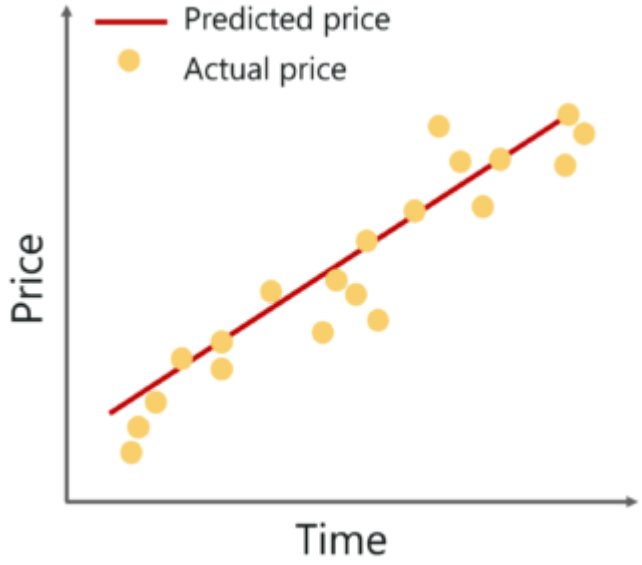
To understand the least-squares regression method lets get familiar with the concepts involved in formulating the line of best fit.

# What is the Line Of Best Fit?

Line of best fit is drawn to represent the relationship between 2 or more variables. To be more specific, the best fit line is drawn across a scatter plot of data points in order to represent a relationship between those data points.

Regression analysis makes use of mathematical methods such as least squares to obtain a definite relationship between the predictor variable (s) and the target variable. The least-squares method is one of the most effective ways used to draw the line of best fit. It is based on the idea that the square of the errors obtained must be minimized to the most possible extent and hence the name least squares method.

If we were to plot the best fit line that shows the depicts the sales of a company over a period of time, it would look something like this:




In [ ]:
```
# import all the lib
import pandas as pd
import seaborn as sns
import numpy as np
import sklearn.metrics as sm
import matplotlib.pyplot as plt
```

In [ ]:
```
# Reading Data
data = pd.read_csv('D:/Python ka Chilla/python_chilla/data/ml_data_salary.csv')
# data = pd.read_csv('D:/Python ka Chilla/python_chilla/data/mldata.csv')


print(data.shape)
data.head()
```

```
(30, 4)
```

Out[ ]:

| | age | distance | YearsExperience | Salary |
|---|---|---|---|---|
| **0** | 31.1 | 77.75 | 1.1 | 39343 |
| **1** | 31.3 | 78.25 | 1.3 | 46205 |
| **2** | 31.5 | 78.75 | 1.5 | 37731 |
| **3** | 32.0 | 80.00 | 2.0 | 43525 |
| **4** | 32.2 | 80.50 | 2.2 | 39891 |

In [ ]:
```
 #split dataset in features and target variable
# data['gender'] = data['gender'].replace('Male',  1)
# data['gender'] = data['gender'].replace('Female',  0)
feature_cols = ['age', ]#'gender', 'weight'
X = data[feature_cols].values # Features
Y = data.YearsExperience.values # Target variable
```

In [ ]:
```
# Mean X and Y
```

```
mean_x = np.mean(X)
mean_y = np.mean(Y)

# Total number of values
n = len(X)
```

In [ ]:

```
print(mean_x)
print(mean_y)
print(n)
```

```
35.31333333333334
5.3133333333333335
30
```

# Calculate the values of the slope and y-intercept

In [ ]:

```
# Using the formula to calculate 'm' and 'c'
numer = 0
denom = 0
for i in range(n):
    numer = (X[i] - mean_x) * (Y[i] - mean_y)
    numer = numer + 1
    denom = (X[i] - mean_x) ** 2
    denom = denom + 1

m = numer / denom
c = mean_y - (m * mean_x)

# Printing coefficients
print("Coefficients")
print(m, c)
```

```
Coefficients
[1.] [-30.]
```

# Plotting the line of best fit

In [ ]:

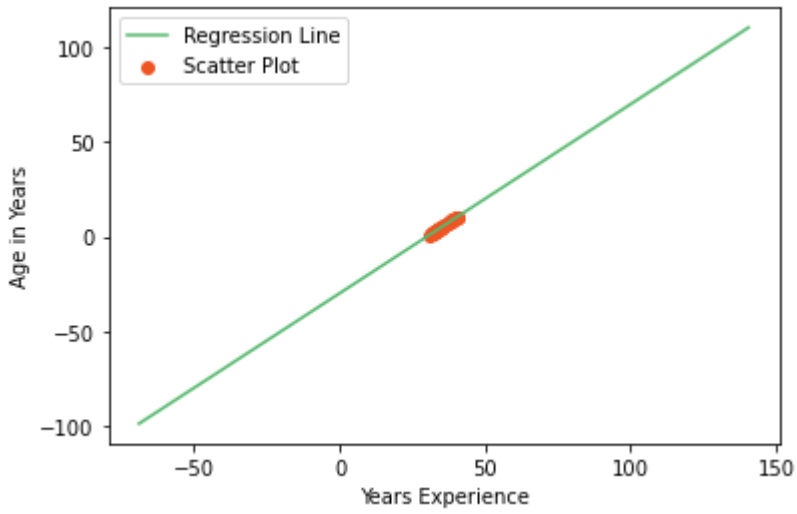
```
print(X.shape)
print(Y.shape)
```

```
(30, 1)
(30,)
```

In [ ]:

```
# Plotting Values and Regression Line

max_x = np.max(X) + 100
min_x = np.min(X) - 100

# Calculating line values x and y
x = np.linspace(min_x, max_x, 1000)
y = c + m * x

# Ploting Line
plt.plot(x, y, color='#58b970', label='Regression Line')
# Ploting Scatter Points
```

```
plt.scatter(X, Y, c='#ef5423', label='Scatter Plot')

plt.xlabel('Years Experience')
plt.ylabel('Age in Years')
plt.legend()
plt.show()
```




In [ ]:

```
# Calculating Root Mean Squares Error
rmse = 0
for i in range(n):
    y_pred = c + m * X[i]
    rmse += (Y[i] - y_pred) ** 2
rmse = np.sqrt(rmse/n)
print("RMSE")
print(rmse)
```

```
RMSE
[1.13849929e-14]
```

# Calculating R2 Score

In [ ]:

```
# Calculating R2 Score
ss_tot = 0
ss_res = 0
for i in range(n):
    y_pred = c + m * X[i]
    ss_tot += (Y[i] - mean_y) ** 2
    ss_res += (Y[i] - y_pred) ** 2
r2 = 1 - (ss_res/ss_tot)
print("R2 Score")
print(r2)
```

```
R2 Score
[1.]
```

In [ ]: