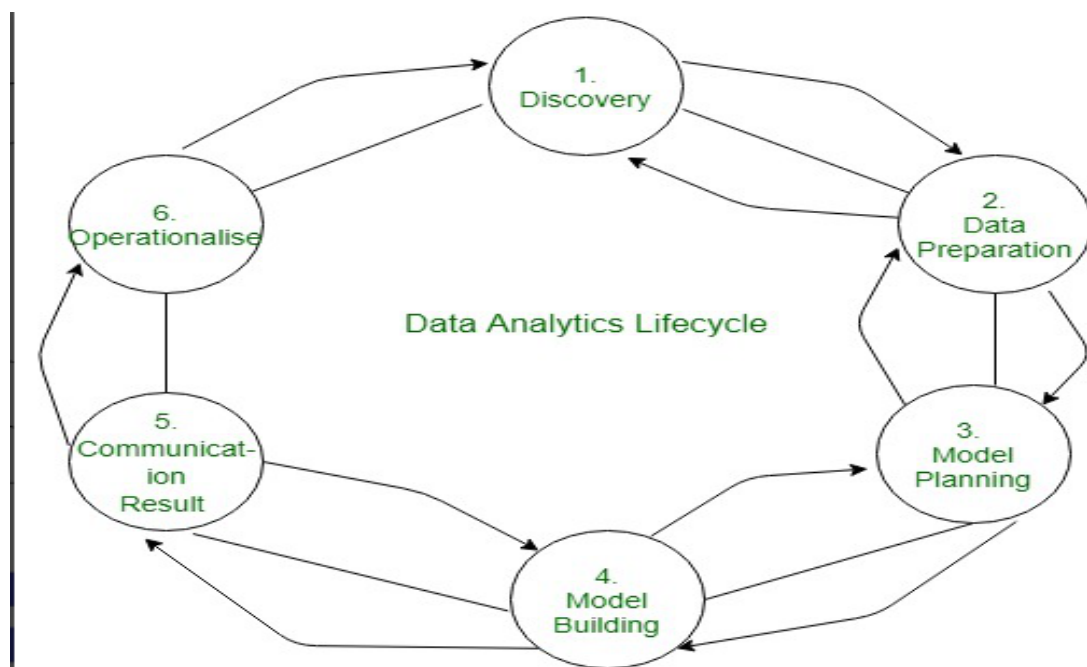
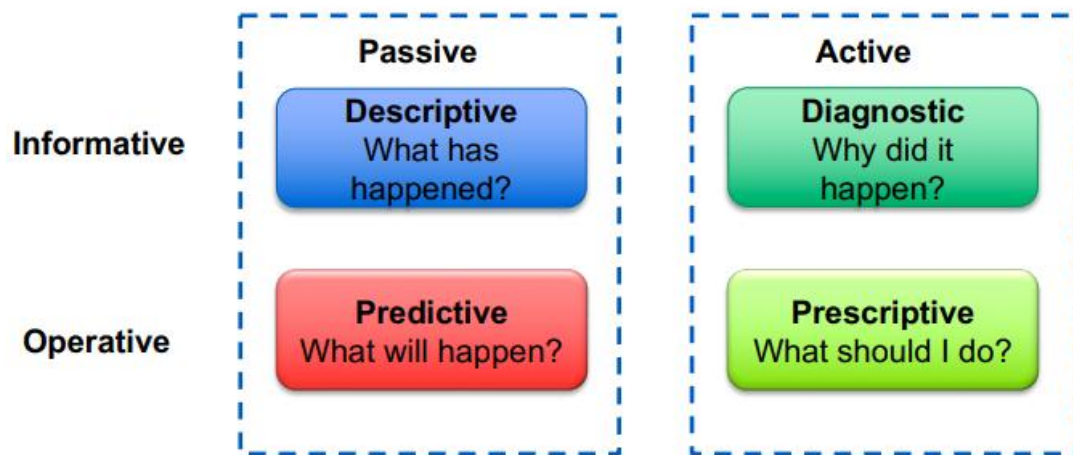


## Unit 1

Data analytics—the practice of examining data to answer questions, identify trends, and extract insights—can provide you with the information necessary to strategize and make impactful business decisions.

There are four key types of data analytics:

- **Descriptive**, which answers the question, “What happened?”
- **Diagnostic**, which answers the question, “Why did this happen?”
- **Prescriptive**, which answers the question, “What should we do next?”
- **Predictive**, which answers the question, “What might happen in the future?”



# **DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM**

## **Data Analytics Lifecycle :**

The [Data analytic](#) lifecycle is designed for Big Data problems and data science projects. The cycle is iterative to represent real project. To address the distinct requirements for performing analysis on Big Data, step – by – step methodology is needed to organize the activities and tasks involved with acquiring, processing, analyzing, and repurposing data.

### **Phase 1: Discovery –**

- The data science team learn and investigate the problem.
- Develop context and understanding.
- Come to know about data sources needed and available for the project.
- The team formulates initial hypothesis that can be later tested with data.

### **Phase 2: Data Preparation –**

- Steps to explore, preprocess, and condition data prior to modeling and analysis.
- It requires the presence of an analytic sandbox, the team execute, load, and transform, to get data into the sandbox.
- Data preparation tasks are likely to be performed multiple times and not in predefined order.
- Several tools commonly used for this phase are – Hadoop, Alpine Miner, Open Refine, etc.

### **Phase 3: Model Planning –**

- Team explores data to learn about relationships between variables and subsequently, selects key variables and the most suitable models.
- In this phase, data science team develop data sets for training, testing, and production purposes.
- Team builds and executes models based on the work done in the model planning phase.
- Several tools commonly used for this phase are – Matlab, STASTICA.

# **DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM**

## **Phase 4: Model Building –**

- Team develops datasets for testing, training, and production purposes.
- Team also considers whether its existing tools will suffice for running the models or if they need more robust environment for executing models.
- Free or open-source tools – R, Python, Octave, WEKA.
- Commercial tools – Matlab, SAS.

## **Phase 5: Communication Results –**

- After executing model team need to compare outcomes of modeling to criteria established for success and failure.
- Team considers how best to articulate findings and outcomes to various team members and stakeholders, taking into account warning, assumptions.
- Team should identify key findings, quantify business value, and develop narrative to summarize and convey findings to stakeholders.

## **Phase 6: Operationalize –**

- The team communicates benefits of project more broadly and sets up pilot project to deploy work in controlled way before broadening the work to full enterprise of users.
- This approach enables team to learn about performance and related constraints of the model in production environment on small scale, and make adjustments before full deployment.
- The team delivers final reports, briefings, codes.
- Free or open source tools – Octave, WEKA, SQL, MADlib.

Predictive analytics is the use of data to predict future trends and events. It uses historical data to forecast potential scenarios that can help drive strategic decisions. The predictions could be for the near future—for instance, predicting the malfunction of a piece of machinery later that day—or the more distant future, such as predicting your company's cash flows for the upcoming year. Predictive analysis can be conducted manually or using machine-learning algorithms. Either way, historical data is used to make assumptions about the future. One predictive analytics tool is regression analysis, which can determine the relationship between two variables (single linear regression) or three or more variables (multiple regression). The relationships between variables are written as a mathematical equation that can help predict the outcome should one variable change.

# **DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM**

## **FIVE EXAMPLES OF PREDICTIVE ANALYTICS IN ACTION**

### **1. Finance: Forecasting Future Cash Flow**

Every business needs to keep periodic financial records, and predictive analytics can play a big role in forecasting your organization's future health. Using historical data from previous financial statements, as well as data from the broader industry, you can project sales, revenue, and expenses to craft a picture of the future and make decisions.

HBS Professor V.G. Narayanan mentions the importance of forecasting in the course Financial Accounting which is also part of CORE.

“Managers need to be looking ahead in order to plan for the future health of their business,” Narayanan says. “No matter the field in which you work, there is always a great amount of uncertainty involved in this process.”

### **2. Entertainment & Hospitality: Determining Staffing Needs**

One example explored in Business Analytics is casino and hotel operator Caesars Entertainment's use of predictive analytics to determine venue staffing needs at specific times. In entertainment and hospitality, customer influx and outflux depend on various factors, all of which play into how many staff members a venue or hotel needs at a given time. Overstaffing costs money, and understaffing could result in a bad customer experience, overworked employees, and costly mistakes. To predict the number of hotel check-ins on a given day, a team developed a multiple regression model that considered several factors. This model enabled Caesars to staff its hotels and casinos and avoid overstaffing to the best of its ability.

### **3. Marketing: Behavioral Targeting**

In marketing, consumer data is abundant and leveraged to create content, advertisements, and strategies to better reach potential customers where they are. By examining historical behavioral data and using it to predict what will happen in the future, you engage in predictive analytics. Predictive analytics can be applied in marketing to forecast sales trends at various times of the year and plan campaigns accordingly. Additionally, historical behavioral data can help you predict a lead's likelihood of moving down the funnel from awareness to purchase. For instance, you could use a single linear regression model to determine that the number of content offerings a lead engages with predicts—with a statistically significant level of certainty—their likelihood of converting to a customer down the line. With this knowledge, you can plan targeted ads at various points in the customer's lifecycle.

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

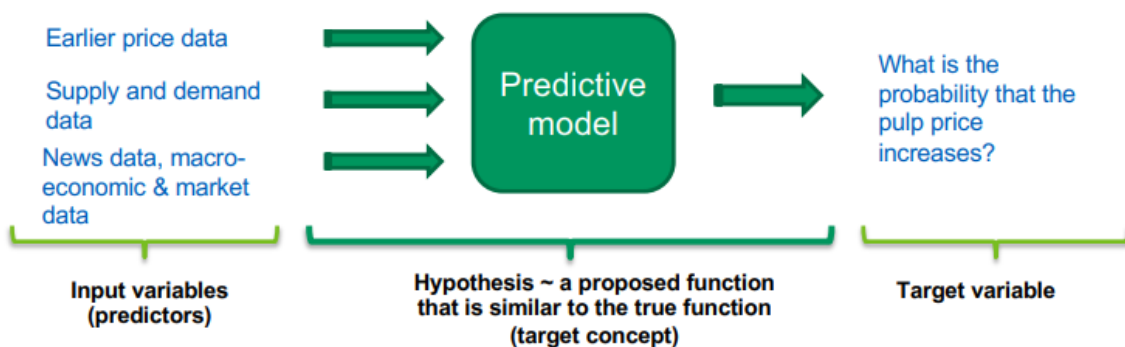
## 4. Manufacturing: Preventing Malfunction

While the examples above use predictive analytics to take action based on likely scenarios, you can also use predictive analytics to prevent unwanted or harmful situations from occurring. For instance, in the manufacturing field, algorithms can be trained using historical data to accurately predict when a piece of machinery will likely malfunction. When the criteria for an upcoming malfunction are met, the algorithm is triggered to alert an employee who can stop the machine and potentially save the company thousands, if not millions, of dollars in damaged product and repair costs. This analysis predicts malfunction scenarios in the moment rather than months or years in advance. Some algorithms even recommend fixes and optimizations to avoid future malfunctions and improve efficiency, saving time, money, and effort. This is an example of prescriptive analytics; more often than not, one or more types of analytics are used in tandem to solve a problem.

## 5. Health Care: Early Detection of Allergic Reactions

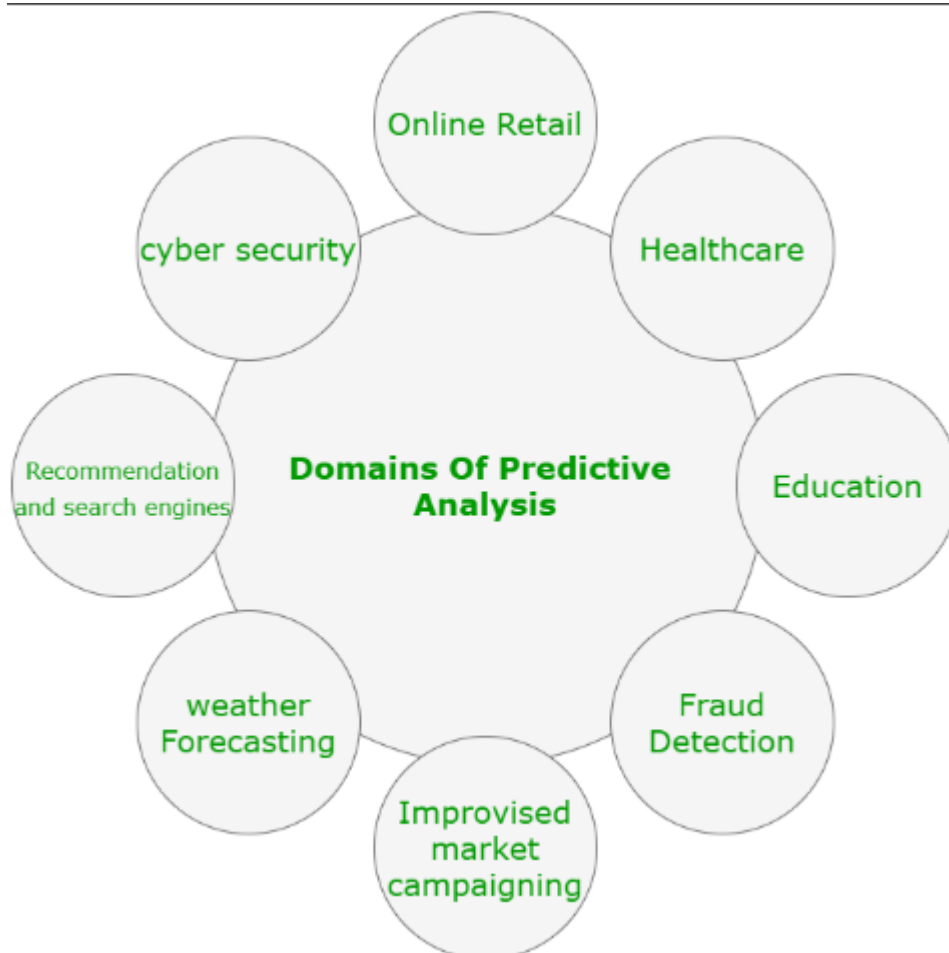
Another example of using algorithms for rapid, predictive analytics for prevention comes from the health care industry. The Wyss Institute at Harvard University partnered with the KeepSmilin4Abbie Foundation to develop a wearable piece of technology that predicts an anaphylactic allergic reaction and automatically administers life-saving epinephrine. The sensor, called AbbieSense, detects early physiological signs of anaphylaxis as predictors of an ensuing reaction—and it does so far quicker than a human can. When a reaction is predicted to occur, an algorithmic response is triggered. The algorithm can predict the reaction's severity, alert the individual and caregivers, and automatically inject epinephrine when necessary. The technology's ability to predict the reaction at a faster speed than manual detection could save lives.

### Example: How to predict development of pulp price?



## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

Attribute	Analytics	Predictive Analytics
	Understand the Past	Gain Insights
Purpose:	Observe Trends	Make Decisions
	Catalyst for Discussion	Take Action
View:	Historical and Current	Future Oriented
Metrics Type:	Lagging Indicators	Leading Indicators
Data Used:	Raw & Compiled	Information
Data Type:	Structured	Structured and Unstructured
Users:	Middle & Senior Mgt	C-Level & Senior Mgt
	Analysts, End Users	Strategists, Analysts, Mgrs
Benefits:	Gaining an understanding of data	Gaining Information & Insights
	Productivity Improvements	Process Improvements





# **DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM**

## **Steps To Perform Predictive Analysis:**

Some basic steps should be performed in order to perform predictive analysis.

### **1. Define Problem Statement:**

Define the project outcomes, the scope of the effort, objectives, identify the data sets that are going to be used.

### **2. Data Collection:**

Data collection involves gathering the necessary details required for the analysis. It involves the historical or past data from an authorized source over which predictive analysis is to be performed.

### **3. Data Cleaning:**

Data Cleaning is the process in which we refine our data sets. In the process of data cleaning, we remove un-necessary and erroneous data. It involves removing the redundant data and duplicate data from our data sets.

### **4. Data Analysis:**

It involves the exploration of data. We explore the data and analyze it thoroughly in order to identify some patterns or new outcomes from the data set. In this stage, we discover useful information and conclude by identifying some patterns or trends.

### **5. Build Predictive Model:**

In this stage of predictive analysis, we use various algorithms to build predictive models based on the patterns observed. It requires knowledge of python, R, Statistics and MATLAB and so on. We also test our hypothesis using standard statistic models.

### **6. Validation:**

It is a very important step in predictive analysis. In this step, we check the efficiency of our model by performing various tests. Here we provide sample input sets to check the validity of our model. The model needs to be evaluated for its accuracy in this stage.

### **7. Deployment:**

In deployment we make our model work in a real environment and it helps in everyday discussion making and make it available to use.

### **8. Model Monitoring:**

Regularly monitor your models to check performance and ensure that we have proper results. It is seeing how model predictions are performing against actual data sets.

## **What is Feature Selection?**

A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

So, we can define feature Selection as, "*It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building.*" Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

### Need for Feature Selection

Before implementing any technique, it is really important to understand, need for the technique and so for the Feature Selection. As we know, in machine learning, it is necessary to provide a pre-processed and good input dataset in order to get better outcomes. We collect a huge amount of data to train our model and help it to learn better. Generally, the dataset consists of noisy data, irrelevant data, and some part of useful data. Moreover, the huge amount of data also slows down the training process of the model, and with noise and irrelevant data, the model may not predict and perform well. So, it is very necessary to remove such noises and less-important data from the dataset and to do this, and Feature selection techniques are used.

Selecting the best features helps the model to perform well. For example, Suppose we want to create a model that automatically decides which car should be crushed for a spare part, and to do this, we have a dataset. This dataset contains a Model of the car, Year, Owner's name, Miles. So, in this dataset, the name of the owner does not contribute to the model performance as it does not decide if the car should be crushed or not, so we can remove this column and select the rest of the features(column) for the model building.

Below are some benefits of using feature selection in machine learning:

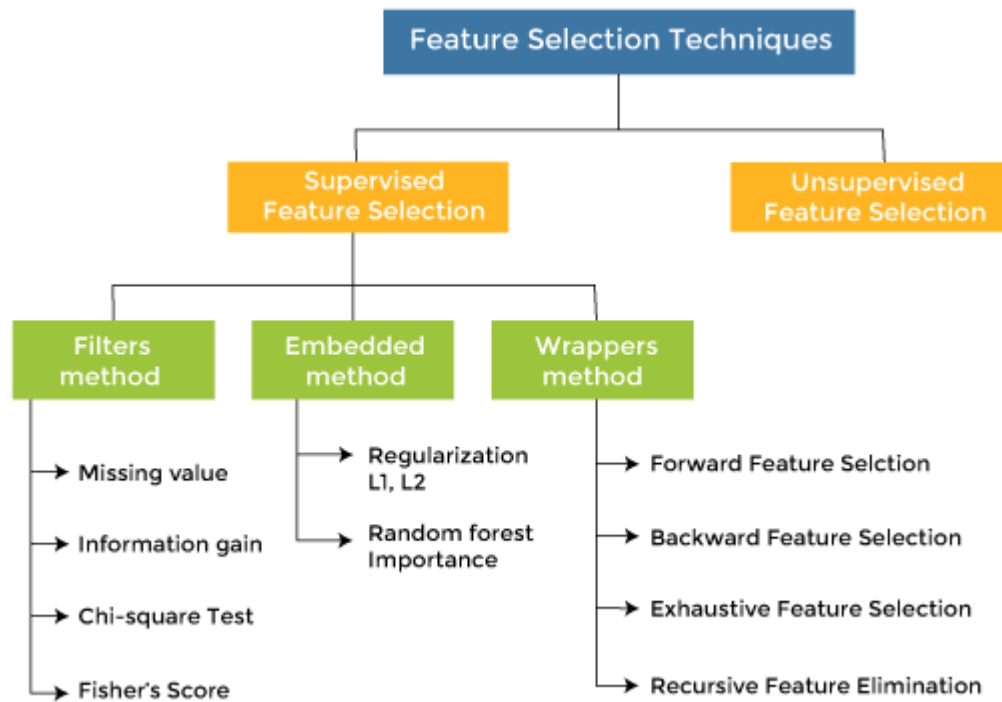
- It helps in avoiding the curse of dimensionality.
- It helps in the simplification of the model so that it can be easily interpreted by the researchers.
- It reduces the training time.
- It reduces overfitting hence enhance the generalization.

### Feature Selection Techniques

There are mainly two types of Feature Selection techniques, which are:

- **Supervised Feature Selection technique**  
Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection technique**  
Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.

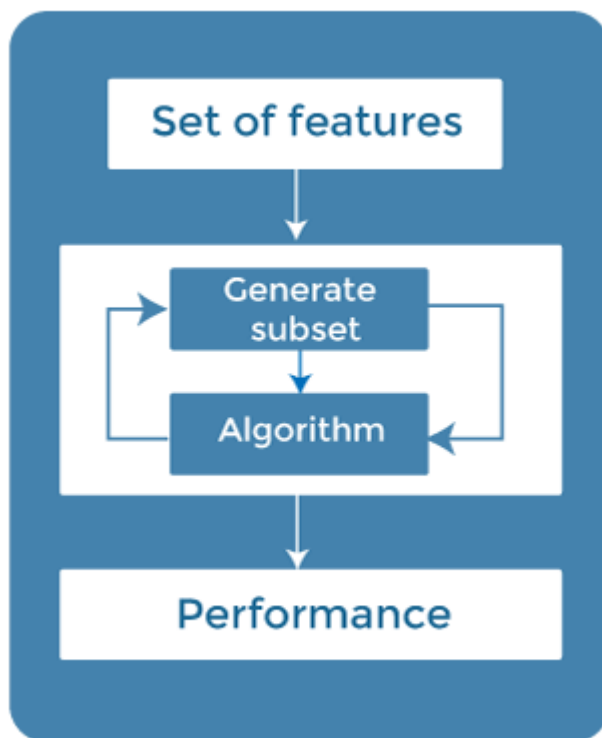




There are mainly three techniques under supervised feature Selection:

## 1. Wrapper Methods

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

Some techniques of wrapper methods are:

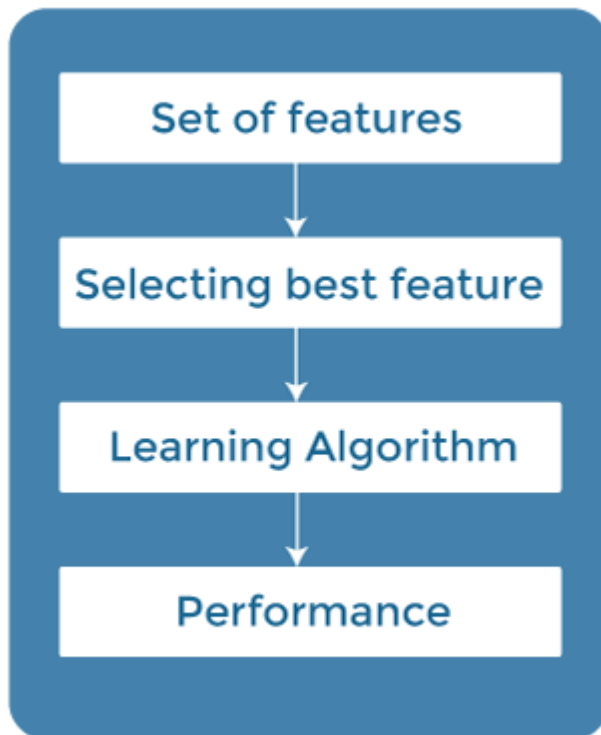
- **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new variable/feature does not improve the performance of the model.
- **Backward elimination** - Backward elimination is also an iterative approach, but it is the opposite of forward selection. This technique begins the process by considering all the features and removes the least significant feature. This elimination process continues until removing the features does not improve the performance of the model.
- **Exhaustive Feature Selection**- Exhaustive feature selection is one of the best feature selection methods, which evaluates each feature set as brute-force. It means this method tries & make each possible combination of features and return the best performing feature set.
- **Recursive Feature Elimination**-  
Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features. Now, an estimator is trained with each set of features, and the importance of each feature is determined using *coef\_attribute* or through a *feature\_importances\_attribute*.

## 2. Filter Methods

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.

The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.

The advantage of using filter methods is that it needs low computational time and does not overfit the data.



Some common techniques of Filter methods are as follows:

- Information Gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio

**Information Gain:** Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

**Chi-square Test:** Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

**Fisher's Score:**

Fisher's score is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

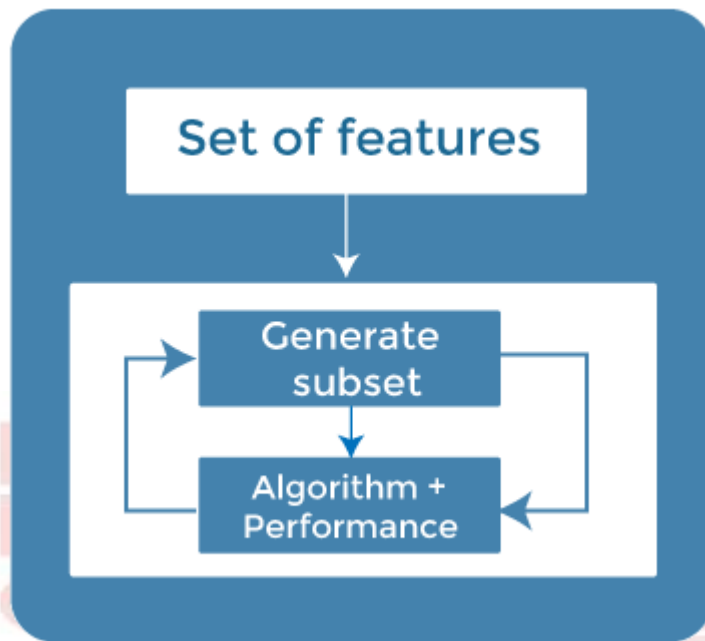
**Missing Value Ratio:**

The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.

$$\text{Missing Value Ratio} = \frac{\text{Number of Missing values} \times 100}{\text{Total number of observations}}$$

## 3. Embedded Methods

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.



These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:

- **Regularization**- Regularization adds a penalty term to different parameters of the machine learning model for avoiding overfitting in the model. This penalty term is added to the coefficients; hence it shrinks some coefficients to zero. Those features with zero coefficients can be removed from the dataset. The types of regularization techniques are L1 Regularization (Lasso Regularization) or Elastic Nets (L1 and L2 regularization).
- **Random Forest Importance** - Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, which is a type of bagging algorithm that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the **impurity (Gini**

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

impurity) over all the trees. Nodes are arranged as per the impurity values, and thus it allows to pruning of trees below a specific node. The remaining nodes create a subset of the most important features.

### How to choose a Feature Selection Method?

For machine learning engineers, it is very important to understand that which feature selection method will work properly for their model. The more we know the datatypes of variables, the easier it is to choose the appropriate statistical measure for feature selection.

To know this, we need to first identify the type of input and output variables. In machine learning, variables are of mainly two types:

- **Numerical Variables:** Variable with continuous values such as integer, float
- **Categorical Variables:** Variables with categorical values such as Boolean, ordinal, nominals.

Input Variable	Output Variable	Feature Selection technique
Numerical	Numerical	<ul style="list-style-type: none"><li>○ Pearson's correlation coefficient (For linear Correlation).</li><li>○ Spearman's rank coefficient (for non-linear correlation).</li></ul>
Numerical	Categorical	<ul style="list-style-type: none"><li>○ ANOVA correlation coefficient (linear).</li><li>○ Kendall's rank coefficient (nonlinear).</li></ul>
Categorical	Numerical	<ul style="list-style-type: none"><li>○ Kendall's rank coefficient (linear).</li><li>○ ANOVA correlation coefficient (nonlinear).</li></ul>
Categorical	Categorical	<ul style="list-style-type: none"><li>○ Chi-Squared test (contingency tables).</li><li>○ Mutual Information.</li></ul>

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

Covariance	Correlation
Covariance is a measure to indicate the extent to which two random variables change in tandem.	Correlation is a measure used to represent how strongly two random variables are related to each other.
Covariance is nothing but a measure of correlation.	Correlation refers to the scaled form of covariance.
Covariance indicates the direction of the linear relationship between variables.	Correlation on the other hand measures both the strength and direction of the linear relationship between two variables.
Covariance can vary between $-\infty$ and $+\infty$	Correlation ranges between -1 and +1
Covariance is affected by the change in scale. If all the values of one variable are multiplied by a constant and all the values of another variable are multiplied, by a similar or different constant, then the covariance is changed.	Correlation is not influenced by the change in scale.
Covariance assumes the units from the product of the units of the two variables.	Correlation is dimensionless, i.e. It's a unit-free measure of the relationship between variables.
Covariance of two dependent variables measures how much in real quantity (i.e. cm, kg, liters) on average they co-vary.	Correlation of two dependent variables measures the proportion of how much on average these variables vary w.r.t one another.
Covariance is zero in case of independent variables (if one variable moves and the other doesn't) because then the variables do not necessarily move together.	Independent movements do not contribute to the total correlation. Therefore, completely independent variables have a zero correlation.



# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

## What is covariance?

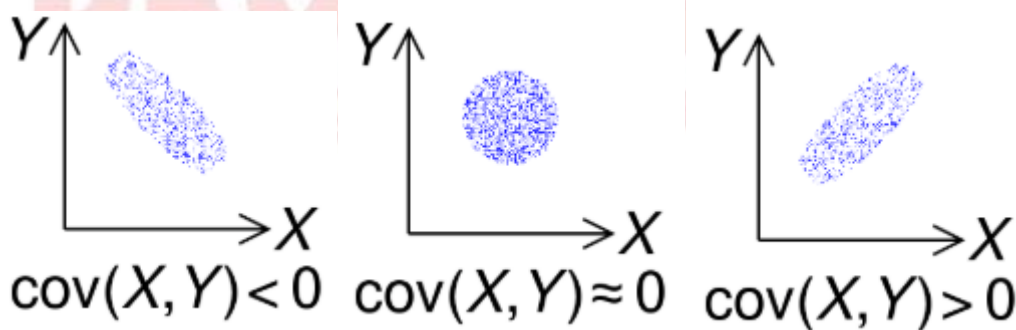
Covariance signifies the direction of the linear relationship between the two variables. By direction we mean if the *variables* are directly proportional or inversely proportional to each other. (Increasing the value of one variable might have a positive or a negative impact on the value of the other variable).

The values of covariance can be any number between the two opposite infinities. Also, it's important to mention that covariance only measures how two variables change together, not the dependency of one variable on another one.

The value of covariance between 2 variables is achieved by taking the summation of the product of the differences from the means of the variables as follows:

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

The upper and lower limits for the covariance depend on the variances of the variables involved. These variances, in turn, can vary with the scaling of the variables. Even a change in the units of measurement can change the covariance. Thus, covariance is only useful to find the direction of the relationship between two variables and not the magnitude. Below are the plots which help us understand how the covariance between two variables would look in different directions.



## Example:

X	Y
10	40
12	48
14	56

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

8	32
---	----

**Step 1: Calculate Mean of X and Y**

Mean of X (  $\mu_x$  ) :  $10+12+14+8 / 4 = 11$

Mean of Y ( $\mu_y$ ) =  $40+48+56+32 = 44$

**Step 2: Substitute the values in the formula**

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

$x_i - \bar{x}$	$y_i - \bar{y}$
$10 - 11 = -1$	$40 - 44 = -4$
$12 - 11 = 1$	$48 - 44 = 4$
$14 - 11 = 3$	$56 - 44 = 12$
$8 - 11 = -3$	$32 - 44 = -12$

**Substitute the above values in the formula**

$$\text{Cov}(x,y) = \frac{(-1)(-4) + (1)(4) + (3)(12) + (-3)(-12)}{4}$$

$$\text{Cov}(x,y) = 8/2 = 4$$

**Hence, Co-variance for the above data is 4**

## What is correlation?

Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables. It not only shows the kind of relation (in terms of direction) but also how strong the relationship is. Thus, we can

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

say the correlation values have standardized notions, whereas the covariance values are not standardized and cannot be used to compare how strong or weak the relationship is because the magnitude has no direct significance. It can assume values from -1 to +1.

To determine whether the covariance of the two variables is large or small, we need to assess it relative to the standard deviations of the two variables. To do so we have to normalize the covariance by dividing it with the product of the standard deviations of the two variables, thus providing a correlation between the two variables. The main result of a correlation is called the correlation coefficient.

$$\text{Correlation} = \frac{\text{Cov}(x, y)}{\sigma_x * \sigma_y}$$

where:

- cov is the covariance
- $\sigma_X$  is the standard deviation of  $X$
- $\sigma_Y$  is the standard deviation of  $Y$

The correlation coefficient is a dimensionless metric and its value ranges from -1 to +1.

The closer it is to +1 or -1, the more closely the two variables are related.

If there is no relationship at all between two variables, then the correlation coefficient will certainly be 0. However, if it is 0 then we can only say that there is no linear relationship. There could exist other functional relationships between the variables.

When the correlation coefficient is positive, an increase in one variable also increases the other. When the correlation coefficient is negative, the changes in the two variables are in opposite directions.

**Example:**

X	Y
10	40
12	48
14	56

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

8	32
---	----

### Step 1: Calculate Mean of X and Y

Mean of X (  $\mu_x$  ) :  $10+12+14+8 / 4 = 11$

Mean of Y ( $\mu_y$ ) =  $40+48+56+32/4 = 44$

### Step 2: Substitute the values in the formula

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

$x_i - \bar{x}$	$y_i - \bar{y}$
$10 - 11 = -1$	$40 - 44 = -4$
$12 - 11 = 1$	$48 - 44 = 4$
$14 - 11 = 3$	$56 - 44 = 12$
$8 - 11 = -3$	$32 - 44 = -12$

Substitute the above values in the formula

$$\text{Cov}(x,y) = \frac{(-1)(-4) + (1)(4) + (3)(12) + (-3)(-12)}{4}$$

$$\text{Cov}(x,y) = 8/2 = 4$$

Hence, Co-variance for the above data is 4

Step 3: Now substitute the obtained answer in Correlation formula

$$\text{Correlation} = \frac{\text{Cov}(x, y)}{\sigma x * \sigma y}$$

Before substitution we have to find standard deviation of x and y  
Lets take the data for X as mentioned in the table that is 10,12,14,8  
To find standard deviation

$$s = \sqrt{\frac{\sum (X - \bar{x})^2}{n - 1}}$$

**Step 1: Find the mean of x that is  $\bar{x}$**

$$10+14+12+8 / 4 = 11$$

**Step 2: Find each number deviation: Subtract each score with mean to get mean deviation**

$10 - 11 = -1$
$12 - 11 = 1$
$14 - 11 = 3$
$8 - 11 = -3$

**Step 3: Square the mean deviation obtained**

<b>-1</b>	<b>1</b>
<b>1</b>	<b>1</b>
<b>3</b>	<b>9</b>
<b>-3</b>	<b>9</b>

**Step 4: Sum the squares**

$$1+1+9+9 = 20$$

**Step5: Find the variance**

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

Divide the sum of squares with n-1 that is  $4-1 = 3$

$$20/3 = 6.6$$

**Step 6: Find the square root**

$$\text{Sqrt of } 6.6 = 2.581$$

**Therefore, Standard Deviation of x = 2.581**

**Find for Y using same method**

**The Standard Deviation of y = 10.29**

$$\text{Correlation} = 4 / (2.581 \times 10.29)$$

$$\text{Correlation} = 0.15065$$

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the *dependent variable must be a continuous/real value*. However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

- **Model the relationship between the two variables.** Such as the relationship between Income and expenditure, experience and Salary, etc.
- **Forecasting new observations.** Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

## Simple Linear Regression Model:

The Simple Linear Regression model can be represented using the below equation:

$$y = a_0 + a_1x + \epsilon$$

Where,

**$a_0$** = It is the intercept of the Regression line (can be obtained putting  $x=0$ )

**$a_1$** = It is the slope of the regression line, which tells whether the line is increasing or decreasing.

**$\epsilon$**  = The error term. (For a good model it will be negligible)

## Implementation of Simple Linear Regression Algorithm using Python

**Problem Statement example for Simple Linear Regression:**

Here we are taking a dataset that has two variables: salary (dependent variable) and experience (Independent variable). The goals of this problem is:

- **We want to find out if there is any correlation between these two variables**
- **We will find the best fit line for the dataset.**



## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

- **How the dependent variable is changing by changing the independent variable.**

In this section, we will create a Simple Linear Regression model to find out the best fitting line for representing the relationship between these two variables.

To implement the Simple Linear regression model in machine learning using Python, we need to follow the below steps:

### Step-1: Data Pre-processing

The first step for creating the Simple Linear Regression model is **data pre-processing**. We have already done it earlier in this tutorial. But there will be some changes, which are given in the below steps:

- First, we will import the three important libraries, which will help us for loading the dataset, plotting the graphs, and creating the Simple Linear Regression model.

1. **import** numpy as nm
2. **import** matplotlib.pyplot as mtp
3. **import** pandas as pd

- Next, we will load the dataset into our code:

1. `data_set= pd.read_csv('Salary_Data.csv')`

DRONACHARYA  
DRONACHARYA  
College of Engineering

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

data\_set - DataFrame

Index	YearsExperience	Salary
0	1	32383
1	1.1	45207
2	1.3	39751
3	2	43525
4	2.2	39891
5	2.7	56642
6	3	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4	55794
12	4	56957
13	4.1	57081

Format    Resize    ☒ Background color    ☒ Column min/max    Save and Close    Close

The above output shows the dataset, which has two variables: Salary and Experience.

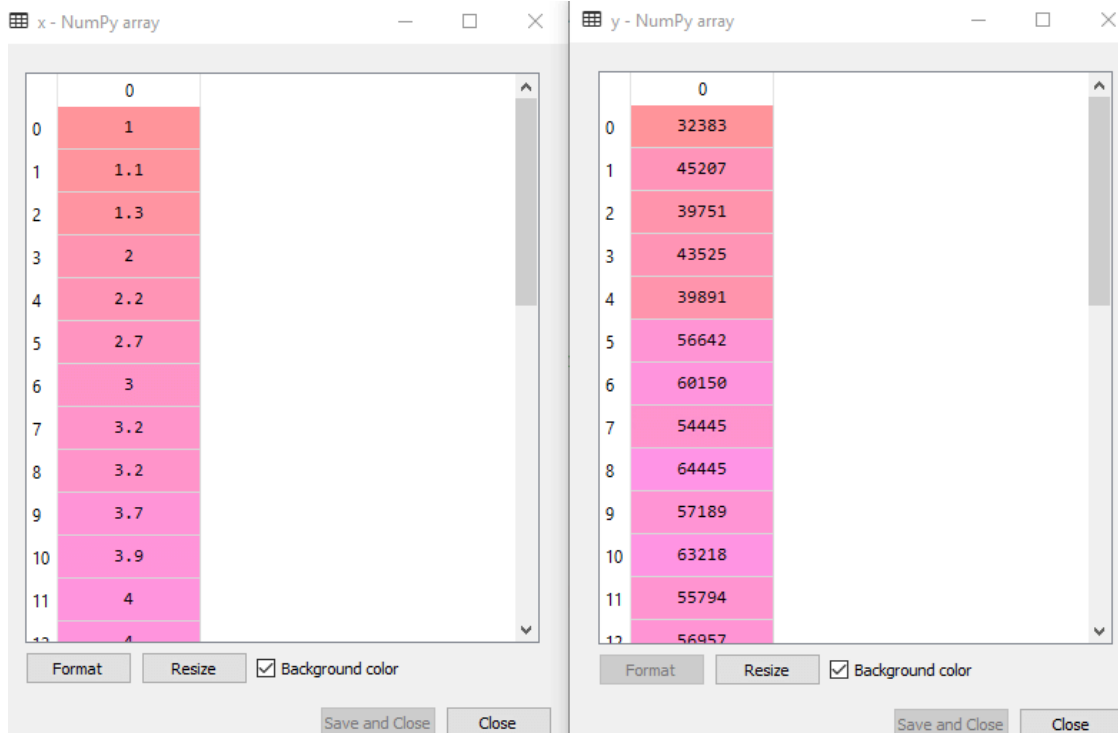
- After that, we need to extract the dependent and independent variables from the given dataset. The independent variable is years of experience, and the dependent variable is salary. Below is code for it:

1. `x= data_set.iloc[:, :-1].values`
2. `y= data_set.iloc[:, 1].values`

In the above lines of code, for x variable, we have taken -1 value since we want to remove the last column from the dataset. For y variable, we have taken 1 value as a parameter, since we want to extract the second column and indexing starts from the zero.

By executing the above line of code, we will get the output for X and Y variable as:

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM



In the above output image, we can see the X (independent) variable and Y (dependent) variable has been extracted from the given dataset.

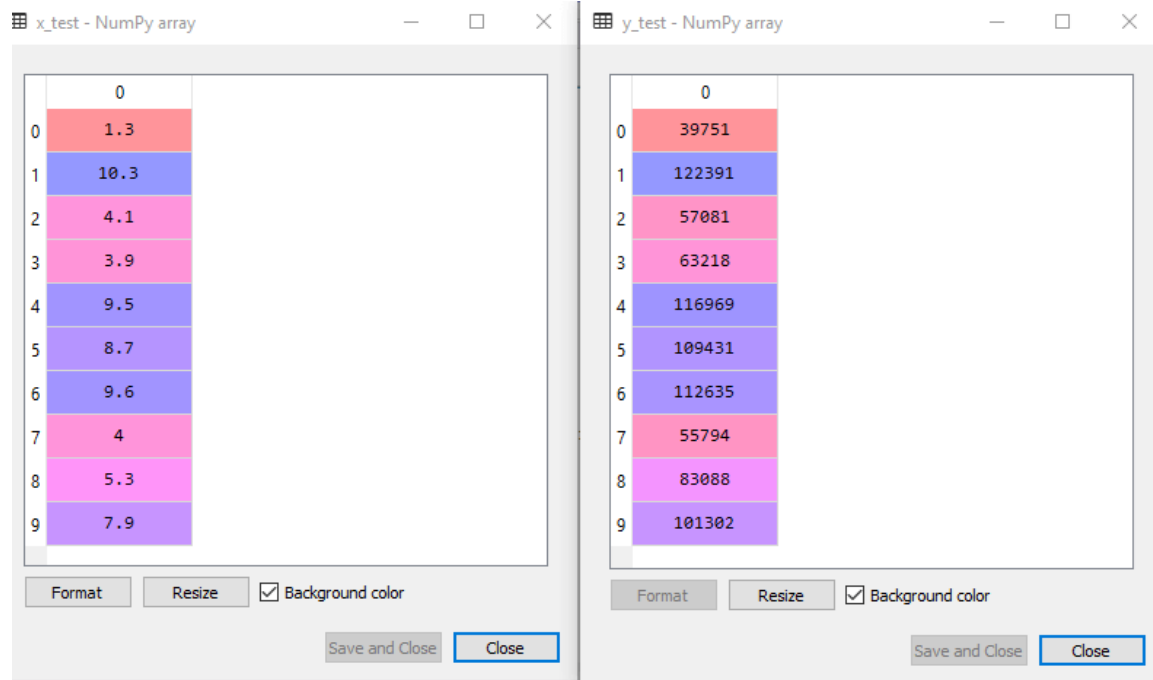
- Next, we will split both variables into the test set and training set. We have 30 observations, so we will take 20 observations for the training set and 10 observations for the test set. We are splitting our dataset so that we can train our model using a training dataset and then test the model using a test dataset. The code for this is given below:

1. # Splitting the dataset into training and test set.
2. from sklearn.model\_selection **import** train\_test\_split
3. x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 1/3, random\_state=0)

By executing the above code, we will get x-test, x-train and y-test, y-train dataset. Consider the below images:

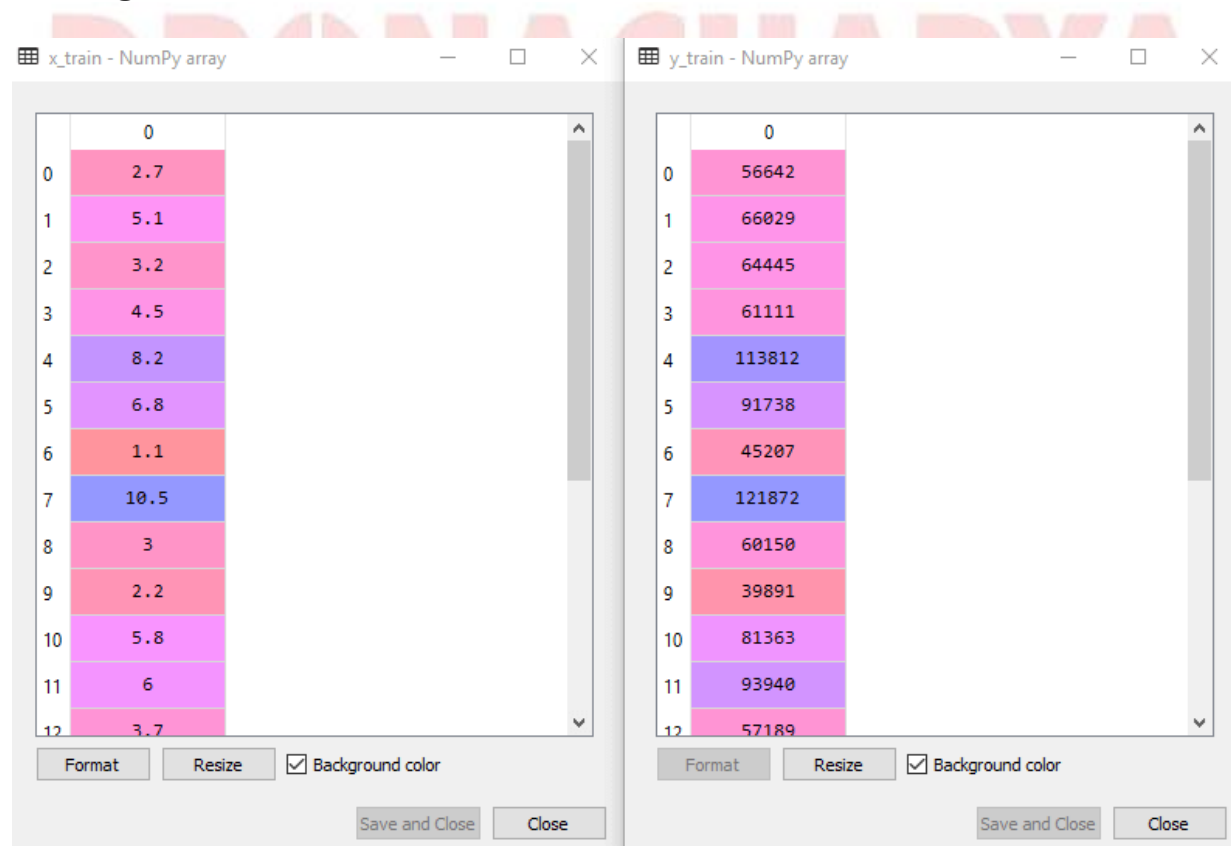
## Test-dataset:

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM



Training Dataset:

Training Dataset:



○

- For simple linear Regression, we will not use Feature Scaling. Because Python libraries take care of it for some cases, so we don't need to perform it here. Now, our

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

dataset is well prepared to work on it and we are going to start building a Simple Linear Regression model for the given problem.

### Step-2: Fitting the Simple Linear Regression to the Training Set:

Now the second step is to fit our model to the training dataset. To do so, we will import the **LinearRegression** class of the **linear\_model** library from the **scikit learn**. After importing the class, we are going to create an object of the class named as a **regressor**. The code for this is given below:

1. #Fitting the Simple Linear Regression model to the training dataset
2. from sklearn.linear\_model **import** LinearRegression
3. regressor= LinearRegression()
4. regressor.fit(x\_train, y\_train)

In the above code, we have used a **fit()** method to fit our Simple Linear Regression object to the training set. In the fit() function, we have passed the x\_train and y\_train, which is our training dataset for the dependent and an independent variable. We have fitted our regressor object to the training set so that the model can easily learn the correlations between the predictor and target variables. After executing the above lines of code, we will get the below output.

### Output:

```
Out[7]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

### Step: 3. Prediction of test set result:

dependent (salary) and an independent variable (Experience). So, now, our model is ready to predict the output for the new observations. In this step, we will provide the test dataset (new observations) to the model to check whether it can predict the correct output or not.

We will create a prediction vector **y\_pred**, and **x\_pred**, which will contain predictions of test dataset, and prediction of training set respectively.

1. #Prediction of Test and Training set result
2. y\_pred= regressor.predict(x\_test)
3. x\_pred= regressor.predict(x\_train)

On executing the above lines of code, two variables named y\_pred and x\_pred will generate in the variable explorer options that contain salary predictions for the training set and test set.

### Output:

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

You can check the variable by clicking on the variable explorer option in the IDE, and also compare the result by comparing values from `y_pred` and `y_test`. By comparing these values, we can check how good our model is performing.

### Step: 4. visualizing the Training set results:

Now in this step, we will visualize the training set result. To do so, we will use the `scatter()` function of the `pyplot` library, which we have already imported in the pre-processing step. The **`scatter ()` function** will create a scatter plot of observations.

In the x-axis, we will plot the Years of Experience of employees and on the y-axis, salary of employees. In the function, we will pass the real values of training set, which means a year of experience `x_train`, training set of Salaries `y_train`, and color of the observations. Here we are taking a green color for the observation, but it can be any color as per the choice.

Now, we need to plot the regression line, so for this, we will use the **`plot()` function** of the `pyplot` library. In this function, we will pass the years of experience for training set, predicted salary for training set `x_pred`, and color of the line.

Next, we will give the title for the plot. So here, we will use the **`title()` function** of the **`pyplot`** library and pass the name ("Salary vs Experience (Training Dataset)").

After that, we will assign labels for x-axis and y-axis using **`xlabel()` and `ylabel()` function**.

Finally, we will represent all above things in a graph using `show()`. The code is given below:

1. `mtp.scatter(x_train, y_train, color="green")`
2. `mtp.plot(x_train, x_pred, color="red")`
3. `mtp.title("Salary vs Experience (Training Dataset)")`
4. `mtp.xlabel("Years of Experience")`
5. `mtp.ylabel("Salary(In Rupees)")`
6. `mtp.show()`

### Output:

By executing the above lines of code, we will get the below graph plot as an output.



## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM



In the above plot, we can see the real values observations in green dots and predicted values are covered by the red regression line. The regression line shows a correlation between the dependent and independent variable.

The good fit of the line can be observed by calculating the difference between actual values and predicted values. But as we can see in the above **plot, most of the observations are close to the regression line, hence our model is good for the training set.**

### Step: 5. visualizing the Test set results:

In the previous step, we have visualized the performance of our model on the training set. Now, we will do the same for the Test set. The complete code will remain the same as the above code, except in this, we will use `x_test`, and `y_test` instead of `x_train` and `y_train`.

Here we are also changing the color of observations and regression line to differentiate between the two plots, but it is optional.

1. `#visualizing the Test set results`
2. `mtp.scatter(x_test, y_test, color="blue")`
3. `mtp.plot(x_train, x_pred, color="red")`
4. `mtp.title("Salary vs Experience (Test Dataset)")`
5. `mtp.xlabel("Years of Experience")`
6. `mtp.ylabel("Salary(In Rupees)")`
7. `mtp.show()`

### Output:

By executing the above line of code, we will get the output as:



In the above plot, there are observations given by the blue color, and prediction is given by the red regression line. As we can see, most of the observations are close to the regression line, hence we can say our Simple Linear Regression is a good model and able to make good predictions.

## Multiple Linear Regression

In the previous topic, we have learned about Simple Linear Regression, where a single Independent/Predictor(X) variable is used to model the response variable (Y). But there may be various cases in which the response variable is affected by more than one predictor variable; for such cases, the Multiple Linear Regression algorithm is used.

Moreover, Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. We can define it as:

*Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.*

### Example:

Prediction of CO<sub>2</sub> emission based on engine size and number of cylinders in a car.

### Some key points about MLR:

- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- Each feature variable must model the linear relationship with the dependent variable.
- MLR tries to fit a regression line through a multidimensional space of data-points.

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

## MLR equation:

In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables  $x_1, x_2, x_3, \dots, x_n$ . Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

$$1. Y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n \quad \dots\dots\dots (a)$$

Where,

**Y= Output/Response variable**

**$b_0, b_1, b_2, b_3, b_n, \dots$  = Coefficients of the model.**

**$x_1, x_2, x_3, x_4, \dots$  = Various Independent/feature variable**

## Assumptions for Multiple Linear Regression:

- A **linear relationship** should exist between the Target and predictor variables.
- The regression residuals must be **normally distributed**.
- MLR assumes little or **no multicollinearity** (correlation between the independent variable) in data.

## Implementation of Multiple Linear Regression model using Python:

To implement MLR using Python, we have below problem:

### Problem Description:

We have a dataset of **50 start-up companies**. This dataset contains five main information: **R&D Spend, Administration Spend, Marketing Spend, State, and Profit for a financial year**. Our goal is to create a model that can easily determine which company has a maximum profit, and which is the most affecting factor for the profit of a company.

Since we need to find the Profit, so it is the dependent variable, and the other four variables are independent variables. Below are the main steps of deploying the MLR model:

1. **Data Pre-processing Steps**
2. **Fitting the MLR model to the training set**
3. **Predicting the result of the test set**

### Step-1: Data Pre-processing Step:

The very first step is **data pre-processing**, which we have already discussed in this tutorial. This process contains the below steps:

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

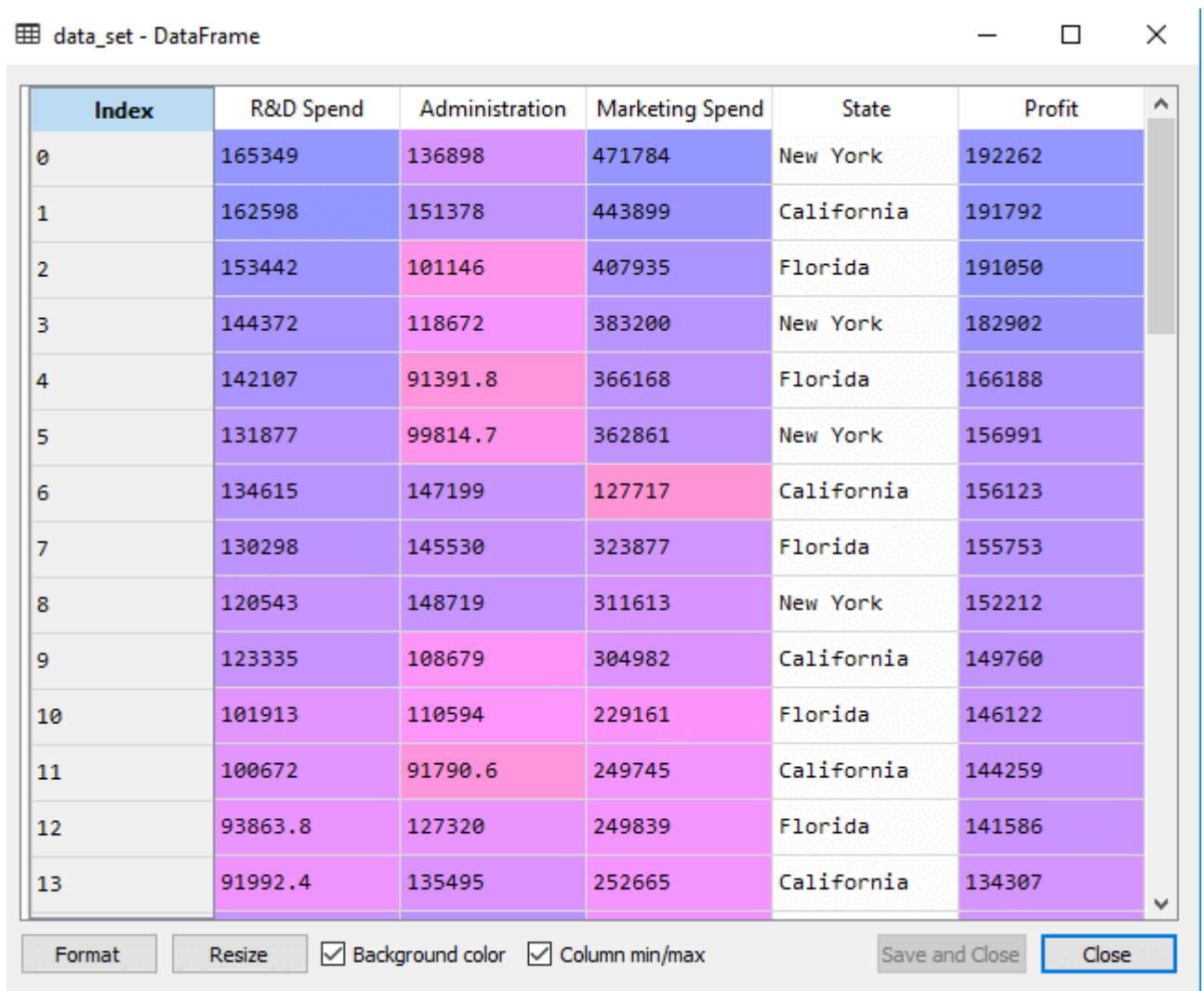
- **Importing libraries:** Firstly we will import the library which will help in building the model. Below is the code for it:

1. # importing libraries
2. **import** numpy as nm
3. **import** matplotlib.pyplot as mtp
4. **import** pandas as pd

- **Importing dataset:** Now we will import the dataset(50\_CompList), which contains all the variables. Below is the code for it:

1. #importing datasets
2. data\_set= pd.read\_csv('50\_CompList.csv')

**Output:** We will get the dataset as:



Index	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349	136898	471784	New York	192262
1	162598	151378	443899	California	191792
2	153442	101146	407935	Florida	191050
3	144372	118672	383200	New York	182902
4	142107	91391.8	366168	Florida	166188
5	131877	99814.7	362861	New York	156991
6	134615	147199	127717	California	156123
7	130298	145530	323877	Florida	155753
8	120543	148719	311613	New York	152212
9	123335	108679	304982	California	149760
10	101913	110594	229161	Florida	146122
11	100672	91790.6	249745	California	144259
12	93863.8	127320	249839	Florida	141586
13	91992.4	135495	252665	California	134307

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

In above output, we can clearly see that there are five variables, in which four variables are continuous and one is categorical variable.

### ○ Extracting dependent and independent Variables:

1. #Extracting Independent and dependent Variable
2. `x= data_set.iloc[:, :-1].values`
3. `y= data_set.iloc[:, 4].values`

**Output:**

**Out[5]:**

**Out[5]:**

### ADVERTISING

```
array([[165349.2, 136897.8, 471784.1, 'New York'],
       [162597.7, 151377.59, 443898.53, 'California'],
       [153441.51, 101145.55, 407934.54, 'Florida'],
       [144372.41, 118671.85, 383199.62, 'New York'],
       [142107.34, 91391.77, 366168.42, 'Florida'],
       [131876.9, 99814.71, 362861.36, 'New York'],
       [134615.46, 147198.87, 127716.82, 'California'],
       [130298.13, 145530.06, 323876.68, 'Florida'],
       [120542.52, 148718.95, 311613.29, 'New York'],
       [123334.88, 108679.17, 304981.62, 'California'],
       [101913.08, 110594.11, 229160.95, 'Florida'],
       [100671.96, 91790.61, 249744.55, 'California'],
       [93863.75, 127320.38, 249839.44, 'Florida'],
       [91992.39, 135495.07, 252664.93, 'California'],
       [119943.24, 156547.42, 256512.92, 'Florida'],
       [114523.61, 122616.84, 261776.23, 'New York'],
       [78013.11, 121597.55, 264346.06, 'California'],
       [94657.16, 145077.58, 282574.31, 'New York'],
       [91749.16, 114175.79, 294919.57, 'Florida'],
       [86419.7, 153514.11, 0.0, 'New York'],
       [76253.86, 113867.3, 298664.47, 'California'],
       [78389.47, 153773.43, 299737.29, 'New York'],
       [73994.56, 122782.75, 303319.26, 'Florida'],
       [67532.53, 105751.03, 304768.73, 'Florida'],
       [77044.01, 99281.34, 140574.81, 'New York'],
       [64664.71, 139553.16, 137962.62, 'California'],
       [75328.87, 144135.98, 134050.07, 'Florida'],
       [72107.6, 127864.55, 353183.81, 'New York'],
       [66051.52, 182645.56, 118148.2, 'Florida'],
       [65605.48, 153032.06, 107138.38, 'New York'],
       [61994.48, 115641.28, 91131.24, 'Florida'],
       [61136.38, 152701.92, 88218.23, 'New York'],
       [63408.86, 129219.61, 46085.25, 'California'],
       [55493.95, 103057.49, 214634.81, 'Florida'],
       [46426.07, 157693.92, 210797.67, 'California'],
       [46014.02, 85047.44, 205517.64, 'New York'],
       [28663.76, 127056.21, 201126.82, 'Florida'],
       [44069.95, 51283.14, 197029.42, 'California'],
       [20229.59, 65947.93, 185265.1, 'New York'],
       [38558.51, 82982.09, 174999.3, 'California'],
       [28754.33, 118546.05, 172795.67, 'California'],
```

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

```
[27892.92, 84710.77, 164470.71, 'Florida'],  
[23640.93, 96189.63, 148001.11, 'California'],  
[15505.73, 127382.3, 35534.17, 'New York'],  
[22177.74, 154806.14, 28334.72, 'California'],  
[1000.23, 124153.04, 1903.93, 'New York'],  
[1315.46, 115816.21, 297114.46, 'Florida'],  
[0.0, 135426.92, 0.0, 'California'],  
[542.05, 51743.15, 0.0, 'New York'],  
[0.0, 116983.8, 45173.06, 'California']], dtype=object)
```

As we can see in the above output, the last column contains categorical variables which are not suitable to apply directly for fitting the model. So we need to encode this variable.

### Encoding Dummy Variables:

As we have one categorical variable (State), which cannot be directly applied to the model, so we will encode it. To encode the categorical variable into numbers, we will use the **LabelEncoder** class. But it is not sufficient because it still has some relational order, which may create a wrong model. So in order to remove this problem, we will use **OneHotEncoder**, which will create the dummy variables. Below is code for it:

1. #Categorical data
2. from sklearn.preprocessing **import** LabelEncoder, OneHotEncoder
3. labelencoder\_x= LabelEncoder()
4. x[:, 3]= labelencoder\_x.fit\_transform(x[:,3])
5. onehotencoder= OneHotEncoder(categorical\_features= [3])
6. x= onehotencoder.fit\_transform(x).toarray()

Here we are only encoding one independent variable, which is state as other variables are continuous.

### Output:



	0	1	2	3	4	5
0	0	0	1	165349	136898	471784
1	1	0	0	162598	151378	443899
2	0	1	0	153442	101146	407935
3	0	0	1	144372	118672	383200
4	0	1	0	142107	91391.8	366168
5	0	0	1	131877	99814.7	362861
6	1	0	0	134615	147199	127717
7	0	1	0	130298	145530	323877
8	0	0	1	120543	148719	311613
9	1	0	0	123335	108679	304982
10	0	1	0	101913	110594	229161
11	1	0	0	100672	91790.6	249745
12	0	1	0	93863.8	127320	249839
13	1	0	0	91992.4	135495	252665

As we can see in the above output, the state column has been converted into dummy variables (0 and 1). **Here each dummy variable column is corresponding to the one State.** We can check by comparing it with the original dataset. The first column corresponds to the **California State**, the second column corresponds to the **Florida State**, and the third column corresponds to the **New York State**.

**Note:** We should not use all the dummy variables at the same time, so it must be 1 less than the total number of dummy variables, else it will create a dummy variable trap.

- Now, we are writing a single line of code just to avoid the dummy variable trap:

- #avoiding the dummy variable trap:
- `x = x[:, 1:]`

If we do not remove the first dummy variable, then it may introduce multicollinearity in the model.

x - NumPy array

	0	1	2	3	4
0	0	1	165349	136898	471784
1	0	0	162598	151378	443899
2	1	0	153442	101146	407935
3	0	1	144372	118672	383200
4	1	0	142107	91391.8	366168
5	0	1	131877	99814.7	362861
6	0	0	134615	147199	127717
7	1	0	130298	145530	323877
8	0	1	120543	148719	311613
9	0	0	123335	108679	304982
10	1	0	101913	110594	229161
11	0	0	100672	91790.6	249745
12	1	0	93863.8	127320	249839

Format    Resize    ☒ Background color

Save and Close    Close

As we can see in the above output image, the first column has been removed.

- Now we will split the dataset into training and test set. The code for this is given below:

1. # Splitting the dataset into training and test set.
2. from sklearn.model\_selection **import** train\_test\_split
3. x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

The above code will split our dataset into a training set and test set.

**Output:** The above code will split the dataset into training set and test set. You can check the output by clicking on the variable explorer option given in Spyder IDE. The test set and training set will look like the below image:

**Test set:**

# DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

y\_test - NumPy array

	0
0	103282
1	144259
2	146122
3	77798.8
4	191050
5	105008
6	81229.1
7	97483.6
8	110352
9	166188

Format

Resize

☒ Background color

x\_test - NumPy array

	0	1	2	3	4
0	1	0	66051.5	182646	118148
1	0	0	100672	91790.6	249745
2	1	0	101913	110594	229161
3	1	0	27892.9	84710.8	164471
4	1	0	153442	101146	407935
5	0	1	72107.6	127865	353184
6	0	1	20229.6	65947.9	185265
7	0	1	61136.4	152702	88218.2
8	1	0	73994.6	122783	303319
9	1	0	142107	91391.8	366168

Format

Resize

☒ Background color

Save and Close

Close

## Training set:

x\_train - NumPy array

	0	1	2	3	4
0	1	0	55493.9	103057	214635
1	0	1	46014	85047.4	205518
2	1	0	75328.9	144136	134050
3	0	0	46426.1	157694	210798
4	1	0	91749.2	114176	294920
5	1	0	130298	145530	323877
6	1	0	119943	156547	256513
7	0	1	1000.23	124153	1903.93
8	0	1	542.05	51743.2	0
9	0	1	65605.5	153032	107138
10	0	1	114524	122617	261776
11	1	0	61994.5	115641	91131.2

Format

Resize

☒ Background color

Save and Close

Close

y\_train - NumPy array

	0
0	96778.9
1	96479.5
2	105734
3	96712.8
4	124267
5	155753
6	132603
7	64926.1
8	35673.4
9	101005
10	129917
11	99937.6

Format

Resize

☒ Background color

Save and Close

Close

**Note:** In MLR, we will not do feature scaling as it is taken care by the library, so we don't need to do it manually.

## Step: 2- Fitting our MLR model to the Training set:

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

Now, we have well prepared our dataset in order to provide training, which means we will fit our regression model to the training set. It will be similar to as we did in [Simple Linear Regression](#) model. The code for this will be:

1. #Fitting the MLR model to the training set:
2. from sklearn.linear\_model **import** LinearRegression
3. regressor= LinearRegression()
4. regressor.fit(x\_train, y\_train)

### Output:

```
Out[9]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

Now, we have successfully trained our model using the training dataset. In the next step, we will test the performance of the model using the test dataset.

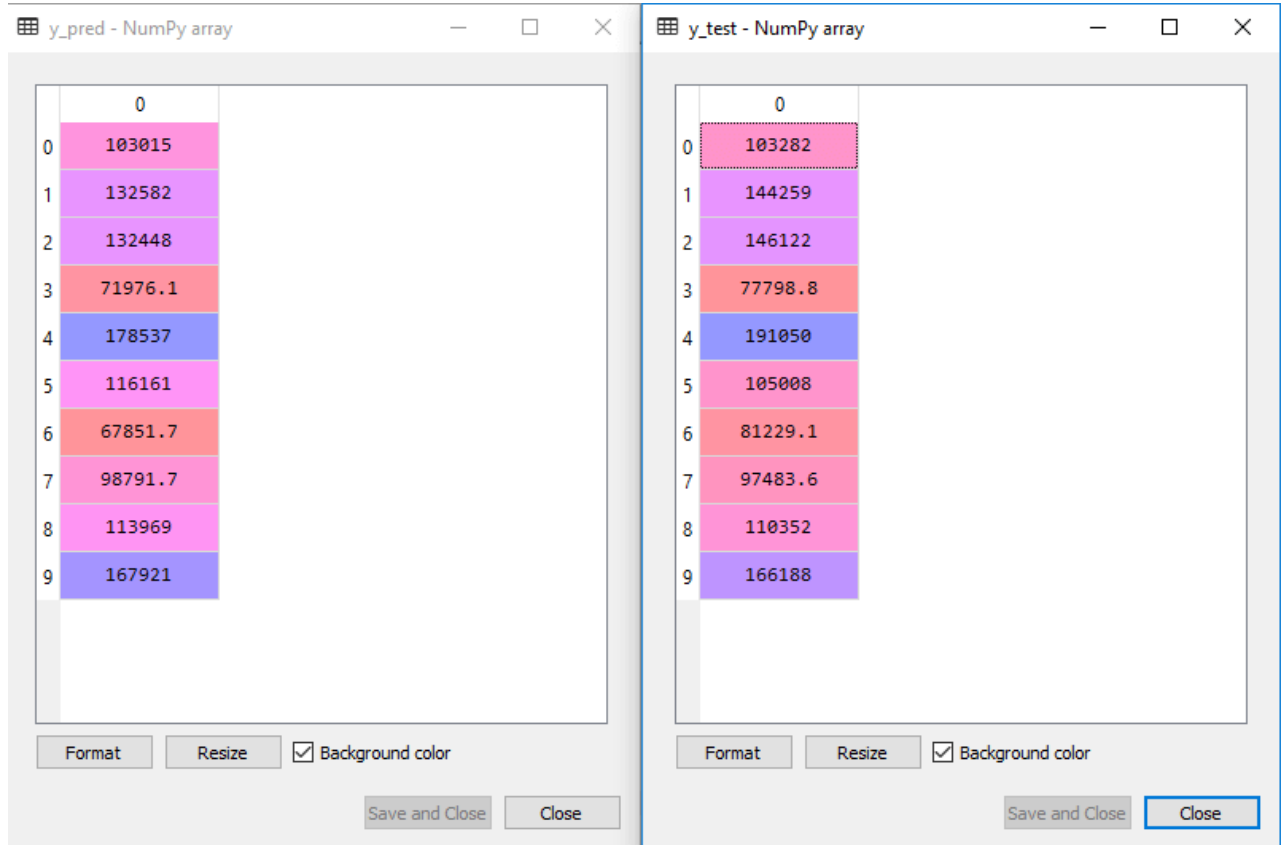
### Step: 3- Prediction of Test set results:

The last step for our model is checking the performance of the model. We will do it by predicting the test set result. For prediction, we will create a **y\_pred** vector. Below is the code for it:

1. #Predicting the Test set result;
2. y\_pred= regressor.predict(x\_test)

By executing the above lines of code, a new vector will be generated under the variable explorer option. We can test our model by comparing the predicted values and test set values.

### Output:



In the above output, we have predicted result set and test set. We can check model performance by comparing these two value index by index. For example, the first index has a predicted value of **103015\$** profit and test/real value of **103282\$** profit. The difference is only of **267\$**, which is a good prediction, so, finally, our model is completed here.

- We can also check the score for training dataset and test dataset. Below is the code for it:

1. `print('Train Score: ', regressor.score(x_train, y_train))`
2. `print('Test Score: ', regressor.score(x_test, y_test))`

**Output:** The score is:

```
Train Score: 0.9501847627493607
Test Score: 0.9347068473282446
```

**The above score tells that our model is 95% accurate with the training dataset and 93% accurate with the test dataset.**

**Note:** In the next topic, we will see how we can improve the performance of the model using the **Backward Elimination** process.

## Applications of Multiple Linear Regression:

There are mainly two applications of Multiple Linear Regression:

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

- Effectiveness of Independent variable on prediction:
- Predicting the impact of changes:

**Every time you run a regression, you MUST make sure that it meets all of the assumptions associated with that type of regression (OLS linear regression in this case).**

Every single time you run a regression, if you want to use the results of that regression for inference (learn something about a population using a sample from that population), you MUST make sure that it meets all of the assumptions associated with that type of regression. In this chapter, multiple OLS linear regression is the type of regression analysis we are using. You cannot trust your results until you are certain that your regression model meets these assumptions. This section will demonstrate how to conduct a series of diagnostic tests that help us determine whether our data and the fitted regression model meet the assumptions for OLS regression to be trustworthy or not for inference.

Here is a summary description of each OLS regression assumption that needs to be satisfied:

- **Residual errors have a mean of 0** – More precisely, the residual errors should have a population mean of 0.<sup>112</sup> In other words, we assume that the residuals are “coming from a normal distribution with mean of zero.”<sup>113</sup> This assumption cannot be tested empirically with our regression model and data. For our purposes, we will keep this assumption in mind and try to look out for any logical reasons in our regression model and data that might cause this assumption to be violated.
- **Residual errors are normally distributed** – Different experts or likely-experts describe this assumption differently. Some just say that the residuals should be normally distributed.<sup>114115</sup> Others say that the residuals must *come from* a distribution that is normally distributed.<sup>116</sup> For our purposes, we will simply test to see if our residuals are normally distributed or not.
- **Residual errors are uncorrelated with all independent variables** – There should not be any evidence that residuals and the independent variables included in the regression model are correlated with each other.<sup>117118</sup> To test this assumption, we will both visually and statistically check for correlation of the residuals with each independent variable (one at a time).
- **All residuals are independent of (uncorrelated with) each other** – There are many ways to phrase this assumption, some of which follow. The residual for any one observation in our data should not allow us to predict the residual for any other observation(s).<sup>119</sup> All residuals should be independent of each other.<sup>120121</sup> To test this assumption, we test our data for what is called serial correlation or autocorrelation.
- **Homoscedasticity of residual errors** – This assumption that our residuals are homoscedastic is often phrased or described in multiple ways, some of which follow. In OLS there cannot be any systematic pattern in the distribution of the residuals.<sup>122</sup> Residual errors have equal<sup>123</sup> and constant<sup>124</sup> variance. When residual errors do not satisfy this assumption, they are said to be heteroscedastic. We will test this assumption by performing both visual and statistical tests of heteroscedasticity.
- **Observations are independently and identically distributed** – This assumption is sometimes referred to as the IID assumption.<sup>125</sup> This means that no observation should

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

have an influence on another observation(s). The observations should be randomly sampled and not influence any other observation's behaviors, results, and/or measurable characteristics (such as dependent and independent variables that we might use in a regression model). We will test this assumption by considering both our research design and the structure of the data and making sure that they likely do not violate this assumption.

- **Linearity of data** – The dependent variable must be linearly related to all independent variables. We will test this assumption by both visual and statistical tests.
- **No multicollinearity** – Independent variables cannot completely (perfectly) predict or correlate with each other. Independent variables cannot partially predict or correlate with each other to an extent that will bias or problematically influence the estimation of regression parameters.

## Regression Diagnostics

---

### Model Assumptions

The model fitting is just the first part of the story for regression analysis since this is all based on certain assumptions. Regression diagnostics are used to evaluate the model assumptions and investigate whether or not there are observations with a large, undue influence on the analysis. Again, the assumptions for linear regression are:

1. **Linearity**: The relationship between X and the mean of Y is linear.
2. **Homoscedasticity**: The variance of residual is the same for any value of X.
3. **Independence**: Observations are independent of each other.
4. **Normality**: For any fixed value of X, Y is normally distributed.

Before we go further, let's review some definitions for problematic points.

- **Outliers**: an outlier is defined as an observation that has a large residual. In other words, the observed value for the point is very different from that predicted by the regression model.
- **Leverage points**: A leverage point is defined as an observation that has a value of x that is far away from the mean of x.
- **Influential observations**: An influential observation is defined as an observation that changes the slope of the line. Thus, influential points have a large influence on the fit of the model. One method to find influential points is to compare the fit of the model with and without each observation.

Weighted least squares (WLS), also known as weighted linear regression, is a **generalization of ordinary least squares and linear regression in which knowledge of the variance of observations is incorporated into the regression.**

Some key points regarding **weighted least squares** are:

1. The difficulty, in practice, is determining estimates of the error variances (or standard deviations).
2. Weighted least squares estimates of the coefficients will usually be nearly the same as the "ordinary" unweighted estimates. In cases where they differ substantially, the

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

procedure can be iterated until estimated coefficients stabilize (often in no more than one or two iterations); this is called iteratively reweighted least squares.

3. In some cases, the values of the weights may be based on theory or prior research.
4. In designed experiments with large numbers of replicates, weights can be estimated directly from sample variances of the response variable at each combination of predictor variables.
5. The use of weights will (legitimately) impact the widths of statistical intervals.

## What is a Generalized Linear Model?

Generalized Linear Model (GLiM, or GLM) is an advanced statistical modelling technique formulated by John Nelder and Robert Wedderburn in 1972. It is an umbrella term that encompasses many other models, which allows the response variable  $y$  to have an error distribution other than a normal distribution. The models include Linear Regression, Logistic Regression, and Poisson Regression.

In a Linear Regression Model, the response (aka dependent/target) variable ' $y$ ' is expressed as a linear function/linear combination of all the predictors ' $X$ ' (aka independent/regression/explanatory/observed variables). The underlying relationship between the response and the predictors is linear (i.e. we can simply visualize the relationship in the form of a straight line). Also, the error distribution of the response variable should be normally distributed. Therefore we are building a linear model.

GLM models allow us to build a linear relationship between the response and predictors, even though their underlying relationship is not linear. This is made possible by using a link function, which links the response variable to a linear model. Unlike Linear Regression models, the error distribution of the response variable need not be normally distributed. The errors in the response variable are assumed to follow an exponential family of distribution (i.e. normal, binomial, Poisson, or gamma distributions). Since we are trying to generalize a linear regression model that can also be applied in these cases, the name Generalized Linear Models.

### Why GLM?

Linear Regression model is not suitable if,

- The relationship between  $X$  and  $y$  is not linear. There exists some non-linear relationship between them. For example,  $y$  increases exponentially as  $X$  increases.
- Variance of errors in  $y$  (commonly called as Homoscedasticity in Linear Regression), is not constant, and varies with  $X$ .
- Response variable is not continuous, but discrete/categorical. Linear Regression assumes normal distribution of the response variable, which can only be applied on a continuous data. If we try to build a linear regression model on a discrete/binary  $y$  variable, then the linear regression model predicts negative values for the corresponding response variable, which is inappropriate.

In the below graph, we can see the response is either 0 or 1. When  $X < 5000$ ,  $y$  is 0, and when  $X \geq 5000$ ,  $y$  is 1

For Example, Consider a linear model as follows:

A simple example of a mobile price in an e-commerce platform:



## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

Price = 12500 + 1.5\*Screen size – 3\*Battery Backup (less than 4hrs)

Data available for,

- Price of the mobile
- Screen size (in inches)
- Is battery backup less than 4hrs – with values either as ‘yes’, or ‘no’.

In this example, if the screen size increases by 1 unit, then the price of the mobile increases by 1.5 times the default price, keeping the intercept (12500) and Battery Backup values constant. Likewise, if the Battery Backup of less than 4hrs is ‘yes’, then the mobile price reduces by three times the default price. If the Battery Backup of less than 4hrs is ‘no’, then the mobile price is unaffected, as the term (3\*Battery Backup) becomes 0 in the linear model. The intercept 12500 indicates the default price for a standard value of screen size. This is a valid model.

However, if we get a model as below:

Price = 12500 + 1.5\*Screen size + 3\*Battery Backup(less than 4hrs)

Here, if the battery backup less than 4 hrs is ‘yes’, then the model is saying the price of the phone increases by three times. Clearly, from practical knowledge, we know this is incorrect. There will be less demand for such mobiles. These are going to be very old mobiles, which when compared to the current range of mobiles with the latest features, is going to be very less in price. This is because the relationship between the two variables is not linear, but we are trying to express it as a linear relationship. Hence, an invalid model is built.

Similarly, if we are trying to predict if a particular phone will be sold or not, using the same independent variables, but the target is we are trying to predict if the phone will sell or not, so it has only binary outcomes.

Using Linear Regression, we get a model like,

Sales = 12500 + 1.5\*Screen size – 3\*Battery Backup(less than 4hrs)

This model doesn’t tell us if the mobile will be sold or not, because the output of a linear regression model is continuous value. It is possible to get negative values as well as the output. It does not translate to our actual objective of whether phones having some specifications based on the predictors, will sell or not (binary outcome).

Similarly if we are also trying to see what is the number of sales of this mobile that will happen in the next month, a negative value means nothing. Here, the minimum value is 0 (no sale happened), or a positive value corresponding to the count of the sales. Having the count as a negative value is not meaningful to us.

Assumptions of GLM

Similar to Linear Regression Model, there are some basic assumptions for Generalized Linear Models as well. Most of the assumptions are similar to Linear Regression models, while some of the assumptions of Linear Regression are modified.

Data should be independent and random (Each Random variable has the same probability distribution).

The response variable y does not need to be normally distributed, but the distribution is from an exponential family (e.g. binomial, Poisson, multinomial, normal)

The original response variable need not have a linear relationship with the independent variables, but the transformed response variable (through the link function) is linearly dependent on the independent variables

Ex., Logistic Regression Equation,  $\text{Log odds} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ ,

where  $\beta_0, \beta_1, \beta_2$  are regression coefficient, and  $X_1, X_2$  are the independent variables

Feature engineering on the Independent variable can be applied i.e instead of taking the original raw independent variables, variable transformation can be done, and the transformed

## DRONACHARYA COLLEGE OF ENGINEERING GURUGRAM

independent variables, such as taking a log transformation, squaring the variables, reciprocal of the variables, can also be used to build the GLM model.

- Homoscedasticity (i.e constant variance) need not be satisfied. Response variable Error variance can increase, or decrease with the independent variables.
- Errors are independent but need not be normally distributed

Components of GLM

There are 3 components in GLM.

- Systematic Component/Linear Predictor:

It is just the linear combination of the Predictors and the regression coefficients.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2$$

- Link Function:

Represented as  $\eta$  or  $g(\mu)$ , it specifies the link between a random and systematic components. It indicates how the expected/predicted value of the response relates to the linear combination of predictor variables.

- Random Component/Probability Distribution:

It refers to the probability distribution, from the family of distributions, of the response variable.

The family of distributions, called an exponential family, includes normal distribution, binomial distribution, or poisson distribution.

Below summarizes the table of Probability Distribution, and their corresponding Link function

<b><i>Probability Distribution</i></b>	<b><i>Link Function</i></b>
Normal Distribution	Identity function
Binomial Distribution	Logit/Sigmoid function
Poisson Distribution	Log function (aka log-linear, log-link)