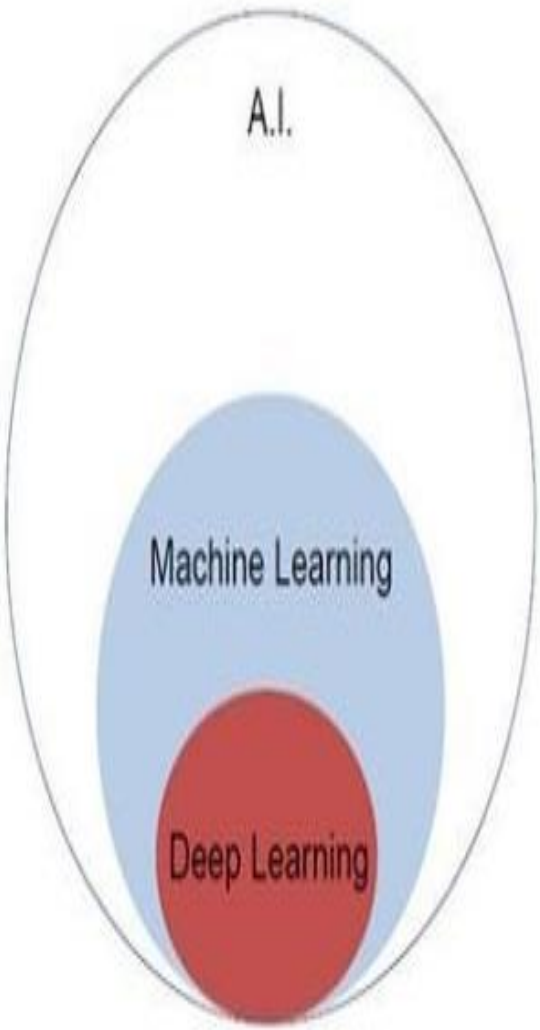
Machine Learning using Python

Introduction

- Artificial Intelligence includes the simulation process of human intelligence by computer systems. Examples of artificial intelligence include learning, reasoning and self-correction. Applications of AI include speech recognition, expert systems, image recognition and computer vision.

- Machine learning is the branch of artificial intelligence, which deals with systems and algorithms that can learn any new data and data patterns. A machine-learning system is trained rather than explicitly programmed.

- Deep learning is a part of machine learning.



Introduction to ML

- "Machine Learning is the field of study which gives computers the ability to learn without being explicitly programmed."

  -- Arthur Samuels, 1959

- **Traditional Programming:**

- **Machine Learning:**

Machine Learning Model

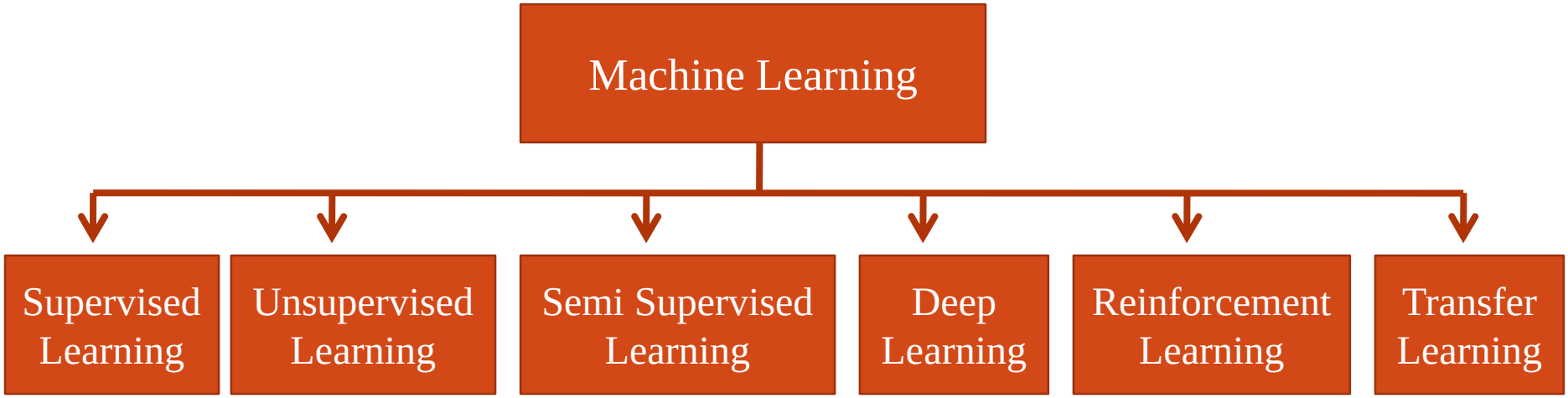
- “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

  -- Tom Michells, 1997

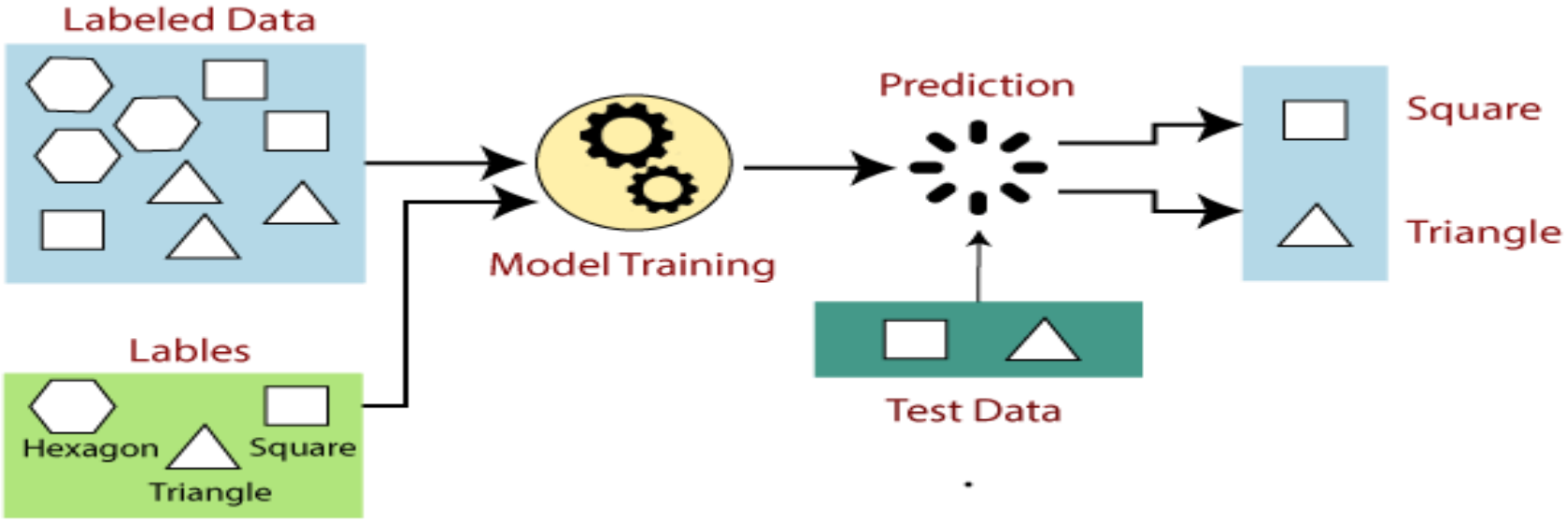
- The main components of any Machine Learning algorithm: **Task(T), Performance(P) and experience (E).**
  - **Task:** We may define the task (T) as any real-world problem that needs to be solved. The examples of ML based tasks are Classification, Regression, Clustering and so on.
  - **Experience:** It is the knowledge gained from inputs provided to the algorithm or model. Once provided with the dataset, the model will run iteratively and try to learn some inherent pattern. The learning thus acquired is called experience(E).
  - **Performance:** It is the measure which tells whether ML algorithm is performing as per expectation or not. (P) is basically a quantitative metric that tells how a model is performing the task, T, using its experience, E. Metrics that help to understand the ML performance are F1 score, confusion matrix, precision, recall and so on.

Types of ML Algorithms

- Machine Learning Algorithms are broadly classified into the following categories:



Supervised Learning



- Supervised learning includes a procedure where the training set and its associated output i.e. labels are given as input to the system.
- After completion of training, the accuracy of each model is measured with respect to the test set or validation set, which is complete disjoint from the training set.

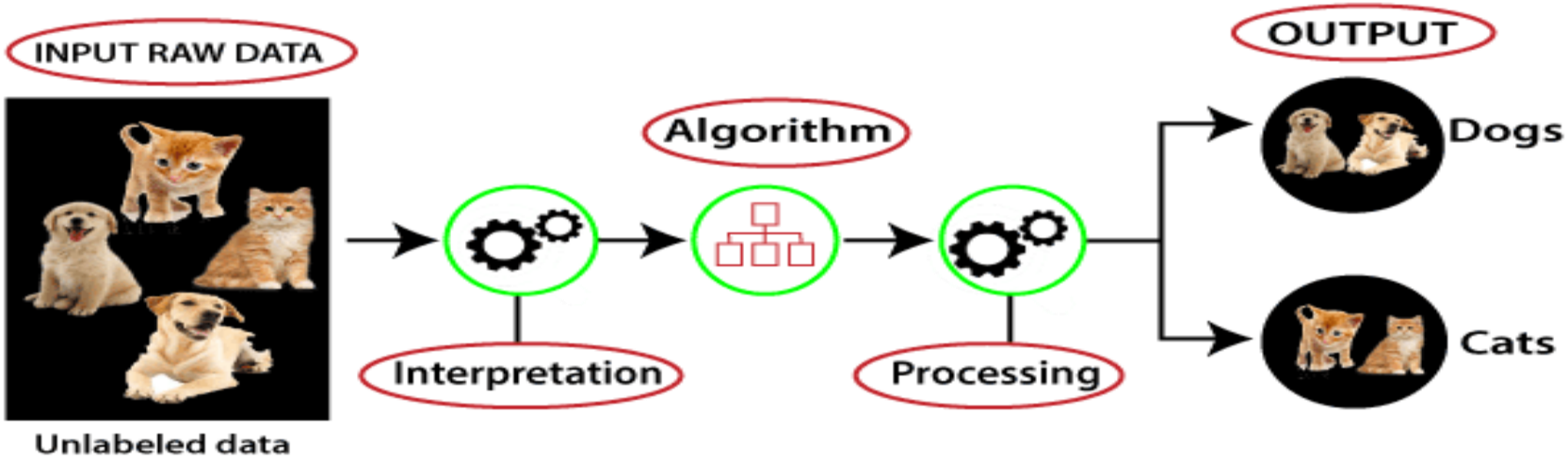
Supervised Learning

- Supervised learning can be grouped further in two categories of algorithms:

1. **Classification:** The key objective of classification problem is to classify objects of similar type. Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Example: Naïve Bayes Classification, Decision Tree, Logistic Regression, Support Vector Machine(SVM) and so on

2. **Regression:** Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Example: Linear Regression

Unsupervised Learning



- Unsupervised learning includes a procedure where the training data are not labelled.
- In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the data.

Unsupervised Learning

- Unsupervised learning can be grouped further in two categories of algorithms:

1. **Clustering:** Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Example: K-Means, Hierarchical clustering and so on.

2. **Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database.

Semi-supervised Learning

- The most basic disadvantage of any Supervised Learning algorithm is that the dataset has to be hand-labelled either by a Machine Learning Engineer or a Data Scientist. This is a very costly process, especially when dealing with large volumes of data. The most basic disadvantage of any Unsupervised Learning is that it's application spectrum is limited.
- To counter these disadvantages, the concept of Semi-Supervised Learning was introduced. Semi-supervised learning uses a combination of a small amount of labelled data and a large amount of unlabeled data to train models. This approach combines both the concept of Supervised and Unsupervised Learning.
- Example: Text document classifier. This is the type of situation where semi-supervised learning is ideal because it would be nearly impossible to label all the words in a text document.

Loading and Processing of Data

- **Header Files:**

  import pandas as pd

  import numpy as np

  import matplotlib.pyplot as plt

  import seaborn as sns

- **Reading a CSV File in a DataFrame :**

  df=pd.read_csv(“xyz.csv")

  print(df) //print the entire CSV file as a DataFrame

  print(df.shape) //display no of rows and columns present in the CSV file

  print(df.info())// shows different data types of all the columns of the file

  print(df.describe()) // it shows different statistical information of the CSV file such as: mean, standard deviation, minimum, maximum

  print(df.head()) // print first five rows from the CSV file

Pre-Processing of Data

print(df.isnull().sum()) // Check for any missing values in the dataset

Regression Analysis

- Regression analysis is a statistical method to model the relationship between a dependent (target) and one or more independent (predictor) variables.
- More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.

- **Terminologies Related to the Regression Analysis:**

➢ **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.

➢ **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.

Regression Analysis

- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.

- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because, when independent variables are correlated, it indicates that changes in one variable are associated with shifts in another variable. The stronger the correlation, the more difficult it is to change one variable without changing another.

- **Under-fitting and Over-fitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Over-fitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **under-fitting**.

Linear Regression

- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables. Which means it finds how the value of the dependent variable is changing according to the value of the independent variables.

- Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x_1+a_2x_2+\ldots+a_nx_n+ \varepsilon$$

- Here,

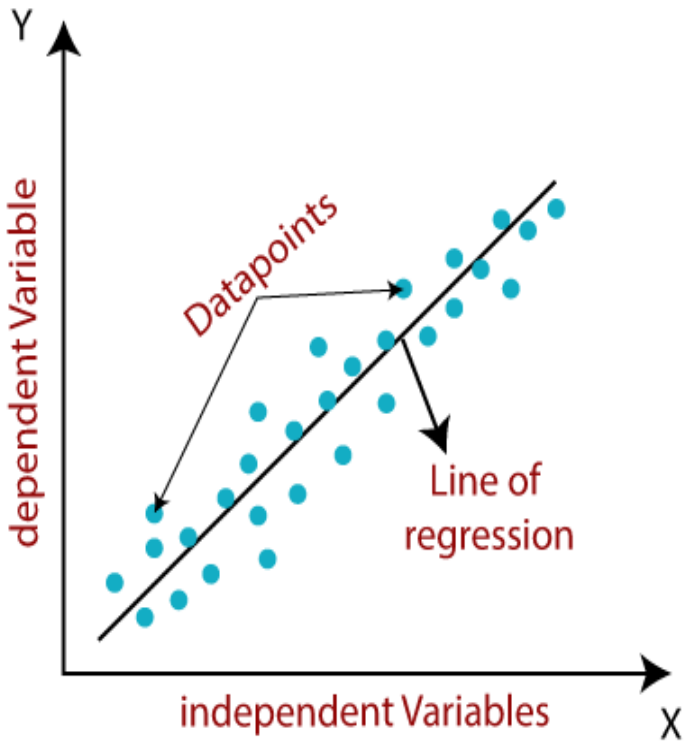
  y = Dependent Variable (Target Variable)

  $x_1,x_2,x_3,\ldots, x_n$ = Independent Variables (Predictor Variables)

  a_0 = intercept of the line (Gives an additional degree of freedom)

  $a_1,a_2,a_3,\ldots,a_n$ = Linear regression coefficients

  ε = random error



Best Fit Line in Linear Regression

- The main objective of Linear Regression model is to find the best fit line that should minimize the error between predicted values and actual values.
- **Cost Function:**

➢ Cost function optimizes the regression coefficients or weights.

➢ The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.

➢ Here we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

➢ MSE : $$MSE = 1\frac{1}{N}\sum_{i=1}^{n}(y_i - (a_1 x_i + a_0))^2$$

Where, N =Total number of observation

yi = Actual value

$(a_1 x_i + a_0)$= Predicted value.

Best Fit Line in Linear Regression

- **Residuals:** The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

- **Gradient Descent:**
- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Performance of Linear Regression

- The Goodness of fit determines how the line of regression fits the set of observations.
- **R-squared method:**
  - R-squared is a statistical method that determines the goodness of fit.
  - The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
  - It can be calculated from the below formula:

$$R^2 = 1 - \frac{SS_r}{SS_t}$$

SS_t : is the total sum of errors if we take the mean of the observed values as the predicted value.

$$SS_t = \sum_{i=1}^{m} (y^i - \bar{y})^2$$

ss_t = np.sum((y_actual - np.mean(y_actual))**2)

SS_r is the sum of the square of residuals. ss_r = np.sum((y_pred - y_actual)**2)

$$SS_r = \sum_{i=1}^{m} (h(x^i) - y^i)^2$$

Simple Linear Regression

- Simple Linear Regression is a type of Regression that models the relationship between a dependent variable and a single independent variable.
- It can be mathematically represented as:

$$y= a_0+a_1x+ \varepsilon$$

- Where,

  a0= It is the intercept of the Regression line (can be obtained putting x=0)
  a1= It is the slope of the regression line,

  ε = The error term. (For a good model it will be negligible)

Implementation of Simple Linear Regression using Python

- Step-1: Read Data :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection
   import train_test_split
from sklearn.linear_model import
   LinearRegression
df= pd.read_csv('salary_data.csv')
print(df)
```

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029

Implementation of Simple Linear Regression using Python

- Step-2: Pre-processing Data :

```
x= data_set.iloc[:,:-1].values  # It selects last column with all rows
y= data_set.iloc[:, 1].values  # It selects column-1 with all rows
```

- Step-3: Splitting the dataset into training and test set:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, rando
                                                                    m_state=0)
```

test_size=1/3: Here we will split our dataset (30 observations) into 2 parts (training set, test set) and the ratio of test set compare to dataset is 1/3 (10 observations will be put into the test set).

random_state=0: This is the seed for the random number generator. If we leave it blank or 0, then np.random is used for randomly choosing the elements for training set and test set.

Implementation of Simple Linear Regression using Python

- Step-4: Build the Regression Model:

from sklearn.linear_model import LinearRegression # Here we are importing the LinearRegression class of the linear_model library from the scikit learn.

linreg= LinearRegression()

linreg.fit(x_train, y_train)

- Step-5: Prediction of Test set Result:

test_pred= linreg.predict(x_test)

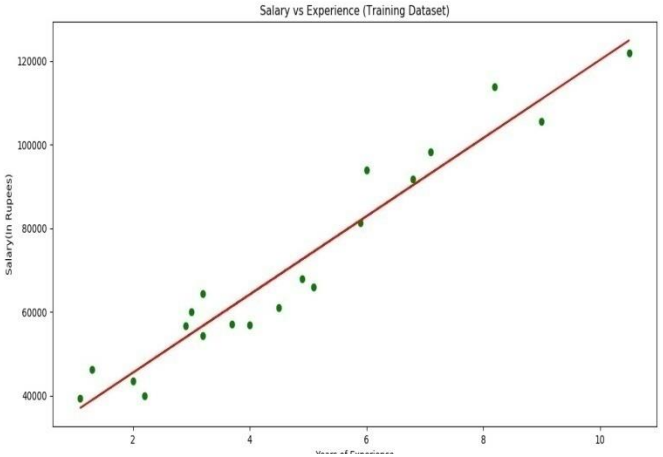
train_pred= linreg.predict(x_train)

Here we create a prediction vector test_pred, and train_pred, which contains predictions of test dataset, and training dataset respectively.

Implementation of Simple Linear Regression using Python

- Step-6: Visualizing the training set results:

```
plt.scatter(x_train, y_train, color="green")
plt.plot(x_train, train_pred, color="red")
plt.title("Salary vs Experience (Training set)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary(In Rupees)")
```



- Step-7:Visualizing the test set results:

```
plt.scatter(x_test, y_test, color="blue")
plt.plot(x_train, train_pred, color="red")
plt.title("Salary vs Experience (Test set)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary(In Rupees)")
```

Implementation of Simple Linear Regression using Python

- Step-8: Predicting the Result:

  Method 1:

```
y_pred= linreg.predict(10)
print(y_pred) # 120275.61667525
```

  Method 2:

```
print(linreg.coef_) # It gives the value of a1
print(linreg.intercept_) # It gives the value of a0
def cal_salary(x):
        y = ((x*linreg.coef_) + linreg.intercept_)
        return y
print(cal_salary(10)) # 120275.61667525
```

Both of them predicts the salary for 10 years experience

Multiple Linear Regression

- Multiple Linear Regression is a type of Regression that models the relationship between a dependent variable(y) and two or more independent variables(x1,x2,x3,…,xn).
- It can be mathematically represented as:

$$\mathbf{y= a_0+a_1x_1+a_2x_2+a_3x_3+…+a_nx_n + \varepsilon}$$

- Where,

  a_0= It is the intercept of the Regression line
  a1,a2,a3,…,an= Coefficients of the model,

  x1,x2,x3,…,xn = Different independent/feature variables

Implementation of Multiple Linear Regression using Python

- Step-1: Read the data:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df= pd.read_csv('insurance.csv')
print(df)
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692
10	25	male	26.220	0	no	northeast	2721.32080
11	62	female	26.290	0	yes	southeast	27808.72510
12	23	male	34.400	0	no	southwest	1826.84300
13	56	female	39.820	0	no	southeast	11090.71780
14	27	male	42.130	0	yes	southeast	39611.75770
15	19	male	24.600	1	no	southwest	1837.23700

Implementation of Multiple Linear Regression using Python

- Step-2: Pre-processing of data:
  - Conversion of categorical data into numerical data or dummy variables:

```
def map_sex(column):
    ls=[]
    for x in column:
        if x=='female':
            ls.append(1)
        else:
            ls.append(0)
    return ls
df['sex_norm']=map_sex(df['sex'])
```

Note: Similarly we can convert ‘smoker’ column into (yes=1) and (no=0)

	age	sex	bmi	...	region	charges	sex_norm
0	19	female	27.900	...	southwest	16884.92400	1
1	18	male	33.770	...	southeast	1725.55230	0
2	28	male	33.000	...	southeast	4449.46200	0
3	33	male	22.705	...	northwest	21984.47061	0
4	32	male	28.880	...	northwest	3866.85520	0
5	31	female	25.740	...	southeast	3756.62160	1
6	46	female	33.440	...	southeast	8240.58960	1
7	37	female	27.740	...	northwest	7281.50560	1
8	37	male	29.830	...	northeast	6406.41070	0
9	60	female	25.840	...	northwest	28923.13692	1
10	25	male	26.220	...	northeast	2721.32080	0
11	62	female	26.290	...	southeast	27808.72510	1
12	23	male	34.400	...	southwest	1826.84300	0
13	56	female	39.820	...	southeast	11090.71780	1
14	27	male	42.130	...	southeast	39611.75770	0

Implementation of Multiple Linear Regression using Python

- Step-2: Pre-processing of data:
  - Conversion of categorical data into numerical data or dummy variables:

Hot encoding is a procedure of transforming categorical variables as binary vectors.

```
region_dummy=pd.get_dummies (df['region'])
df=pd.concat([df,region_dummy], axis=1)
df.drop(['sex','smoker','region'], inplace=True,axis=1)
print(df)
```

	age	bmi	children	...	northwest	southeast	southwest
0	19	27.900	0	...	0	0	1
1	18	33.770	1	...	0	1	0
2	28	33.000	3	...	0	1	0
3	33	22.705	0	...	1	0	0
4	32	28.880	0	...	1	0	0
5	31	25.740	0	...	0	1	0
6	46	33.440	1	...	0	1	0
7	37	27.740	3	...	1	0	0
8	37	29.830	2	...	0	0	0
9	60	25.840	0	...	1	0	0
10	25	26.220	0	...	0	0	0
11	62	26.290	0	...	0	1	0
12	23	34.400	0	...	0	0	1
13	56	39.820	0	...	0	1	0
14	27	42.130	0	...	0	1	0
15	19	24.600	1	...	0	0	1

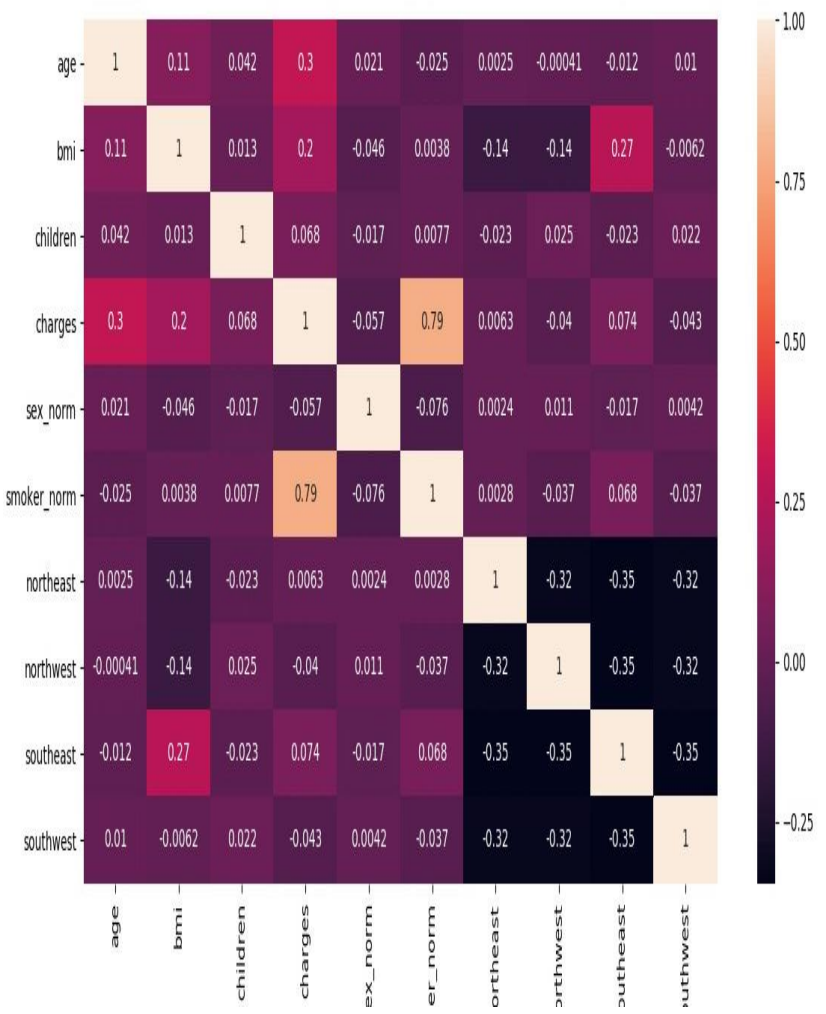
Implementation of Multiple Linear Regression using Python

- Step-2: Pre-processing of data:

➢ Create a correlation matrix that measures the linear relationships between the variables:

```
cor_matrix = df.corr()
sns.heatmap(data=cor_matrix, annot=True)
```

Note: An important point in selecting features for a linear regression model is to check for multi-co-linearity. If any feature pairs are strongly correlated to each other, we should not select both these features together for training the model.



Implementation of Multiple Linear Regression using Python

- Step-3: Extracting independent and dependent variables :

```
x=df[['age','bmi','children','sex_norm','smoker_norm','northwest','southeast','
   southwest']]
y=df['charges']
```

- Step-4: Splitting the dataset into training and test set:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, rando
                                                              m_state=0)
```

test_size=1/3: Here we will split our dataset (30 observations) into 2 parts (training set, test set) and the ratio of test set compare to dataset is 1/3 (10 observations will be put into the test set).

random_state=0: This is the seed for the random number generator. If we leave it blank or 0, then np.random is used for randomly choosing the elements for training set and test set.

Implementation of Simple Linear Regression using Python

- Step-5: Build the Regression Model:

from sklearn.linear_model import LinearRegression # Here we are importing the LinearRegression class of the linear_model library from the scikit learn.

linreg= LinearRegression()

linreg.fit(x_train, y_train)

- Step-6: Prediction of Test set Result:

test_pred= linreg.predict(x_test)

train_pred= linreg.predict(x_train)

Here we create a prediction vector test_pred, and train_pred, which contains predictions of test dataset, and training dataset respectively.

Implementation of Simple Linear Regression using Python

Step-7: Model Evaluation using R2-Score:

Model evaluation for training set:

```
r2 = r2_score(y_train, train_pred)
print("The model performance for training set:")
print("--------------------------------")
print('R2 score is {}'.format(r2))
```

The model performance for training Set:

R2 score is 0.728752734590003

Model evaluation for testing set:

```
r2 = r2_score(y_test, test_pred)
print("The model performance for testing set:")
print("--------------------------------")
print('R2 score is {}'.format(r2))
```

The model performance for testing set

R2 score is 0.787711930313088

Linear Regression using Gradient Descent

- **Loss Function:** $$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)^2$$ **(MSE)**

Here y_i is the actual value and $\bar{y}_i$ is the predicted value.

- It can be re-written as: $$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$
- Our goal is to minimize this loss function by obtaining the most accurate value of **m** and **c**.
- **Gradient descent** is an iterative optimization algorithm to find the minimum of a function. Here that function is our Loss Function.
- It is done by a random selection of values of coefficient **(m and c)** and then iteratively update those values to reach the minimum of cost function.

Linear Regression using Gradient Descent

- The person decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal.
- Here we are applying gradient descent to find m and c.

- Initially let m = 0 and c = 0. Let L be our learning rate. This controls how much the value of **m** changes with each step. L could be a small value like 0.0001 for good accuracy.
- Calculate the partial derivative of the loss function with respect to **m** ($\mathbf{D_m}$):

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

- Similarly lets find the partial derivative with respect to **c** ($\mathbf{D_c}$):

$$D_c = \frac{-2}{n}\sum_{i=0}^{n} (y_i - \bar{y}_i)$$

Linear Regression using Gradient Descent

- Now we update the current value of **m** and **c** using the following equation:

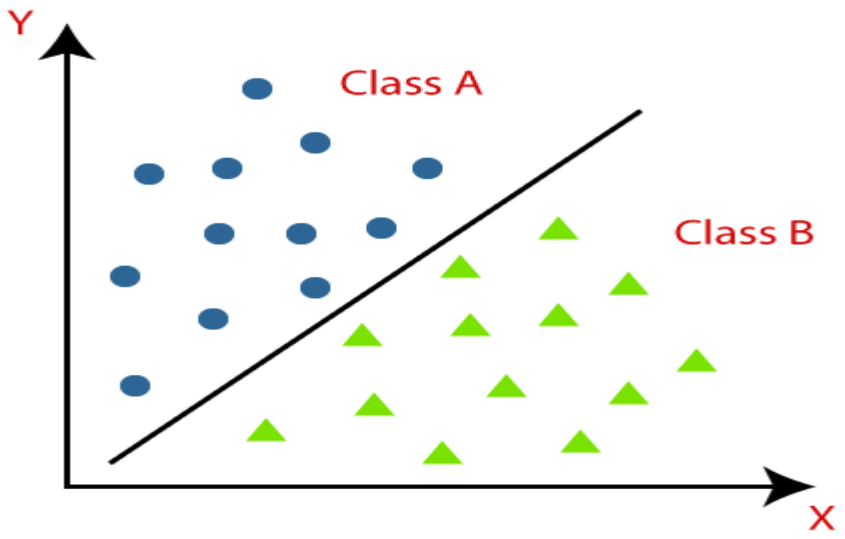
$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

- We repeat this process until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of **m** and **c** that we are left with now will be the optimum values.

Classification Algorithms

- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
- Unlike regression, the output variable of Classification is categorical, not a value, such as Yes-No, Male-Female, True-false etc.



Classification Algorithms

- There are two types of Classifications:
- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier. Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. Example: Classifications of types of fruits, Classification of types of music.
- **Types of ML Classification Algorithms:**

  Classification Algorithms can be further divided into the Mainly two category:
- **Linear Models**
  - Logistic Regression
  - Support Vector Machines
- **Non-linear Models**
  - K-Nearest Neighbours
  - Kernel SVM
  - Naïve Baye's
  - Decision Tree Classification
  - Random Forest Classification

Evaluation of Classification Model

- **Confusion Matrix:**

	Actual Positive	Actual negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Suppose a ML Algorithm identifies 8 dogs in a picture containing 10 cats and 12 dogs. Of the 8 identified as dogs, 5 actually are dogs (true positives), while the other 3 are cats (false positives). 7 dogs were missed (false negatives), and 7 cats were correctly excluded (true negatives).

- **Accuracy:** In classification, the commonly used metric is accuracy which is defined as: (TP + TN) / (TP + FP + FN +TN)

The algorithm's Accuracy is: (5 + 7) / (5 + 3 + 7 + 7) = 12 / 22 = .5454

Evaluation of Classification Model

- **Precision:** Precision is defined as: TP / (TP + FP).

  This fraction shows the ratio of the true positive prediction among all positive predictions.

  The algorithm's Precision is: 5 / (5 + 3) = 5 / 8 = .625

- **Recall/Sensitivity/True Positive Rate:** In classification, recall or true positive rate shows how many of the positives returned by the ML Algorithm. It is defined as:

  $$\text{Recall(TPR)} = TP / (TP + FN)$$

  The algorithm's Recall is: 5 / (5 + 7) = 5 / 12 = .4166

- **F1-Score:** Precision and recall are often combined into a single measure using their harmonic mean, known as the F1-score.

  $$\text{F1-Score} = 2.\text{recall}.\text{precision} / (\text{recall} + \text{precision})$$

  $$= 2TP / 2TP + FP + FN$$

  The algorithm's f1-Score is: 2*5 / 2*5 + 3 + 7 = 10 / 20 = .5

Evaluation of Classification Model

- **AUC-ROC Curve:**
  - ROC curve stands for **Receiver Operating Characteristics Curve** and AUC stands for **Area Under the Curve**.
  - It is a graph that shows the performance of the classification model at different thresholds.
  - The ROC curve is plotted with TPR and FPR, where TPR (True Positive Rate) on Y-axis and FPR(False Positive Rate) on X-axis.
  - **Specificity/True Negative Rate:** It tells us what proportion of the negative class got correctly classified.

    Specificity = TN / (TN + FP)
  - **False Positive Rate:** It tells us what proportion of the negative class got incorrectly classified by the classifier.

    FPR = FP / (TN + FP) = 1- Specificity

Naïve Bayes Classification Algorithm

- Naïve Bayes algorithm is a supervised learning classification algorithm based on Bayes theorem.
- The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:
  - **Naïve:** It is called Naïve because it assumes that the occurrence of a particular feature is independent of the occurrence of other features.
  - **Bayes:** This algorithm uses Bayes' Theorem to predict the probability that a given feature set belongs to a particular class.

$$P(C / F) = \frac{P(C) * P(F / C)}{P(F)}$$

Where,

P(C/F) is the posterior probability of hypothesis C (target) based on observed event F(features).

P(C) is the prior probability of hypothesis before observing the evidence.

P(F/C) is the likelihood probability of the evidence given that the probability of a hypothesis is true.

P(F) is the marginal probability of evidence.

Naïve Bayes Classification Algorithm

- Let's understand it using an example. Suppose we have a dataset on **Weather conditions** and corresponding target variable **Rain**. So using this dataset we need to decide that whether rain will occur or not on a particular day according to the weather conditions.

DAY	TEMPERA	HUMIDITY	CLOUDS	RAIN
1	HIGH	HIGH	YES	YES
2	HIGH	HIGH	NO	NO
3	LOW	HIGH	NO	NO
4	MEDIUM	LOW	NO	NO
5	HIGH	LOW	YES	YES
6	MEDIUM	LOW	YES	YES
7	LOW	LOW	YES	NO
8	LOW	LOW	NO	NO
9	LOW	HIGH	YES	YES
10	MEDIUM	LOW	NO	NO

Problem Statement:

TEMPERATURE	HIGH
HUMIDITY	LOW
CLOUDS	NO
RAIN	?

Naïve Bayes Classification Algorithm

- So to solve this problem, we need to follow the below steps:

1. **Calculate the frequency tables for all the features:**

FREQUENCY TABLE	RAIN	
TEMPERATURE	YES	NO
HIGH	2	1
MEDIUM	1	2
LOW	1	3

FREQUENCY TABLE	RAIN	
HUMIDITY	YES	NO
HIGH	2	2
LOW	2	4

FREQUENCY TABLE	RAIN	
CLOUDS	YES	NO
YES	4	1
NO	0	5

Naïve Bayes Classification Algorithm

2. **Generate Likelihood tables by finding the probabilities of given features.**

Using the concept of Conditional Probability :**P(A/B) =P(A and B)/P(B)**

P(Temp=High / Rain=Yes) = P(Temp=High and Rain=Yes)/P(Rain=Yes)

Now, P(Temp=High and Rain=Yes) = 2 / 10 and P(Rain=Yes)= 4 / 10

Hence, P(Temp=High/Rain=Yes)= (2/10) / (4/10) =**2/4**

PROBABILITY DISTRIBUTION	RAIN			
TEMPERATURE	YES	NO		
HIGH	0.5	0.1667	P(HIGH/YES) = 2/4	P(HIGH/NO) = 1/6
MEDIUM	0.25	0.333	P(MEDIUM/YES) = 1/4	P(MEDIUM/NO) = 2/6
LOW	0.25	0.5	P(LOW/YES) = 1/4	P(LOW/NO) = 3/6

PROBABILITY DISTRIBUTION	RAIN			
HUMIDITY	YES	NO		
HIGH	0.5	0.33	P(HIGH/YES) = 2/4	P(HIGH/YES) = 2/6
LOW	0.5	0.6667	P(LOW/YES) = 2/4	P(LOW/NO) = 4/6

PROBABILITY DISTRIBUTION	RAIN			
CLOUDS	YES	NO		
YES	1	0.1667	P(CLOUDS_YES/YES) = 4/4	P(CLOUDS_YES/NO) = 1/6
NO	0	0.8333	P(CLOUDS_NO/YES) = 0/4	P(CLOUDS_NO/NO) = 5/6

Naïve Bayes Classification Algorithm

3. **Now, use Bayes theorem to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.**

- Naïve Baye's Classifier assumes that all features are independent, so the previous equation can be rewritten as:

$$P(class / f1,f2,\ldots,fn) = \frac{P(class) * P(f1 / class) * \ldots.. * P(fn / class)}{P(f1)P(f2)\ldots.P(fn)}$$

- Hence probability of "Rain =Yes" with given conditions Temp="High", Humidity="Low" and Clouds="No" is:

P(Yes/(Temp=High, Humidity=Low, Clouds=No))

= (P(Temp=High/Yes) * P(Humidity=Low/Yes) * P(Clouds=No/Yes) * P(Yes)) / (P(Temp=High) * P(Humidity=Low) * P(Clouds=No))

= (0.5 * 0.5 * 0 * 0.4) = 0 [Excluding the denominator]

Naïve Bayes Classification Algorithm

- Again, probability of "Rain =No" with given conditions Temp="High", Humidity="Low" and Clouds="No" is:

  P(No/(Temp = High, Humidity=Low, Clouds=No))

  = (P(Temp=High/No) * P(Humidity=Low/No) * P(Clouds=No/No) * P(No)) / (P(Temp=High) * P(Humidity=Low) * P(Clouds=No))

  = (0.1667 * 0.6667 * 0.8333 * 0.6)

  = 0.05556 [Excluding the denominator]

- So here we get,

  Posterior Probability(Yes) < Posterior Probability(No)

  Hence, **Rain wont occur on that day**.

Implementation of Naïve Bayes Classifier using Python

- Step-1: Preparing the Dataset:

  Here we use the dummy dataset with four columns: Temperature, Humidity, Clouds and Rain. The first three columns are features and the last is the label.

```
temperature=['high','high','low','medium','high','medium','low','low','low','medium]
humidity=['high','high','high','low','low','low','low','low','high','low']
clouds=['yes','no','no','no','yes','yes','yes','no','yes','no']
rain=['yes','no','no','no','yes','yes','no','no','yes','yes']
```

- Step-2: Encoding Features:

  Here, we need to convert these string labels into numbers. for example: ‘high', ‘medium', ‘low' as 0, 1, 2. This is known as label encoding. Scikit-learn provides LabelEncoder library for encoding labels with a value between 0 and one less than the number of discrete classes.

Implementation of Naïve Bayes Classifier using Python

```
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers
temp_encoded=le.fit_transform(temperature)
humidity_encoded=le.fit_transform(humidity)
clouds_encoded=le.fit_transform(clouds)
```

Temperature: [0 0 1 2 0 2 1 1 1 2]
Humidity: [0 0 0 1 1 1 1 1 0 1]
Clouds: [1 0 0 0 1 1 1 0 1 0]

- Step-3: Combining all the Features in a single variable:

```
features=list(zip(temp_encoded,humidity_encoded,clouds_encoded))
```

[(0, 0, 1), (0, 0, 0), (1, 0, 0), (2, 1, 0), (0, 1, 1), (2, 1, 1), (1, 1, 1), (1, 1, 0), (1, 0, 1), (2, 1, 0)]

Implementation of Naïve Bayes Classifier using Python

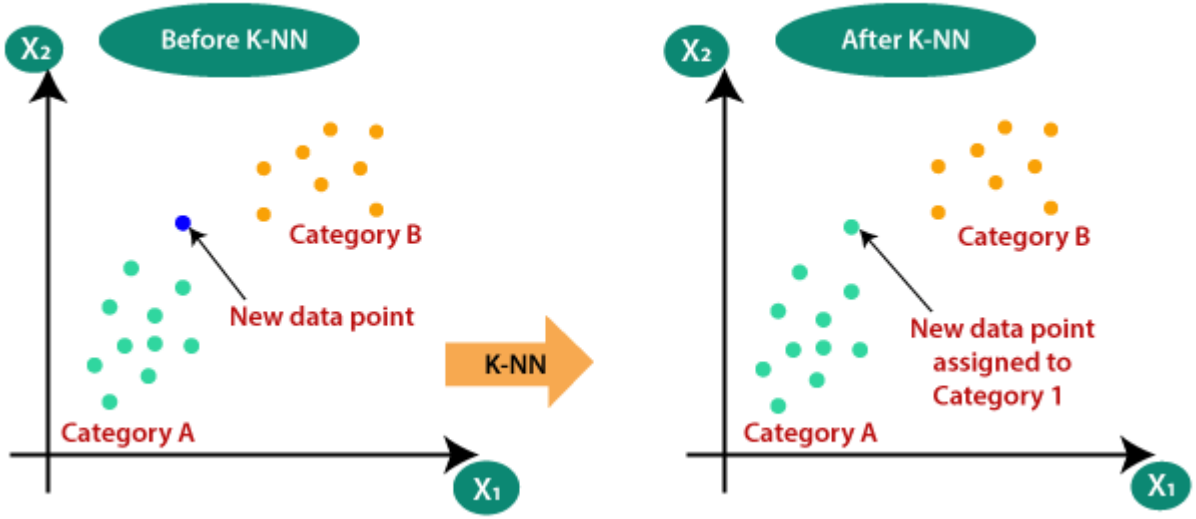
- Step-4: Generating Model:

```
#Create a Gaussian Classifier
model = GaussianNB()
# Train the model using the training sets
model.fit(features,rain)
#Predict Output
predicted= model.predict([[0,1,0]])
print ("Predicted Value:", predicted)
```

Predicted Value: **['no']**

K-Nearest Neighbor Algorithm

- This algorithm is used to solve the classification problems.
- K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.
- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point(in blue colour), so K-NN algorithm can be used to identify the category in which this data point will lie.



K-Nearest Neighbor Algorithm

- Select the number ‘k’ of the neighbors.(k=5)
- Calculate the Euclidian Distances of the new data point with all the existing data points
- Take the k nearest neighbors and count the number of data points in each category.
- Assign the new data points to that category for which the number of the neighbour is maximum.

- **Standardization: When independent variables in training data are measured in different units, it is important to standardize variables before calculating distance.**
- **Xstd = (X – mean) / standard deviation**
- **Xstd = (X-mean)/(max-min)**

K-Nearest Neighbor Example

- Suppose we have height, weight and T-shirt size of some customers.
- We have to predict the size of the T-shirt of a new customer whose height and weight is given.
- Let us consider the following available information:
- Find T-shirt size of the new customer whose
  - Height: 161 cm
  - Weight: 61 kg
  - T-Shirt Size: ?

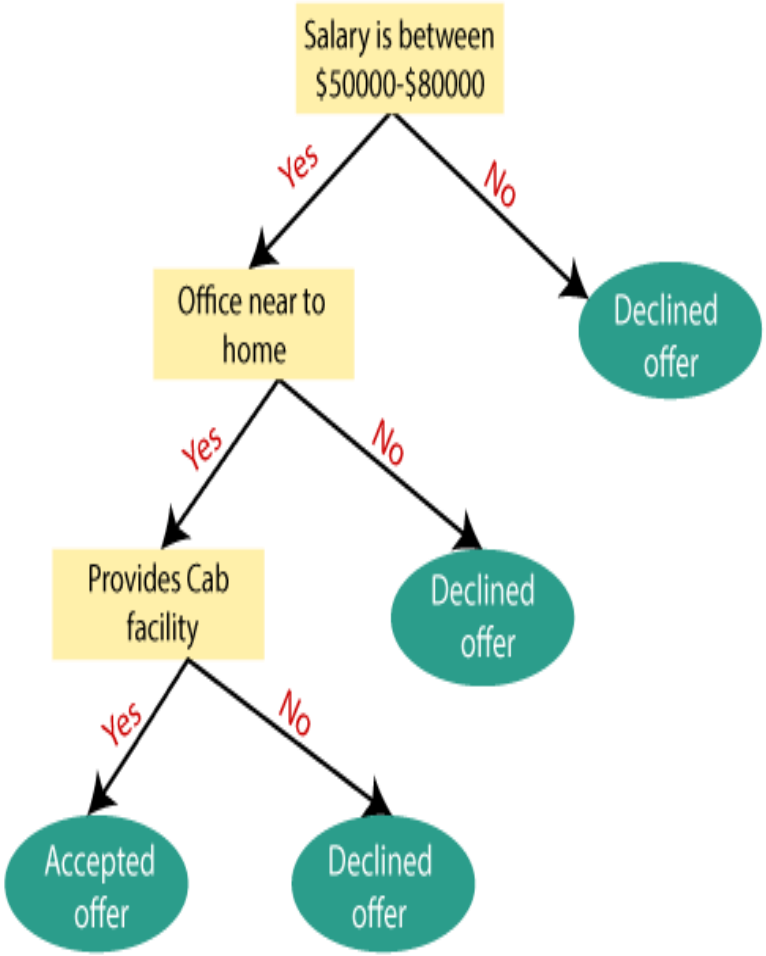
Height (cm)	Weight (kg)	T-shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Decision Tree

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems.
- **Here internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

Decision Tree Algorithm

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



Decision Tree

Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- There are two popular techniques for ASM, which are:
- **Information Gain**
- **Gini Index**
- **Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes})\log_2 P(\text{yes}) - P(\text{no})\log_2 P(\text{no})$$

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

1. Information Gain:

•Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

•It calculates how much information a feature provides us about a class.

•According to the value of information gain, we split the node and build the decision tree.

•A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1.Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

2. Gini Index:

•Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

•An attribute with the low Gini index should be preferred as compared to the high Gini index.

•It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

•Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Random Forest

- It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

How Random Forest Works

Step-1: Select random K data points from the training set.
Step-2: Build the decision trees associated with the selected data points (Subsets).
Step-3: Choose the number N for decision trees that you want to build.
Step-4: Repeat Step 1 & 2.
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.