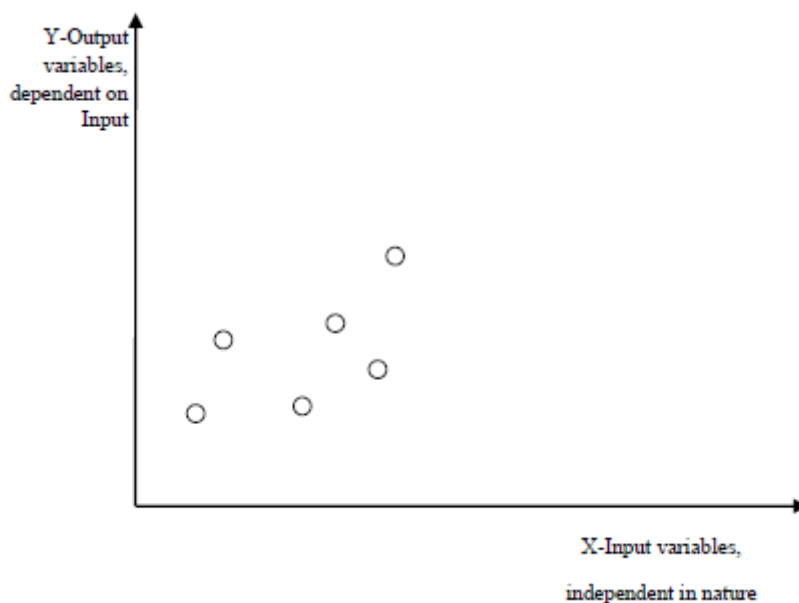


CONTENTS

S.No	Topic
2.1	Introduction to Regression
2.2	Math for Regression
2.3	Regression Types
2.4	Simple Linear Regression
2.5	Evaluation Metrics – MSE, MAE, RMSE, R2
2.6	Multiple Linear Regression
2.7	Stepwise Regression
2.8	Polynomial Regression
2.9	Shrinkage Methods - Regularization
2.10	Introduction to Classification
2.11	Logistic Regression
2.12	Evaluation Metrics- Confusion Matrix, Accuracy, Precision, Recall, AUC
2.13	KNN Classification
2.14	Naïve Bayes Classification

2.1 Introduction to Regression

Regression is a broadly used statistical and machine learning tool. The key objective of regression-based tasks is to **predict** output labels or responses which are **continuous numeric values**, for the given input data. The output will be based on what the model has learned in training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific association between inputs and corresponding outputs. Regression analysis estimates the relationship between two or more variables. It indicates the significant relationships between dependent variable and independent variable. It indicates the strength of impact of multiple independent variables on a dependent variable.



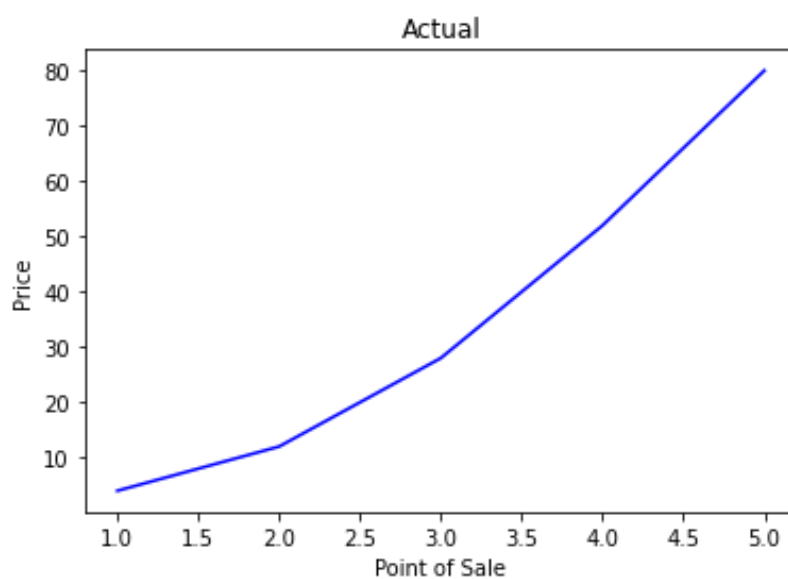
2.2 Math for Regression

Let's take a real world example of the price of agricultural products and how it varies based on the location its sold. The price will be low when bought directly from farmers and high when brought from the downtown area.

Given this dataset, we can predict the price of the product in intermediate locations.

Agricultural Product @ Point of Sale (X)	Price (Y)
Farmer (1)	4
Village (2)	12
Town (3)	28
City (4)	52
Metro (5)	80

If we plot the data points we get the following graph.



The data we take for learning is called the training data. Our aim is to come with a straight line which minimizes the error between training data and our prediction model when we draw the line using the equation of straight line.

Equation of Straight Line is $y = mx + b$, Where m is the slope(Gradient) and b is the y-intercept (bias)

We have 2 unknown quantities, m and b

Slope m = Change in y / Change in x

$$= \frac{y_2 - y_1}{x_2 - x_1}$$

Also, represented as **rise/run**

Let us take the points (farmer and the metro data)

$$(x_1, y_1) = (1, 4)$$

$$(x_2, y_2) = (5, 80)$$

$$\text{Given } (x_1, y_1) = (1, 4) \text{ and } (x_2, y_2) = (5, 80)$$

$$m = \text{Change in Y} / \text{Change in X}$$

$$m = (y_2 - y_1) / (x_2 - x_1)$$

$$m = (80 - 4) / (5 - 1)$$

$$m = 76 / 4$$

$$m = 19$$

The y-intercept / bias shall be calculated using the formula

$$Y - Y_1 = m(x - x_1)$$

$$\text{Given } m = 19 \text{ and } (x_1, y_1) = (1, 4)$$

$$y - y_1 = m(x - x_1)$$

$$y - 4 = 19(x - 1)$$

$$y - 4 = 19x - 19$$

$$y = 19x - 19 + 4$$

$$y = 19x - 15$$

This can be written in the form of $y = mx + b$ as

$$y = 19x + (-15), \text{ so } b = -15$$

Once we arrived at our formula, we can verify the same by substituting x for both starting and ending points which we used to calculate the formula as it should provide the same y value.

$$\text{Given Formula } \{ y = 19x + (-15) \}$$

$$x_1 \Rightarrow 1 \quad \{ y = 19 * 1 - 15 \Rightarrow 19 - 15 \Rightarrow 4 \text{ (i.e. } y_1 \text{)} \}$$

$$x_2 \Rightarrow 5 \quad \{ y = 19 * 5 - 15 \Rightarrow 95 - 15 \Rightarrow 80 \text{ (i.e. } y_2 \text{)} \}$$

Now we know that our formula is correct as we get the same y value by substituting the x value, but what about other values of x in between i.e 2,3,4 , let's find out

Given Formula $\{y = 19x + (-15)\}$

$x \Rightarrow 2$ $\{y = 19 * 2 - 15 \Rightarrow 38 - 15 \Rightarrow 23\}$

$x \Rightarrow 3$ $\{y = 19 * 3 - 15 \Rightarrow 57 - 15 \Rightarrow 42\}$

$x \Rightarrow 4$ $\{y = 19 * 4 - 15 \Rightarrow 76 - 15 \Rightarrow 61\}$

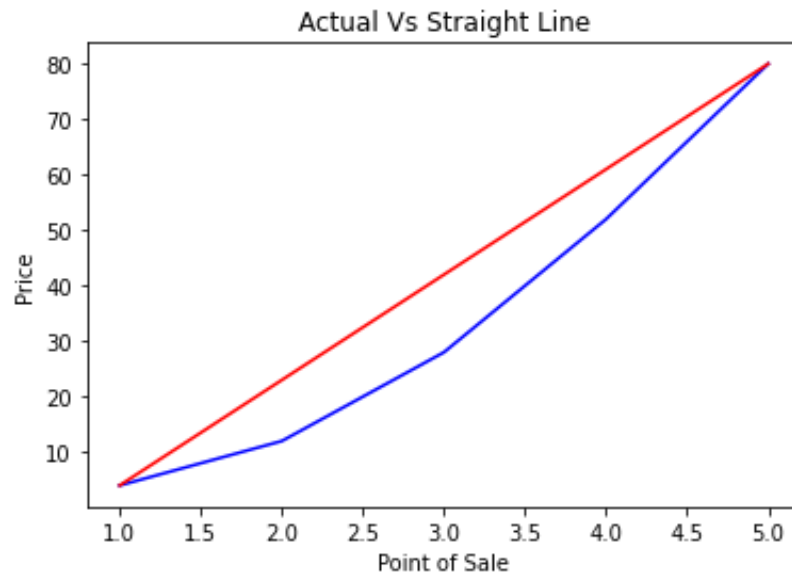
Residuals

Residuals are the difference between the actual value of y and the predicted values of y which is \hat{y} . It signifies the error caused by drawing the linear regression line. $e = y - \hat{y}$

For the given example

Actual y	Predicted \hat{y}	Residual $y - \hat{y}$
4	4	0
12	23	-11
28	42	-14
52	61	-9
80	80	0

Plotting the graph for the original values and the straight line we have got thro' calculation



Minimizing the Error

- The error is defined as the difference of values between actual points and the points on the straight line
- Ideally, we'd like to have a straight line where the error is minimized across all points
- *There are many mathematical ways to do the same and one of the methods is called Least Square Regression*

Least Square Regression

- It is also called as Ordinary Least Square Method / OLS method
- Least Square Regression is a method which minimizes the error in such a way that the sum of all square error is minimized.

$$m = \frac{\text{Sum of ALL } (x - x_{\text{mean}}) * (y - y_{\text{Mean}})}{\text{Sum of } (x - x_{\text{mean}})^2}$$

To Calculate $\Rightarrow (x - x_{\text{mean}})$

Mean Value of all 'x' $\Rightarrow (1 + 2 + 3 + 4 + 5) / 5 = 3$

At $x = 1$: $(x - x_{\text{mean}}) \Rightarrow (1 - 3) = -2$

At $x = 2$: $(x - x_{\text{mean}}) \Rightarrow (2 - 3) = -1$

At $x = 3$: $(x - x_{\text{mean}}) \Rightarrow (3 - 3) = 0$

At $x = 4$: $(x - x_{\text{mean}}) \Rightarrow (4 - 3) = 1$

At $x = 5$: $(x - x_{\text{mean}}) \Rightarrow (5 - 3) = 2$

To Calculate $\Rightarrow (y - y_{\text{mean}})$

Mean Values of all 'y' $\Rightarrow (4 + 12 + 28 + 52 + 80) / 5 = 35.2$

At $y = 1$: $(y - y_{\text{mean}}) \Rightarrow (4 - 35.2) = -31.2$

At $y = 2$: $(y - y_{\text{mean}}) \Rightarrow (12 - 35.2) = -23.2$

At $y = 3$: $(y - y_{\text{mean}}) \Rightarrow (28 - 35.2) = -7.2$

At $y = 4$: $(y - y_{\text{mean}}) \Rightarrow (52 - 35.2) = 16.8$

At $y = 5$: $(y - y_{\text{mean}}) \Rightarrow (80 - 35.2) = 44.8$

To Calculate $\Rightarrow \text{SUM OF ALL } \{ (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \}$

$(x,y) \Rightarrow (1,4) : (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \Rightarrow \{-2 * -31.2 = 62.4\}$

$(x,y) \Rightarrow (2,12) : (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \Rightarrow \{-1 * -23.2 = 23.2\}$

$(x,y) \Rightarrow (3,28) : (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \Rightarrow \{0 * -7.2 = 0\}$

$(x,y) \Rightarrow (4,52) : (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \Rightarrow \{1 * 16.8 = 16.8\}$

$(x,y) \Rightarrow (5,80) : (x - x_{\text{mean}}) * (y - y_{\text{mean}}) \Rightarrow \{2 * 44.8 = 89.6\}$

Sum of All $\Rightarrow \{62.4 + 23.2 + 0 + 16.8 + 89.6\} = 192$

To Calculate $\Rightarrow \text{Sum of } (x - x_{\text{mean}})^2$

$\{-2^2, -1^2, 0^2, 1^2, 2^2\} =$

$\{4, 1, 0, 1, 4\} =$

$\{10\}$

$$m = \frac{\text{Sum of ALL } (x - x_{\text{mean}}) * (y - y_{\text{mean}})}{\text{Sum of } (x - x_{\text{mean}})^2}$$

$$m = 192 / 10 = 19.2$$

- The y-intercept is calculated using the formula $b = y_{\text{mean}} - m * x_{\text{mean}}$

$$\text{To Calculate } \Rightarrow b = y_{\text{mean}} - m * x_{\text{mean}}$$

$$b = 35.2 - 19.2 * 3$$

$$b = 35.2 - 57.6$$

$$b = -22.4$$

- The overall formula can now be written in the form of $y = mx + b$ as

$$y = mx + b \Rightarrow \{ y = 19.2x + (-22.4) \}$$

Using Least Square Regression on X,Y values, Let's see how the prediction y changes when we apply $y = 19.2x + (-22.4)$ on all x values.

$$\text{Given Formula } \Rightarrow \{ y = 19.2x + (-22.4) \}$$

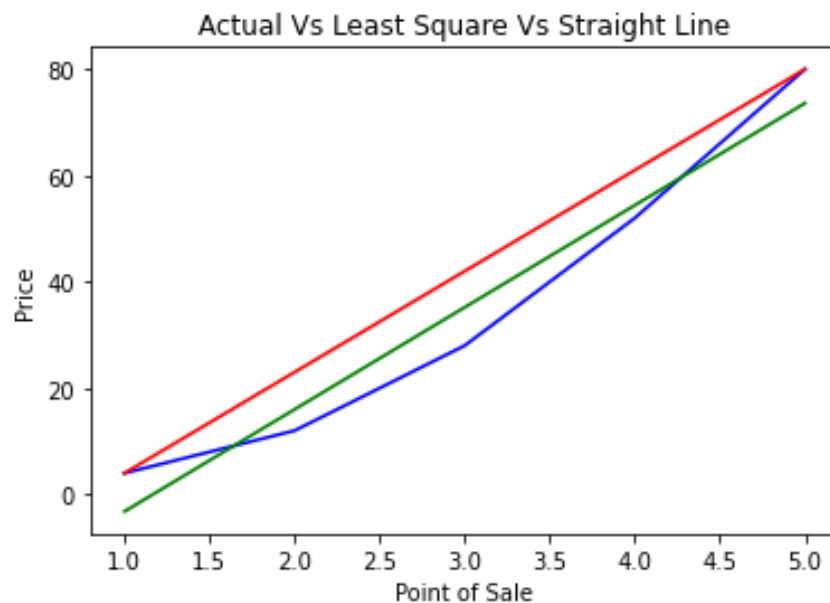
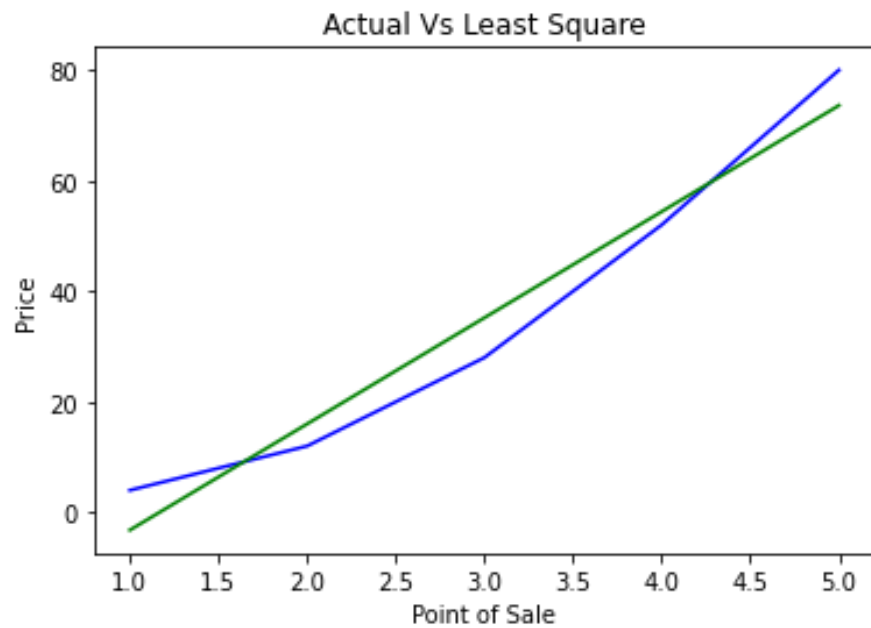
$$x \Rightarrow 1 \quad \{ y = 19.2 * 1 - 22.4 \} = -3.2$$

$$x \Rightarrow 2 \quad \{ y = 19.2 * 2 - 22.4 \} = 16$$

$$x \Rightarrow 3 \quad \{ y = 19.2 * 3 - 22.4 \} = 35.2$$

$$x \Rightarrow 4 \quad \{ y = 19.2 * 4 - 22.4 \} = 54.4$$

$$x \Rightarrow 5 \quad \{ y = 19.2 * 5 - 22.4 \} = 73.6$$



Why this method is called Least Square Regression ?

This method is intended to reduce the sum square of all error values. The lower the error, lesser the overall deviation from the original point.

We can compare the same with the errors generated out of the straight line as well as with the Least Square Regression

Error in 'Y' With Straight Line Equation

At X-Value	Y-Value	Actual-Value	Error	Square-Error
1	4	4	0	0
2	12	23	-11	121
3	28	42	-14	196
4	52	61	-9	81
5	80	80	0	0

$$\text{Sum of Square Error} = \{ 0 + 121 + 196 + 81 + 0 \} = 398$$

Error in 'Y' With Least Square Equation

At X-Value	Y-Value	Actual-Value	Error	Square-Error
1	4	-3.2	7.2	51.84
2	12	16	-4	16
3	28	35.2	-7.2	51.84
4	52	54.4	-2.4	5.76
5	80	73.6	6.4	40.96

$$\text{Sum of Square Error} = \{ 51.84 + 16 + 51.84 + 5.76 + 40.96 \} = 166.4$$

Least Square Method provide better results than a plain straight line between two points calculation.

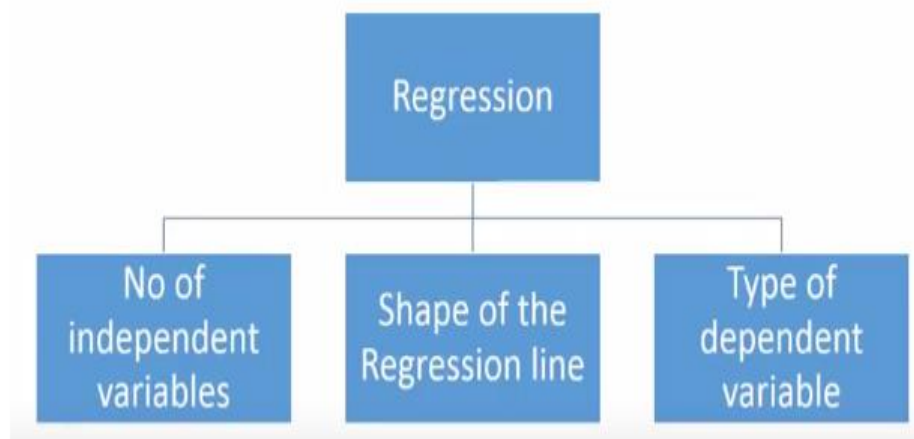
Why we have to calculate the squared errors instead of considering the residuals as such?

Both the sum and the mean of the **residuals** are equal to zero. Hence

Find the squares and sum it, it is sum of squared residuals

2.3 Regression Types

Regression Types can be based on No. of independent variables, Shape of the regression line and the Type of dependent variable



According to

No of Independent variables: If the number of independent variable is 1 then, it is simple linear regression (SLR); if it is more than 1 then it is multiple linear regression (MLR).

Shape of the Regression Line: The shape of the regression line can be linear or non-linear. SLR, MLR is a linear type whereas Polynomial regression is a non-linear type. That means, it uses the curve to fit the training data.

Type of Dependent variable: The type of the dependent variable can be continuous as in the case of SLR, MLR or polynomial regression and can be categorical as in the case of Logistic Regression.

2.4 Simple Linear Regression

When there is a single input variable, i.e. line equation is considered as $y=mx+c$, then it is Simple Linear Regression. Let us learn by an example.

The relationship between the height and weight of a person is directly proportional. A study has been performed on the volunteers to determine the height and ideal weight of the person and the values have been recorded. This will be considered as our training data set. Using the training data, a regression line equation is calculated. This linear equation is then

used for making predictions on new data. That is, if we give the height of the person, then the corresponding weight should be predicted by the model developed by us $Y(\text{pred}) = b_0 + b_1 * x$

The values b_0 is the y-intercept and b_1 is the coefficient of x . These values are chosen such that they minimize the error.

Line of Best fit

The biggest question arise here is how find the Line of Best fit. The one solution is by **Ordinary Least Square(OLS)** method.

- Find the mean of x and mean of y .
- Any line we decide, should pass through this coordinate
- Draw all possible lines
- Notice for every point on the graph, our line is wrong by some distance
- **This amount wrong is called a residual**
- Difference between the observed value of the dependent variable (y) and the predicted value (\hat{y}) is called the **residual** (e). Each data point has one **residual**.
 - **Residual** = Observed value - Predicted value. $e = y - \hat{y}$
- Both the sum and the mean of the **residuals** are equal to zero.
- Find the squares and sum it, it is sum of squared residuals
- Find the least squares to fit the regression line
- **Line of Best Fit = Lowest SS residuals**

2.5 Evaluation Metrics

Error Metric

If the sum of squared error is taken as a metric to evaluate the model, then the goal to obtain a line that best reduces the error.

$$\text{Error} = \sum_{i=1}^n (\text{actual_output} - \text{predicted_output}) ** 2$$

We are squaring out the error so that positive and negative values will not cancel out each other. For model with one predictor:

Calculation of Intercept (b_0) in the line equation is done by:

$$b_0 = \bar{y} - b_1\bar{x}$$

Calculation of the coefficient for the input value x is done by:

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Understanding the coefficient b_1 :

- If $b_1 > 0$, then x(input) and y(output) are directly proportional. That is an increase in x will increase y such as height increases, weight increases.
- If $b_1 < 0$, then x(predictor) and y(target) are inversely proportional. That is an increase in x will decrease y such as the speed of a vehicle increase, time is taken decreases.

Understanding the coefficient b_0 :

B_0 takes up the residual value for the model and ensures that the prediction is not biased. If we have not B_0 term then the line equation ($y=B_1x$) is forced to pass through origin i.e. the input and output values put into the model result in 0. But this will never be the case, if we have 0 in input, then B_0 will be average of all predicted values when $x=0$. Setting all predictor values to be 0 in the case of $x=0$ will result in data loss and is often impossible.

Root Mean Square Error(RMSE)

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

Accuracy Metrics

R Square

It is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It is also defined as coefficient of determination.

$$\text{Coefficient of Determination} \rightarrow R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Sum of Squares Total} \rightarrow SST = \sum (y - \bar{y})^2$$

$$\text{Sum of Squares Regression} \rightarrow SSR = \sum (\hat{y} - \bar{y})^2$$

$$\text{Sum of Squares Error} \rightarrow SSE = \sum (y - \hat{y})^2$$

Simple Linear Regression for House Price Prediction based on lot size

```

x=df1[['lotsize']]
y=df1['price']

##Split into Train and Test

from sklearn.model_selection import train_test_split
Xtrain,Xtest,Ytrain,Ytest=train_test_split(x,y,test_size=0.3,random_state=20)

## Build the model

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()

## Train the data

slr=regressor.fit(Xtrain,Ytrain)
print(slr.intercept_,slr.coef_)

## Predict

Ypred=slr.predict(Xtest)
Ypred

## Evaluate the model

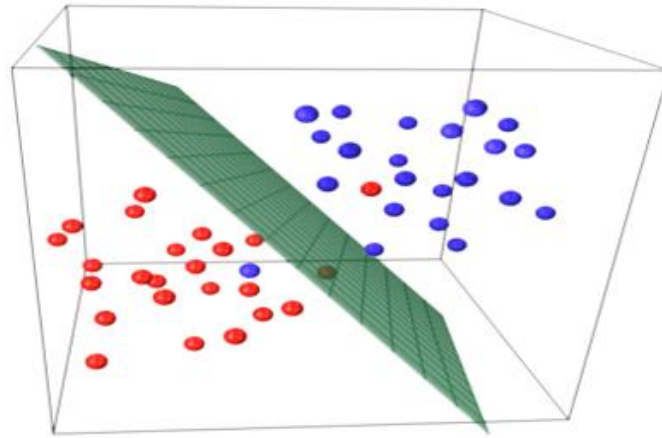
from sklearn.metrics import mean_squared_error,r2_score
import math
accuracy=r2_score(Ytest,Ypred)
error=mean_squared_error(Ytest,Ypred)
print("Accuracy=",accuracy)
print("RMSE=",math.sqrt(error))

```

2.6 Multiple Linear Regression

When there are multiple input variables i.e. line equation is considered as $y = ax_1 + bx_2 + \dots + nx_n$, then it is Multiple Linear Regression. Simple Linear Regression models are very rare in ML because generally, we will be having various input factors to determine the outcome. When there are multiple input values and one output value, then the equation formed is that of a plane or hyper-plane.

$$y = ax_1 + bx_2 + \dots + nx_n$$



The core idea in the regression model is to obtain a line equation that best fits the data. The best fit line is the one where the total prediction error for all the data points considered as small as possible. The error is the distance between the points on the plane to the regression line.

Multiple Linear Regression

```
x=df1[['lotsize','bathrms']]    ## more than one feature of X
y=df1['price']
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=20)
mlr=regressor.fit(xtrain,ytrain)
print(mlr.intercept_,mlr.coef_)
ypred=mlr.predict(xtest)
accuracy=r2_score(ytest,ypred)
error=mean_squared_error(ytest,ypred)
print("\t Accuracy=",accuracy)
print("\t RMSE=",math.sqrt(error))
```

2.7 Stepwise Regression

Stepwise regression is the step-by-step iterative construction of a regression model that involves automatic selection of independent variables.

- Stepwise regression is a method that examines the statistical significance of each independent variable within the model.

- The forward selection approach adds a variable and then tests for statistical significance.
- The backward elimination method begins with a model loaded with many variables and then removes one variable to test its importance relative to overall results.

Stepwise Regression

```
from mlxtend.feature_selection import SequentialFeatureSelector

x=df1[['lotsize','bedrooms','bathrms','stories','garagepl','driveway','recroom','fullbase','gashw','airco','prefarea']]
y=df1['price']

sfs1 = SequentialFeatureSelector(regressor,k_features = 5,forward=True,scoring='r2')
feat=sfs1.fit(x,y)
names=list(feat.k_feature_names_)
names

x_sel=df1[['lotsize', 'bathrms', 'stories', 'airco', 'prefarea']]
y=df1['price']
xtrain,xtest,ytrain,ytest=train_test_split(x_sel,y,test_size=0.3,random_state=1)

regressor = LinearRegression()
sr=regressor.fit(xtrain, ytrain)
```

- Stepwise regression has many critics, as it is approach that fits data into a model to achieve a desired result.

2.8 Polynomial Regression

This is one of the regression technique which is used by the professionals to predict the outcome. It is defined as the relationship between the independent and dependent variables when the dependent variable is related to the independent variable having an nth

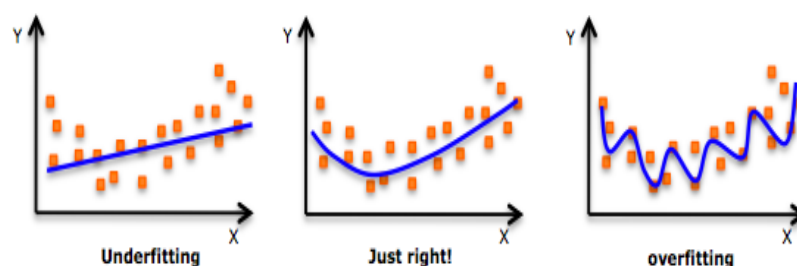
degree. It does not require the relationship between dependent and independent variables to be linear, so if the line is a curve than it may have any polynomial term.

The main difference between linear and polynomial regression is that linear regression requires the dependent and independent variables to be linearly related while this may better fit the line if we include any higher degree to the independent variable term in the equation. The equation of the polynomial regression having an nth degree can be written as:

$$Y = b_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

If we add higher degrees such as quadratic, then it turns the line into a curve that better fits the data. Generally, it is used when the points in the data set are scattered and the linear model is not able to describe the result clearly. We should always keep an eye on Overfitting and Underfitting while considering these degrees to the equation.

It is better to consider the degree that passes through all the data points but sometimes taking higher degree such as 10 or 20 may pass through all the data points and reduce the error but it also captures the noise of the data which is overfitting the model and it can be avoided by adding more samples to the training data set. So, it is always advisable to choose an optimal degree to fit the model.



There are two techniques which are used in deciding the degree of the equation:

Forward Selection: It is the method of increasing the degree until it is significant enough to define the model.

Backward Selection: It is the method of decreasing the degree until it is significant enough to define the model.

Advantages of using Polynomial Regression

- Broad range of function can be fit under it.
- Polynomial basically fits wide range of curvature.
- Polynomial provides the best approximation of the relationship between dependent and independent variable.

Disadvantages of using Polynomial Regression

- These are too sensitive to the outliers.
- The presence of one or two outliers in the data can seriously affect the results of a nonlinear analysis.
- In addition there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures

x=df[['lotsize','bathrms']]
y=df['price']

poly = PolynomialFeatures(degree = 2)
xpoly = poly.fit_transform(x)

xtrain,xtest,ytrain,ytest=train_test_split(xpoly,y,test_size=0.3,random_state=1)

regressor = LinearRegression()
pr=regressor.fit(xtrain, ytrain)
```

2.9 Shrinkage Methods

The regression methods arrived by including the regularization parameter is called as shrinkage methods. **Lasso, Ridge and Elastic Net** regression models falls under the category of shrinkage methods.

Regularization

Regularize means to make things regular or acceptable. Regularization is a technique used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting. It is used for tuning the function by adding an additional penalty term in the error function. It tries to shrink the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. It is artificially penalize higher degree polynomials.

Regularization types – L1 and L2

- L1 – Loss/Cost function – **sum(|y-f(x)|)** -**absolute penalty**
- L2 – Loss/Cost Function - **Sum(y-f(x))^2** - **squared penalty**

The cost function of simple linear regression model is

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2$$

Cost function for simple linear model

Lasso Regression

It uses L1 regularization.

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p |w_j|$$

Cost Function for Lasso Regression

Ridge Regression

It uses L2 Regularization.

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p w_j^2$$

Cost function for ridge regression

ElasticNet Regression - uses both L1 and L2 regularization

```
## Lasso Regression
from sklearn.linear_model import Lasso
laso=Lasso()
lr=laso.fit(xtrain,ytrain)

## Ridge Regression
from sklearn.linear_model import Ridge
rid=Ridge()
rr=rid.fit(xtrain,ytrain)

## ElasticNet
from sklearn.linear_model import ElasticNet
enet=ElasticNet()
enr=enet.fit(xtrain,ytrain)
```

2.10 Introduction to Classification

Classification is a supervised learning approach in which the computer program learns from the input data (x features and y labels) and then uses this learning to classify new observations. This data set may simply be bi-class (binomial classification - like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class (multinomial classification). Some practical examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. A Classifier is a linguistic symbol that represents a class or group of objects

or subjects. Some of the classifiers are Naïve Bayes, Decision Tree Classifier, kNN classifier, SVM etc.

2.11 Logistic Regression

It is a form of regression which allows the prediction of discrete variables by a mixture of continuous and discrete predictors. It results in a unique transformation of dependent variables which impacts not only the estimation process but also the coefficients of independent variables. It addresses the same question which multiple regression does but with no distributional assumptions on the predictors. In logistic regression the outcome variable is binary. The purpose of the analysis is to assess the effects of multiple explanatory variables, which can be numeric or categorical or both.

Input $x_1 \dots x_n$ may be nominal or continuous or mixture of two types

We can't use a normal linear regression model on binary groups. It won't lead to a good fit.

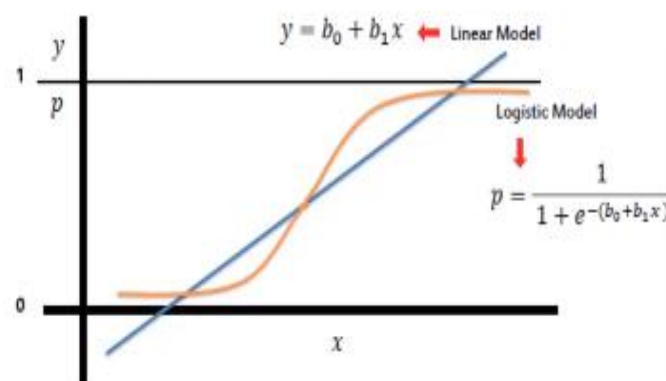
Hence the solution is to transform linear Regression into logistic regression curve.

Sigmoid function


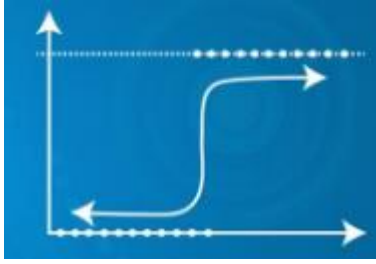
The sigmoid function also known as logistic function is used

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

The output can be 0 or 1. The Cutoff point at 0.5, anything below it results in 0 class and above results in class 1. Substitute the linear regression model in the sigmoid function.



Difference between Linear Regression and Logistic Regression

Linear Regression	Logistic Regression
It is used to solve regression problems	It is used to solve classification problems
It models the relationship between a dependent variable and one or more independent variable	It predicts the probability of an outcome that can only have two values at the output either 0 or 1
The predicted output is a continuous variable	The predicted output is a discrete variable
Predicted output Y can exceed 0 and 1 range	Predicted output Y lies within 0 and 1 range
	
Predicted output Y can exceed 0 and 1 range	Predicted output

Types of Logistic Regression

Below are the 2 types of Logistic Regression:

1. Binary Logistic Regression

It is used when the dependent variable is dichotomous i.e. like a tree with two branches. It is used when the dependent variable is non-parametric.

Used when

- If there is no linearity
- There are only two levels of the dependent variable.
- If multivariate normality is doubtful.

2. Multinomial Logistic Regression

Multinomial logistic regression analysis requires that the independent variables be metric or dichotomous. It does not make any assumptions of linearity, normality, and homogeneity of variance for the independent variables.

It is used when the dependent variable has more than two categories. It is used to analyze relationships between a non-metric dependent variable and metric or dichotomous independent variables, then compares multiple groups through a combination of binary logistic regressions. In the end, it provides a set of coefficients for each of the two comparisons. The coefficients for the reference group are taken to be all zeros. Finally, prediction is done based on the highest resultant probability.

2.12 Evaluation Metrics

Confusion Matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. It is the cross tabulation of actual vs the predicted.

n=165		Predicted: NO	Predicted: YES
Actual: NO		50	10
Actual: YES		5	100

In python No – Negative Test – False = 0, Yes – Positive Test – True – 1

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy: Overall, how often is the classifier correct?

$$(TP+TN)/total$$

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) = (100+50)/165 = 0.91$$

Misclassification Rate: Overall, how often is it wrong?

$$(FP+FN)/total = (10+5)/165 = 0.09$$

equivalent to 1 minus Accuracy

also known as "Error Rate"

True Positive Rate (Recall/Sensitivity): When it's actually yes, how often does it predict yes?

$$TP/actual\ yes$$

$$TPR\ or\ Recall = TP/TP+FN = 100/105 = 0.95$$

Specificity: When its actually No, how often does it predict No?

$$TN/actual\ No = TN/TN+FP = 50/60 = 0.83$$

False Positive Rate (Precision): When it is actually No, how often it is wrong?

$$FP/Actual\ No = 1-Specificity$$

$$FPR = Precision = FP/FP+TN = 10/60 = 0.167$$

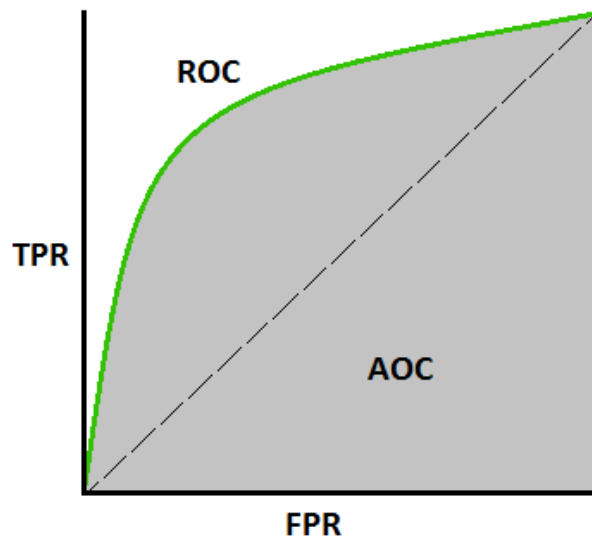
AUC-ROC

ROC (Receiver Operating Characteristics)

AUC (Area Under The Curve)

ROC is a probability curve and AUC represents degree or measure of separability. It tells how much the model is capable of distinguishing between classes. The ROC curve is plotted

with FPR against the TPR where FPR is on x-axis and TPR is on the y-axis. AUC value should be higher than 50%. Higher the AUC better the model. When it less than or equal to 50%, model is worse. AUC can be any value between 0 and 1.



Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

Build the model

```
logmodel = LogisticRegression()
```

```
logmodel.fit(xtrain,ytrain)
```

Predict

```
lg_pred = logmodel.predict(xtest)
```

Model Evaluation

```
from sklearn.metrics import classification_report,accuracy_score,  
confusion_matrix
```

```
print(confusion_matrix(ytest,lg_pred))
```

```
print(accuracy_score(ytest,lg_pred))
```

```
print(classification_report(ytest,lg_pred))
```

AUC- ROC

```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
from matplotlib import pyplot
```

calculate AUC -- Area under Roc

```
auc = roc_auc_score(ytest, lg_pred)
```

```
print('AUC=',auc)
```

calculate roc -- receiver optimistic curve

```
fpr, tpr, threshold = roc_curve(ytest, lg_pred)
```

plot the roc curve for the model

```
pyplot.plot(fpr, tpr)
```

```
pyplot.show()
```

2.13 KNN Classification

K Nearest Neighbor (KNN) algorithm is basically a classification algorithm in Machine Learning which belongs to the supervised learning category. However, it can be used in regression problems as well. KNN algorithms have been used since 1970 in many applications like pattern recognition, data mining, statistical estimation, and intrusion detection and many more. It is widely disposable in real-life scenarios since it is non-parametric, i.e. it does not make any underlying assumptions about the distribution of data. KNN under classification problem basically classifies the whole data into training data and test sample data. The distance between training points and sample points is evaluated and the point with the lowest distance is said to be the nearest neighbor. KNN algorithm predicts the result on the basis of the majority. Since, it calculates the distance only when the sample point arrives, it is called as lazy learning algorithm. This makes training faster and testing phase slower and costlier.

In K-NN whole data is classified into training and test sample data. In a classification problem, k nearest algorithm is implemented using the following steps.

1. Pick a value for k, where k is the number of training examples in feature space.
2. Calculate the distance of unknown data points from all the training examples.

Distance Metrics

The distance can be calculated in the following ways:

a. Euclidian distance,

$$dist(d_i, d_j) = \sqrt{\sum_k (a_{i,k} - a_{j,k})^2}$$

b. Manhattan distance,

$$D = \sum_{i=1}^n |x_i - y_i|$$

3. Search for the k observations in the training data that are nearest to the measurements of the unknown data point.
4. Calculate the distance between the unknown data point and the training data.
5. The training data which is having the smallest value will be declared as the nearest neighbor.

In the KNN-regression problem, the only difference is that the distance between training points and sample points is evaluated and the point with the lowest average distance is declared as the nearest neighbor. It predicts the result on the basis of the average of the total sum.

Example

Consider the following data points

Name	Acid Durability	Strength	class
Type-1	7	7	Bad
Type-2	7	4	Bad
Type-3	3	4	Good
Type-4	1	4	Good
Type-5	3	7	??

Find the Euclidean distance measure and rank them according to the closeness

Name	Acid Durability	Strength	class	Distance	Rank
Type-1	7	7	Bad	$\text{Sqrt}(7-3)^2 + (7-7)^2 = 4$	3
Type-2	7	4	Bad	5	4
Type-3	3	4	Good	3	1
Type-4	1	4	Good	3.6	2
Type-5	3	7	??		

If $k=1 \Rightarrow$ The new test data is close to Type-3 and hence class label is good.

If $k=3 \Rightarrow$ The data that is close to test data is Type-3, Type-4 and Type-1 where 2 are good and 1 is bad. The probability of good is high, hence the new test data belongs to class label good.

How to Choose the K Value?

Hence the value of k is chosen properly according to the need.

- If k is chosen large it will be less sensitive to noise and hence performance increases.
- If k is chosen optimal/small it will be able to capture fine structures if exist in the feature space.
- if k is too small it may lead to overfitting i.e. algorithm performs excellently on the training set and its performance degrades on unseen test data.

```
## KNN Classification

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(xtrain,ytrain)
knnpred = knn.predict(xtest)

from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(ytest,knnpred))
print(classification_report(ytest,knnpred))
print("Accuracy=",accuracy_score(ytest,knnpred))
```

2.14 Naïve Bayes Classification

Naive Bayes Algorithm is one of the popular classification machine learning algorithms that helps to classify the data based upon the conditional probability values computation. It implements the Bayes theorem for the computation and used class levels represented as feature values or vectors of predictors for classification. Naive Bayes Algorithm is a fast algorithm for classification problems. This algorithm is a good fit for real-time prediction, multi-class prediction, recommendation system, text classification, and sentiment analysis use cases. Naive Bayes Algorithm can be built using Gaussian, Multinomial and Bernoulli distribution. This algorithm is scalable and easy to implement for the large data set.

It helps to calculate the posterior probability $P(c|x)$ using the prior probability of class $P(c)$, the prior probability of predictor $P(x)$ and the probability of predictor given class, also called as likelihood $P(x|c)$.

The formula or equation to calculate posterior probability is:

- $P(c|x) = (P(x|c) * P(c)) / P(x)$

Steps

- **Step 1:** Make Frequency Tables Using Data Sets.
- **Step 2:** Make a likelihood table by calculating the probabilities of each features with its levels
- **Step 3:** Now we need to calculate the posterior probability using Naive Bayes equation for each class

Example

COLOR	TYPE	ORIGIN	STOLEN
RED	SPORTS	DOMESTIC	YES
RED	SPORTS	DOMESTIC	NO
RED	SPORTS	DOMESTIC	YES
YELLOW	SPORTS	DOMESTIC	NO
YELLOW	SPORTS	IMPORTED	YES
YELLOW	SUV	IMPORTED	NO
YELLOW	SUV	IMPORTED	YES
YELLOW	SUV	DOMESTIC	NO
RED	SUV	IMPORTED	NO
RED	SPORTS	IMPORTED	YES

The prior probability of target (stolen) variable is

$$P(\text{Stolen=Yes}) = 5/10$$

$$P(\text{Stolen=No}) = 5/10$$

The predictors cross tabulation is as follows:

	YES	NO
RED	3 / 5	2 / 5
YELLOW	2 / 5	3 / 5

	YES	NO
SPORTS	1 / 5	2 / 5
SUV	1 / 5	3 / 5

	YES	NO
DOMESTIC	2 / 5	3 / 5
IMPORTED	3 / 5	2 / 5

Say, an unseen instance X comes with feature values $X=(RED,SUV,DOMESTIC)$, Now it is easy to predict the class label of this instance.

$$P(X|YES) = P(RED|YES) * P(SUV|YES) * P(DOMESTIC|YES) P(YES)$$

$$= 3/5 * 1/5 * 2/5 = 0.024 * 0.5 = \mathbf{0.012}$$

$$P(X|NO) = P(RED|NO) * P(SUV|NO) * P(DOMESTIC|NO) P(NO)$$

$$= 2/5 * 3/5 * 3/5 = 0.072 * 0.5 = \mathbf{0.036}$$

0.036 > 0.012, Hence unseen example is classified as **NO**

Advantages

- Easy to implement & Fast
- If the independence assumption holds then it works more efficiently than other algorithms.
- It requires less training data.
- It is highly scalable.
- It can make probabilistic predictions.
- Can handle both continuous and discrete data.
- Insensitive towards irrelevant features.
- It can work easily with missing values.
- Easy to update on arrival of new data.
- Best suited for text classification problems.

Disadvantages

- The strong assumption about the features to be independent which is hardly true in real life applications.
- Data scarcity may lead to chances of loss of accuracy.
- Zero Frequency i.e. if the category of any categorical variable is not seen in training data set then model assigns a zero probability to that category and then a prediction cannot be made.

Problem of Zero Frequency

If a particular attribute value does not occur in the training set in conjunction with every class value, then things go badly

Solution: Laplace Estimation/ smoothing Add 1 to all the numerator and compensate by adding the factor times to the denominator.

Problem					
outlook			temperature		
	yes	no		yes	no
sunny	2	3	hot	2	2
overcast	4	0	mild	4	2
rainy	3	2	cool	3	1
	yes	no		yes	no
sunny	2/9	3/5	hot	2/9	2/5
overcast	4/9	0/5	mild	4/9	2/5
rainy	3/9	2/5	cool	3/9	1/5

Solution

	yes	no
sunny	2/9	4/8
overcast	4/9	1/8
rainy	3/9	3/8

Naive Classification

```

from sklearn.naive_bayes import GaussianNB          #works with normal distribution
naivebayes = GaussianNB()
naivebayes.fit(xtrain, ytrain)
nbpred=naivebayes.predict(xtest)
print(confusion_matrix(ytest,nbpred))
print("Accuracy=",accuracy_score(ytest,nbpred))
print("AUC=",roc_auc_score(ytest,nbpred))

```