What is Regression analysis?

-> Regression im statstics is te process of predicting a label(or Dependet variable) based on the features(Independent variable).

-->Regression analysis is a **form of predictive modelling techique which investigates the realashion ship between a dependent (target) and independent variable s(predictor).**

--> Here, we fit a curve/ line to the data points ,in such a manner that the differences between the distances of data points from the curve or line is minimized.

-->This tecqunique is used for forecasting and finding the relationship between the variables.

The use of Regression:

-->Regression analysis predict the relationship between two or more features.

- The benifit of using regresion analysis are as follows:

```
  --> it shows the significant relationships between the label
and the features.

  --> it indicates th strength of impact of multiple independent
variabales o dependent variables
```

These benefits help market researchers / data analyst / data scientists to eliminat and evaluate the best set of variables to be used for building predictive models.

Linear Regression:

*Regression models are used to predict the continous values.

*It is supervised techmique.

*Linear Regression is a **stastical analysis for predicting the values of a quantitative variable**.

- In this technique , **the dependent variable is continous ,independent variable can be contionous or discrete , and nature of regression line is linear.**

--> Linear reression establishes a relationship between dependent variable(Y) and one or more independent variables (X)inpendent variables(X) usin a best fit straight line(also known as regression line).

Y=m*X+b+e

where b is intercept of y

m is the slope of the line

e is the error term

Problem Statement:

Predict the salary based on Experience:

The senerio is you are a HR officer, you got a candidate with 5 years of experience .Then what is the best salary you shoud offer him.

Loading basic libraries

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

Load data set

```
# we are loading salay_data.csv
df=pd.read_csv("salary_data.csv")
```

Basic Information of data :

checking shape(Row,columns) of the DataFrame

```
df.shape
```

```
(30, 2)
```

```
df.head()  # prints first  5  rows of the DataFrame
```

```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
```

```
df.tail() # prints last five rows of the DataFrame
```

```
    YearsExperience    Salary
25              9.0  105582.0
26              9.5  116969.0
27              9.6  112635.0
28             10.3  122391.0
29             10.5  121872.0
```

```
#if suppose i want view random rows then we have to use sample method
df.sample(10)
```

```
    YearsExperience    Salary
7               3.2   54445.0
3               2.0   43525.0
22              7.9  101302.0
17              5.3   83088.0
27              9.6  112635.0
26              9.5  116969.0
4               2.2   39891.0
11              4.0   55794.0
18              5.9   81363.0
28             10.3  122391.0
```

Checking the Data types of features:

```
df.dtypes
```

```
YearsExperience    float64
Salary             float64
dtype: object
```

Describing the DataFrame in statistically

```
df.describe()
```

```
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000
```

Information of Datafrmae:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Data Cleaning:

Dropping the duplicates:

```
# checking the number of rows and columns before removing duplicates:
df.shape
```

```
(30, 2)
```

```
#using drop_duplicates on DataFrame to remove duplicates and assign it
back to DataFrame
df=df.drop_duplicates()
```

```
df.shape
```

```
(30, 2)
```

So after romoving duplicates we having same no of rows and columns it means that there are no duplicates in the dataframe

Check the null vaues

```
df.isnull().sum()
```

```
YearsExperience    0
Salary             0
dtype: int64
```

There are no null vlues present in dataframe

Create a Dependent(y) and independent(X) variable:

```
target_feature='Salary'

# Seperate object for target variable(dependent variable)
y=df[target_feature]

# Seperate object for input etures
X=df.drop(target_feature,axis=1)
```

```
X.shape
```

```
(30, 1)
```

```
X.head()
```

```
   YearsExperience
0              1.1
1              1.3
2              1.5
3              2.0
4              2.2
```

```
y.shape  # only thirty rows
```

```
(30,)
```

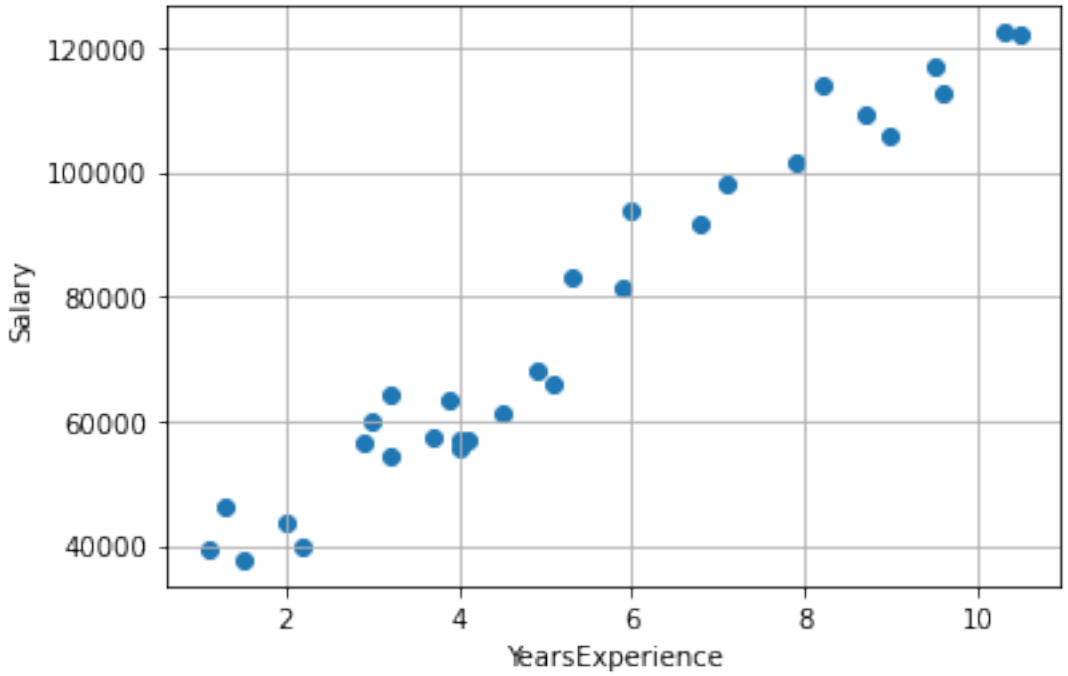
```
y.head()
```

```
0    39343.0
1    46205.0
2    37731.0
```

```
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

Data visualiztion before Train the Model

```
plt.scatter(X,y)
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.grid()
plt.show()
```



Split Dataset into Train and Test

```
# to split dataset into train and test we have to import sklean
library with model selection
from sklearn.model_selection import train_test_split

#splitting the data into train and test with test size 20%
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,rando
m_state=0)

x_train.shape,y_train.shape # 80% of train data takes 24 rows

((24, 1), (24,))

x_test.shape,y_test.shape  #20% of test data takes 6 rows

((6, 1), (6,))
```

Apply linear Regression on Train Dataset:

```
# to perform linear regression first we have to import linear
regression algorithm from sklearn model.linra_model
from sklearn.linear_model import LinearRegression

reg=LinearRegression()  # our training model which will implement the
Linear Regression

reg.fit(x_train,y_train)# train the train data with fit method

LinearRegression()
```

Geting the paramters

```
reg.intercept_
print('intercept(b) is :',reg.intercept_)

intercept(b) is : 26780.099150628186

reg.coef_
print('coefficient (m) is :',reg.coef_)

coefficient (m) is : [9312.57512673]
```

Apply the model on test Dataset to get the predicted values

```
y_pred=reg.predict(x_test)
y_pred

array([ 40748.96184072, 122699.62295594,  64961.65717022,
63099.14214487,
       115249.56285456, 107799.50275317])

y_pred.shape

(6,)
```

Comparing the actual output values with the predicted values:

```
df1=pd.DataFrame({'Actual': y_test, 'Predicted': y_pred,'varaince':
(y_test-y_pred)})

df1
```

	Actual	Predicted	varaince
2	37731.0	40748.961841	-3017.961841
28	122391.0	122699.622956	-308.622956
13	57081.0	64961.657170	-7880.657170
10	63218.0	63099.142145	118.857855
26	116969.0	115249.562855	1719.437145
24	109431.0	107799.502753	1631.497247

Prediction

```
# predicting the result of 1.5 years of experience
pred=np.array([1.5]).reshape(-1,1)
```

```
reg.predict(pred)
```

```
array([40748.96184072])
```

```
# predicting the result of 1.5 years of experience mathemaically
# y=m*x+b [ m is coefficent of x i.e. slope and value is 9312.57512673
,   b is intercept of y and value is 26780.099150628186}
# and given that X is 1.5
9312.57512673*1.5+26780.099150628186
```

```
40748.96184072319
```

visualization

visualize our training model:

```
plt.scatter(x_train,y_train,color='r')
plt.plot(x_train,reg.predict(x_train),color='g')
plt.title("Salary vs Experince(Training Set)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.grid()
plt.show()
```

visualize test set results

```
plt.scatter(x_test,y_test,color='r')
plt.plot(x_train,reg.predict(x_train),color='g')
plt.title("Salary vs Experince(Test Set)")
plt.xlabel("Years of Experience")
```

```
plt.ylabel("Salary")
plt.grid()
plt.show()
```

Evaluation metrics of regression algorithms

```
from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred)*100
print('score: ',score)
```

```
score:  98.8169515729126
```

```
from sklearn import metrics
print("Mean Absolute
Error:",metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared erroe:",metrics.mean_squared_error(y_test,y_pred))
print("Root Mean squared error:
",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error: 2446.1723690465064
Mean Squared erroe: 12823412.298126562
Root Mean squared error:  3580.979237321345
```

END