DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Machine Learning
Course Code:22AIM52
Module 1

Academic Year:2024-2025
BATCH: 2022 – 2026
YEAR: III
SEMESTER: V

Subject-Teachers: Dr. Sowmya HK
Prof. Rajsree R S

MODULE 1

MODULE-1 Introduction to Machine Learning: Understanding Machine Learning: Definition and Types of Machine Learning-Application of Machine Learning-Machine Learning Algorithms: Supervised, Unsupervised, and Semi-Supervised Learning Algorithms. Machine Learning Models Model Evaluation Metrics: Confusion Matrix, Precision, Recall, F1 Score -ROC Curve and AUC-ROC. Advanced Techniques: Feature Scaling and Normalization -Encoding Categorical Variables-Train-test Split and Cross-validation

Introduction to Machine Learning

- In algorithm based solution data and program are fed into the computer, program gets executed and the desired output is produced.In ML based solution data and target labels(output) is fed into the system, system learns from this data and build the model(program).

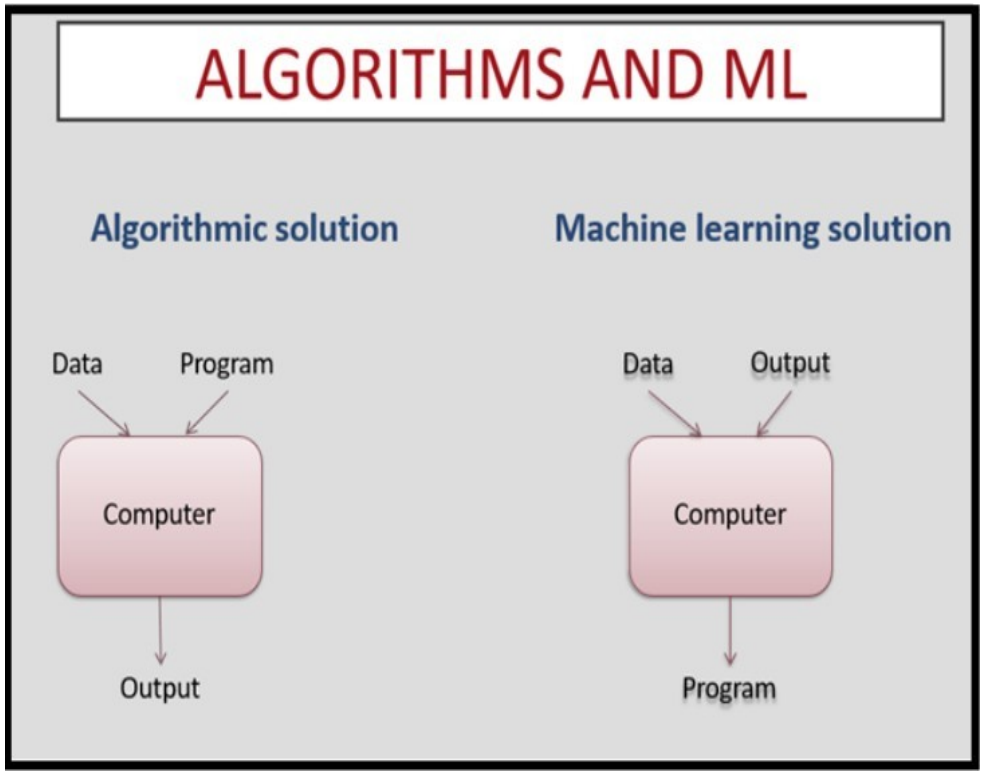


Figure 1: Diagram representing difference between Algorithmic and ML based solution

Definition of Machine Learning

- Machine Learning explores algorithms that can
  - learn from data / build a model from data
  - use the model for prediction, decision making or solving some tasks
- **Machine learning (ML) is** defined as a discipline of artificial intelligence (AI) that provides machines the ability to automatically learn from data and past experiences to identify patterns and make predictions with minimal human intervention.

Terminologies in Machine Learning

- **Data set** : Dataset is a collection of various types of data stored in a digital format.
- **Feature**s: A feature is one column of the data in your input set. feature is input; The key identified properties based on which we would like to predict the result of our problem statement.
- **Labels** are also known as tags, which are used to give an identification to a piece of data and tell some information about that element.-A label is the thing we're predicting -Eg: Price of stock
- Labels are also referred to as the final output for a prediction. For example, as in the below image, we have labels such as a cat and dog, etc.
- label is output. This applies to both classification and regression problems
- **Training data** is the initial dataset you use to teach a machine learning application to recognize patterns or perform to your criteria.
- **Testing or validation data** is used to evaluate your model's accuracy.
- An **Algorithm** is a set of rules that a machine follows to achieve a particular goal.
- An algorithm can be considered as a recipe that defines the inputs, the output and all the steps needed to get from the inputs to the output.
- **Model:** System that has been trained to recognize certain type of patterns.
- A model defines the relationship between features and label.

- Train the model over set of data by providing algorithm to learn from the data.
- **Regression vs. classification**
  A **regression** model predicts continuous values.
  -Eg:What is the value of a house in Bangalore?(price of a house)
  A **classification** model predicts discrete values.
  Eg: Is a given email message spam or not spam?

Applications of Machine learning

Image Recognition:

- It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, Automatic friend tagging suggestion:
- Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's face detection and recognition algorithm.
  It is based on the Facebook project named "Deep Face," which is responsible for face recognition and person identification in the picture

Speech Recognition

- While using Google, we get an option of "Search by voice," it comes under speech recognition, and it's a popular application of machine learning.
- Speech recognition is a process of converting voice instructions into text, and it is also known as "Speech to text", or "Computer speech recognition."
- At present, machine learning algorithms are widely used by various applications of speech recognition. Google assistant, Siri, Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

Traffic prediction:

- If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

- It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:
    - Real Time location of the vehicle form Google Map app and sensors
    - Average time has taken on past days at the same time.
- Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

Product recommendations:

- Machine learning is widely used by various e-commerce and entertainment companies such as Amazon, Netflix, etc., for product recommendation to the user.
- Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.
- Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.
- As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning

  Self-driving cars:
- One of the most exciting applications of machine learning is self-driving cars.
- Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car.
- It is using unsupervised learning method to train the car models to detect people and objects while driving.

Email Spam and Malware Filtering:

- Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning.
- Below are some spam filters used by Gmail:
    - Content Filter

- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

- Some machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier are used for email spam filtering and malware detection.

.

Virtual Personal Assistant:

- We have various virtual personal assistants such as Google assistant, Alexa, Cortana, Siri. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.
- These virtual assistants use machine learning algorithms as an important part.
- These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

- **Online Fraud Detection:**
- Machine learning is making our online transaction safe and secure by detecting fraud transaction.
- Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as fake accounts, fake ids, and steal money in the middle of a transaction.
- So to detect this, Feed Forward Neural network helps us by checking whether it is a genuine transaction or a fraud transaction.
- For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round.
- For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

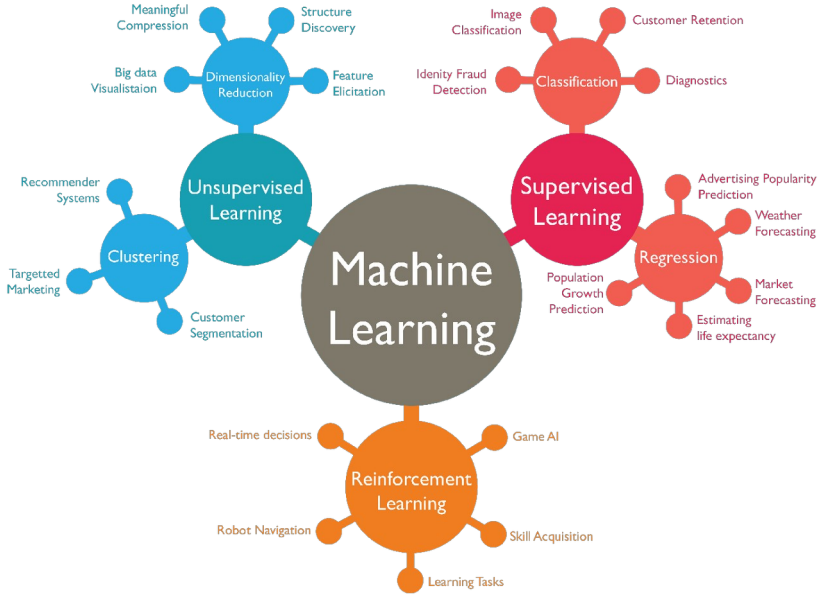
Stock Market trading:

- Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's long short term memory neural network is used for the prediction of stock market trends.
- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.
-
- **10. Medical Diagnosis:**
- In medical science, machine learning is used for *diseases diagnoses*. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain
- Automatic Language Translation:
- Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages.
- Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.
- The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

Automatic Language Translation:

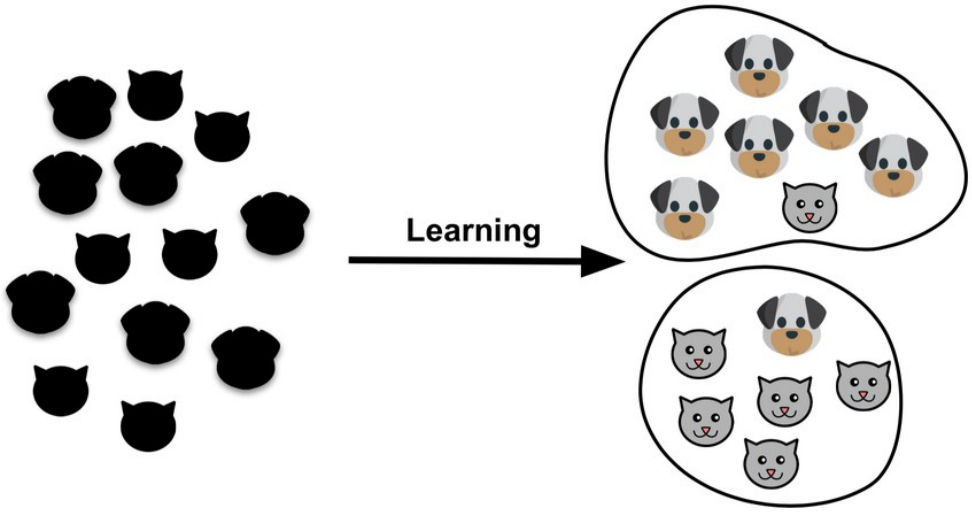
- Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages.
- **Google's GNMT (**Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.
- The technology behind the automatic translation is a **sequence to sequence learning algorithm**, which is used with image recognition and translates the text from one language to another language.
-

Types of Machine Learning

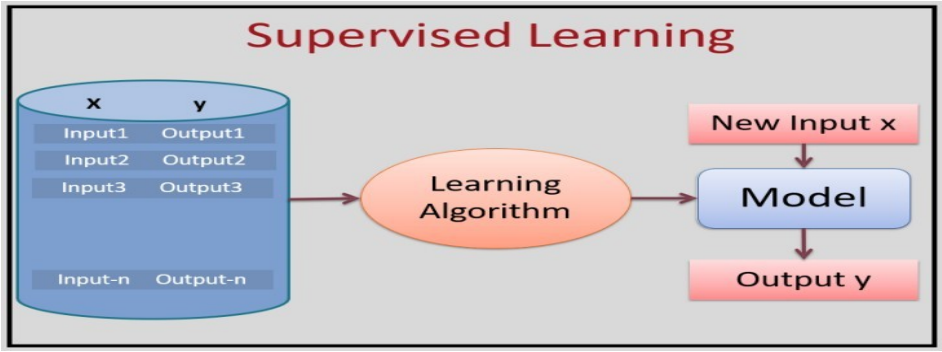


Supervised Machine Learning

- In the supervised learning technique, we train the machines using the labeled or trained dataset, and on the basis of this training, the machine predicts the output.
- In this context, the labeled data specifies that some of the inputs that we feed to the algorithm are already mapped to the output.



- Supervised learning algorithms are used when the **output is classified or labeled.**

- These algorithms learn from the past data that is inputted, called training data, runs its analysis and uses this analysis to predict future events of any new data within the known classifications.
- The accurate prediction of test data requires large data to have a sufficient understanding of the patterns.
- The algorithm can be trained further by comparing the training outputs to actual ones and using the errors for modification of the algorithms.
- 

Advantages of Supervised Learning

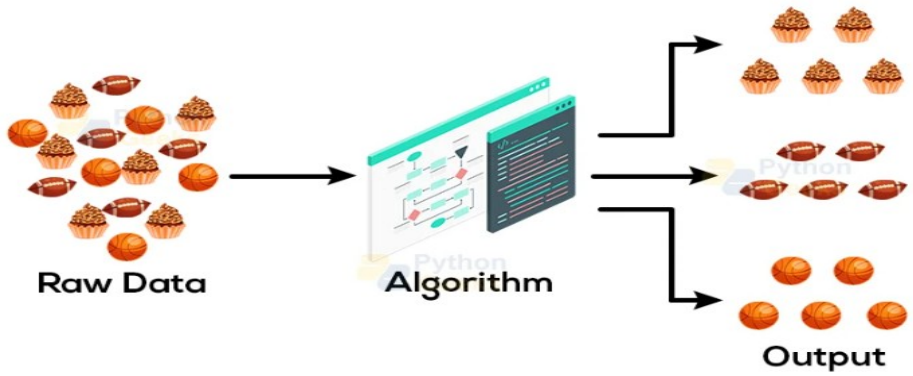
- Since supervised learning works with the labeled dataset, we can have an *exact idea about the classification* of objects and other variables that we feed in as input.
- These algorithms are helpful in predicting the output on the basis of *prior experience and trained data.*

Disadvantages of Supervised Learning

- These algorithms are not able to solve complex tasks due to a lack of data.
- It may predict the *wrong output if the test data is different from the training data* or the training data has some noise.
- It requires *lots of computational time to train* the algorithm and a huge load of trained data.

Unsupervised Machine Learning

- In unsupervised machine learning, we train the machine using the unlabeled or untrained dataset, and the machine predicts the output without any supervision of such data.

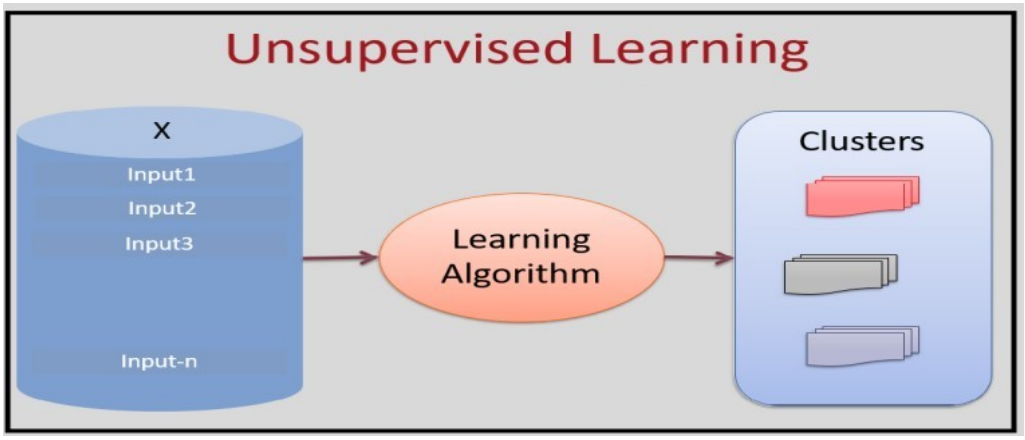


Advantages of Unsupervised Learning

- We can use these algorithms for complicated tasks as compared to the supervised ones because these algorithms work on unlabeled datasets and *do not require large datasets*.
- Unsupervised algorithms are *preferable for various tasks* as getting the unlabeled dataset is easier as compared to the labeled dataset for the training of these algorithms.

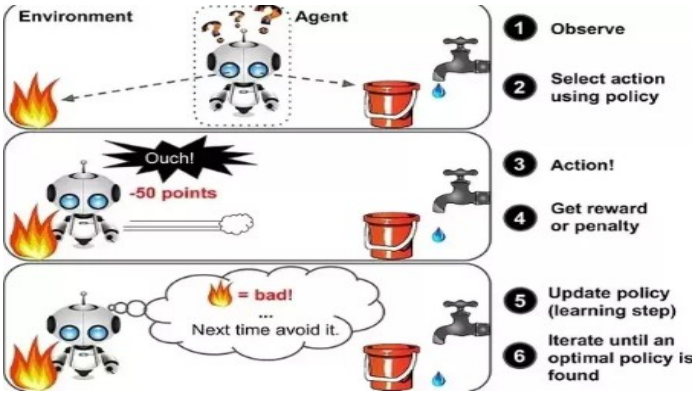
Disadvantages of Unsupervised Learning

- An unsupervised algorithm may result *in less accurate outputs* as the dataset is not labeled, and we have not trained the algorithms with the exact output beforehand.
- Working with Unsupervised learning is more *difficult as compared to other types* as it works with the unlabelled dataset that does not map with the output precisely.
- The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.



Reinforcement Learning

- Reinforcement learning operates on a **feedback-based process**, in which an Artificial Intelligence agent automatically explores its surroundings by *hit and trial methods*.
- It takes action, *learns from experiences*, and improves its performance. The algorithm **rewards** such agents for **each good action** and **punishes** it for each **bad action**.
- Hence, the reinforcement learning agent aims to **maximize** the rewards and **minimize** the punishments.
- In reinforcement learning, the algorithm does not require any labeled data like supervised learning, and agents solely learn from their experiences.



Advantages of Reinforcement Learning

- This type of learning assists us in solving *complex real-world problems* which are difficult to be solved by general techniques that we use conventionally.
- The learning model of Reinforcement Learning is similar to the learning process of human beings; therefore, we can look for the most accurate results.
- This type of learning helps us in achieving *long-term results*.

Disadvantages of Reinforcement Learning

- We generally do not prefer these algorithms for simple problems.
- These algorithms require *huge data and high computational powers*.

Too much reinforcement learning can lead to an *overload of states* which can weaken the results defying the purpose of deploying them.

Semi-Supervised Learning

- Semi-Supervised learning is a type of Machine Learning algorithm that represents the intermediate ground between Supervised and Unsupervised learning algorithms.
- It uses the combination of labeled and unlabeled datasets during the training period.
- The basic disadvantage of supervised learning is that it requires hand-labeling by ML specialists or data scientists, and it also requires a high cost to process.
- Unsupervised learning also has a limited spectrum for its applications.
- To overcome these drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced.

- In semi-supervised learning algorithm, training data is a combination of both labeled and unlabeled data.
- However, labeled data exists with a very small amount while it consists of a huge amount of unlabeled data.
- Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labeled data.

Advantages of semi-supervised learning

- It is quite simple and easy to understand the algorithm and does not encounter anomalies.
- This is *highly efficient in predicting the output* on the basis of input data.
- It overcomes the drawbacks of Supervised and Unsupervised Learning algorithms.

Disadvantages of semi-supervised learning

- Iteration results may not be stable, and outputs may vary significantly.
- We cannot apply these algorithms to network-level data due to its complexities.
- The *accuracy rate* for this type of Learning is *low.*

Criteria	Supervised ML	Unsupervised ML	Reinforcement ML
Definition	Learns by using labelled data	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of data	Labelled data	Unlabelled data	No – predefined data
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Supervision	Extra supervision	No supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

Machine Learning Models and Performance Measures

Machine Learning models are programs that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response.

First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

Confusion Matrix

- Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.
- A confusion matrix is defined as the table that is often used to describe the **performance of a classification model** on a *set of the test data* for which the *true values are known.*
- Let's take an example of a patient who has gone to a doctor with certain symptoms. Since it's the season of Covid, let's assume that he went with fever, cough, throat ache, and cold. These are symptoms that can occur during any seasonal changes too. Hence, it is tricky for the doctor to do the right diagnosis.
- ***True Positive (TP):***

- Let's say the patient was actually suffering from Covid and on doing the required assessment, the doctor classified him as a Covid patient. This is called TP or True Positive.
- This is because the case is positive in real and at the same time the case was **classified correctly**. Now, the patient can be given appropriate treatment which means, the decision made by the doctor will have a positive effect on the patient and society.
- ***False Positive (FP):***
- Let's say the patient was not suffering from Covid and he was only showing symptoms of seasonal flu but the doctor diagnosed him with Covid. This is called FP or False Positive.

  This is because the case was actually negative but was **falsely classified as positive**. Now, the patient will end up getting admitted to the hospital or home and will be given treatment for Covid.
- This is an unnecessary inconvenience for him and others as he will get unwanted treatment and quarantine. This is called **Type I Error**.
- ***True Negative (TN):***
- Let's say the patient was not suffering from Covid and the doctor also gave him a clean chit. This is called TN or True Negative. This is because the case was **actually negative and was also classified as negative** which is the right thing to do. Now the patient will get treatment for his actual illness instead of taking Covid treatment.
- ***False Negative (FN):***
- Let's say the patient was suffering from Covid and the doctor did not diagnose him with Covid. This is called FN or False Negative as the case was actually positive but was **falsely classified as negative**.
- Now the patient will not get the right treatment and also he will spread the disease to others. This is a highly dangerous situation in this example. This is also called **Type II Error.**
-

		PREDICTED	
		Positive	Negative
ACTUAL	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

•

Accuracy:

Accuracy = (TP + TN) / (TP + FP +TN + FN)

This term tells us how many right classifications were made out of all the classifications. In other words, how many TPs and TNs were done out of TP + TN + FP + FNs. It tells the ratio of "True"s to the sum of "True"s and "False"s.

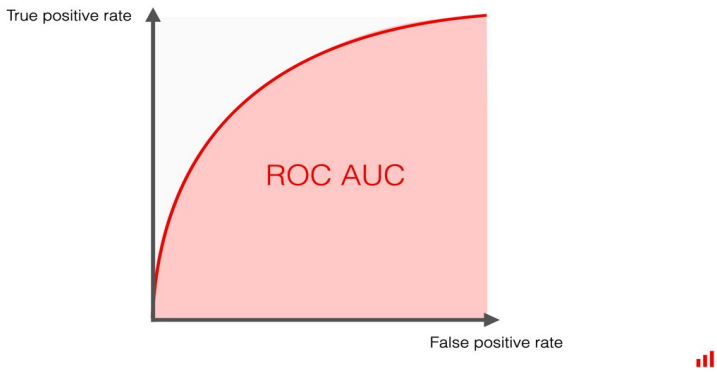
- **Precision:**
  - Precision = TP / (TP + FP)
  - Out of all that positive predicted, how many are actually truly positive.
  - The precision value lies between 0 and 1.
- **Recall or Sensitivity:**
  - Recall = TP/ (TP + FN)
  - Out of all total positive cases, what percentage is predicted as positive.
- **F1-Score:**
  - F1 score = 2* (Precision * Recall) / (Precision + Recall)
  - F1- score is a weighted average of precision and recall.
  - It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

A machine learning model is trained to predict tumor in patients. The test dataset consists of 100 people.

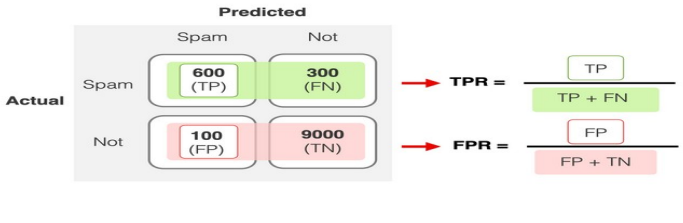
		ACTUAL	
		Negative	Positive
PREDICTION	Negative	60	8
	Positive	22	10

AUC - ROC curve

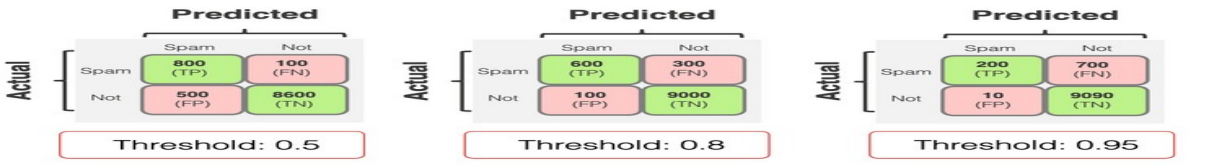
- The ROC(Receiver Operating Characteristic) curve: It is a graphical representation of the performance of a binary classifier at different classification thresholds.
- To compute it, you must measure the area under the ROC curve, which shows the classifier's performance at varying decision thresholds.
    - The ROC curve shows the performance of a binary classifier with different decision thresholds. It plots the True Positive rate (TPR) against the False Positive rate (FPR).
    - The ROC AUC score is the area under the ROC curve. It sums up how well a model can produce relative scores to discriminate between positive or negative instances across all classification thresholds.
    - The ROC AUC score ranges from 0 to 1, where 0.5 indicates random guessing, and 1 indicates perfect performance.



    - ROC AUC score shows how well the classifier distinguishes positive and negative classes. It can take values from 0 to 1.
    - A higher ROC AUC indicates better performance. A perfect model would have an AUC of 1, while a random model would have an AUC of 0.5.
- **True vs. False Positive rates:**
- The ROC curve plots the True Positive rate (TPR) against the False Positive rate (FPR) at various classification thresholds. You can derive TPR and FPR from a confusion matrix.



- **TPR** (True Positive rate, also known as recall) shows the share of detected true positives. For example, the share of emails correctly labeled as spam out of all spam emails in the dataset.
- **FPR** (False Positive rate) shows the share of objects falsely assigned a positive class out of all objects of the negative class. For example, the proportion of legitimate emails falsely labeled as spam
- When you **set the threshold higher**, you make the model "more conservative." It assigns the *True* label when it is "more confident." But as a consequence, you typically lower recall: you detect fewer examples of the target class overall.



- When you **set the threshold lower**, you make the model "less strict." It assigns the *True* label more often, even when "less confident." Consequently, you increase recall: you will detect more examples of the target class. However, this may also lead to lower precision, as the model may make more False Positive predictions.
- In the example above, the recall (TPR) decreases as we set the different decision higher:
- - 0.5 threshold: 800/(800+100)=0.89
- - 0.8 threshold: 600/(600+300)=0.67
- - 0.95 threshold: 200/(200+700)=0.22
- **The FPR also goes down:**
- - 0.5 threshold: 500/(500+8600)=0.06
- - 0.8 threshold: 100/(100+9000)=0.01

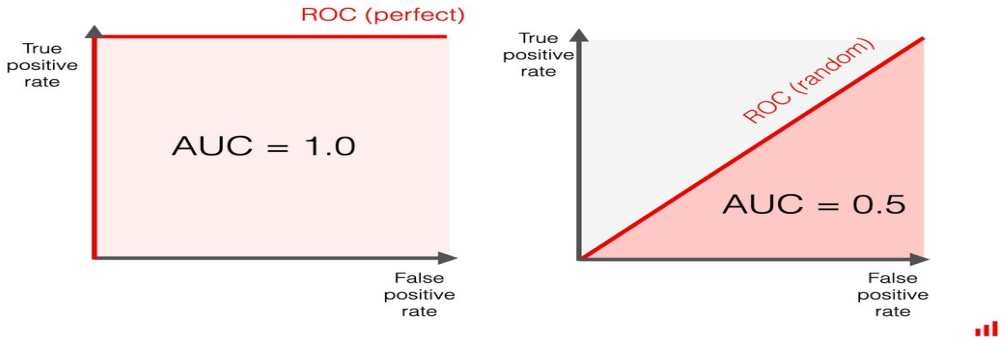
- - 0.95 threshold: 10/(10+9090)=0.001

ROC curve

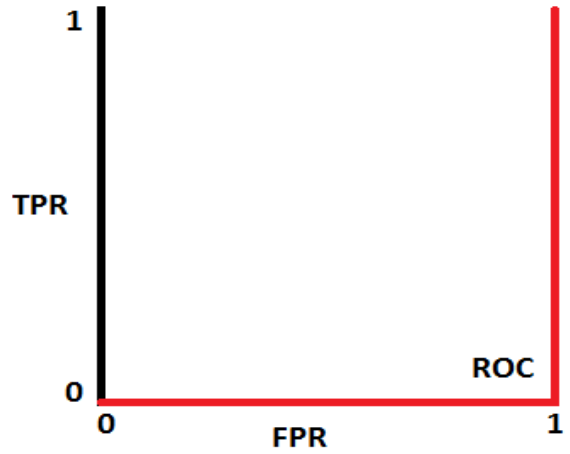
- An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds.
- This curve plots two parameters:
    - True Positive Rate
    - False Positive Rate
- **True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

- In the perfect scenario, we measure the square area: ROC AUC is 1. In the random scenario, it is precisely half: ROC AUC is 0.5.
- The ROC AUC score can range from 0 to 1. A score of 0.5 indicates random guessing, and a score of 1 indicates perfect performance.
- A score slightly above 0.5 shows that a model has at least "some" (albeit small) predictive power. This is generally inadequate for any real applications.



- When AUC is approximately 0, the model is actually reciprocating the classes. It means the model is predicting a negative class as a positive class and vice versa.



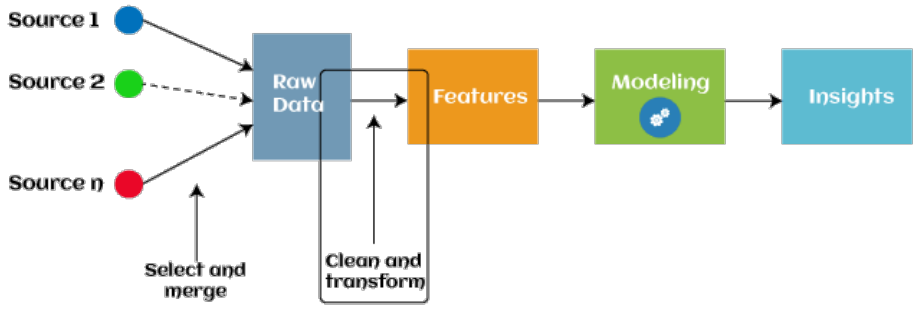
What is a feature?

- Generally, all machine learning algorithms take input data to generate the output.
- The input data remains in a tabular form consisting of rows (instances or observations) and columns (variable or attributes), and these attributes are often known as **features**.
- The input variables that we give to our machine learning models are called features. Each column in our dataset constitutes a feature.
- Features are nothing but the **independent variables** in machine learning models. What is required to be learned in any specific machine learning problem is a set of these features (independent variables), coefficients of these features, and parameters for coming up with appropriate functions or models (also termed hyperparameters).
    - Example: A model for predicting the risk of cardiac disease may have features such as the following:
    - Age
    - Gender
    - Weight
    - Whether the person smokes
    - Whether the person is suffering from diabetic disease, etc.

What is Feature Engineering?

- **Feature engineering is the pre-processing step of machine learning, which extracts features from raw data**.
- Feature engineering refers to a process of selecting and transforming variables when creating a predictive model using machine learning.
- It helps to *represent an underlying problem* to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data.
- The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.

- Eg: *size of the house, number of bedrooms, and location*, are the *predictor variables*. These are believed to determine the value of the property or house. *Actual prices* is the *outcome variable.*



Feature Creation:

- Feature creation is finding the most useful variables to be used in a predictive model.
- The process is subjective, and it requires human creativity and intervention.
- The new features are created by mixing existing features using *addition, subtraction, and ration,* and these new features have great flexibility.
- Types of Feature Creation:

Domain-Specific: Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.

Data-Driven: Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.

Synthetic: Generating new features by combining existing features or synthesizing new data points.

Transformations:

The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model.

For example, it ensures that the model is flexible to take input of the variety of data; it ensures that all the variables are on the same scale, making the model easier to understand.

It improves the model's accuracy and ensures that all the features are within the acceptable range to avoid any computational error.

Types of Feature Transformation:

Normalization: Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.

Scaling: Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.

Encoding: Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.

Transformation: Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

Feature Extraction

- **Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data.**
- **The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling.**
- **Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).**
- **Types of Feature Extraction:**

Dimensionality Reduction: Reducing the number of features by transforming the data into a lower-dimensional space while retaining important information. Examples are PCA and t-SNE.

Feature Combination: Combining two or more existing features to create a new one. For example, the interaction between two features.

Feature Aggregation: Aggregating features to create a new one. For example, calculating the mean, sum, or count of a set of features.

Feature Transformation: Transforming existing features into a new representation. For example, log transformation of a feature with a skewed distribution.

Feature Selection:

- While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant.
- If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model.

- Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.

- **Types of Feature Selection:**

1. **Filter Method:** Based on the statistical measure of the relationship between the feature and the target variable. Features with a **high correlation** are selected.

2. **Wrapper Method:** In wrapper methods, we try to use a **subset of features** and train a model using them. Based on the inferences that we draw from the previous model, we decide **to add or remove features** from your subset.

3. **Embedded Method:** Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.

- **Feature Scaling**

- Feature Scaling is the process of transforming the features so that they have a similar scale. This is important in machine learning because the scale of the features can affect the performance of the model.

- **Types of Feature Scaling:**

1. **Min-Max Scaling:** Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.

2. **Standard Scaling:** Rescaling the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation.

3. **Robust Scaling:** Rescaling the features to be robust to outliers by dividing them by the interquartile range.

Min-Max Scaling-Example

A value is normalized as follows:

$y = (x – min) / (max – min)$

Where the minimum and maximum values pertain to the value x being normalized.

For example, for a dataset, the min and max observable values as 30 and -10. We can then normalize any value, like 18.8, as follows:

$y = (x – min) / (max – min)$

$y = (18.8 – (-10)) / (30 – (-10))$

y = 28.8 / 40

y = 0.72

Standard Scaling -Example

A value is standardized as follows:

y = (x – mean) / standard_deviation

Where the *mean* is calculated as:

mean = sum(x) / count(x)

And the *standard_deviation* is calculated as:

standard_deviation = sqrt(sum((x – mean)^2) / count(x))

We can estimate a mean of 10.0 and a standard deviation of about 5.0. Using these values, we can standardize the first value of 20.7 as follows:

y = (x – mean) / standard_deviation

y = (20.7 – 10) / 5

y = (10.7) / 5

y = 2.14

Robust Scaler

- Robust Scaler algorithms scale features that are robust to outliers.
- It, thus, follows the following formula: $\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$
- Where Q1 is the 1st quartile, and Q3 is the third quartile.

Code

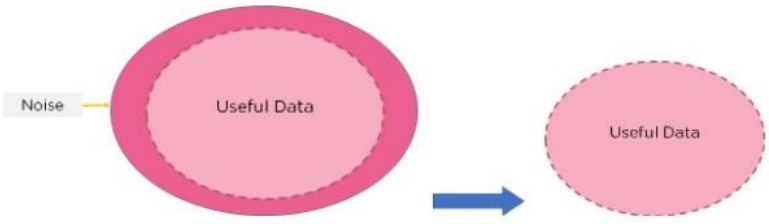
```
import pandas as pd
df = pd.DataFrame({
   'Income': [15000, 1800, 120000, 10000],
   'Age': [25, 18, 42, 51],
   'Department': ['HR','Legal','Marketing','Management']
})
df_scaled = df.copy()
col_names = ['Income', 'Age']
features = df_scaled[col_names]
        from sklearn.preprocessing import QuantileTransformer
        scaler = QuantileTransformer()
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

	Income	Age	Department
0	0.666667	0.333333	HR
1	0.000000	0.000000	Legal
2	1.000000	0.666667	Marketing
3	0.333333	1.000000	Management

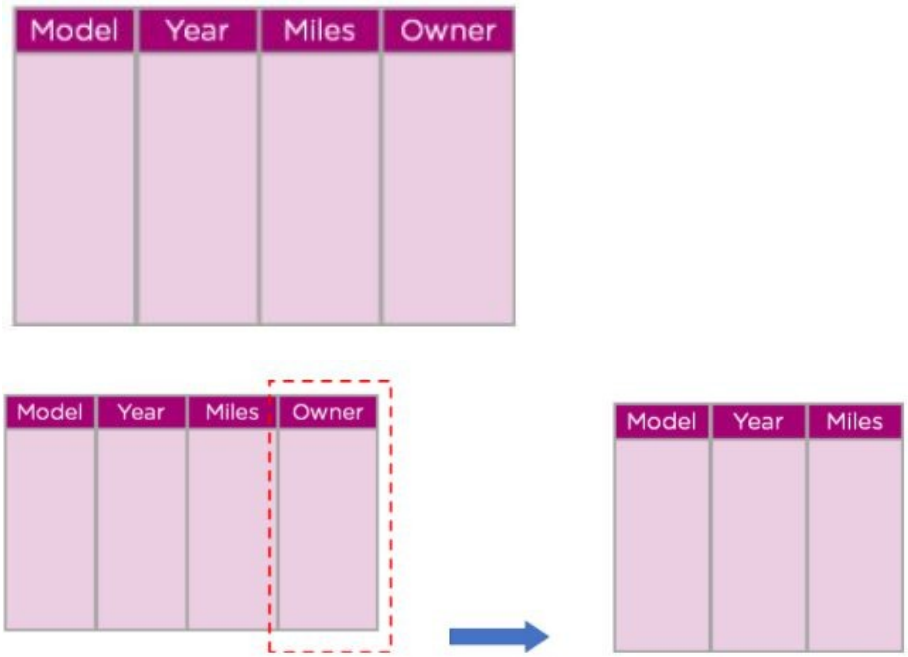
Feature Selection

- To train an optimal model, we need to make sure that we use only the essential features.
- If we have too many features, the model can capture the unimportant patterns and learn from noise. The method of choosing the important parameters of our data is called Feature Selection.
- To train a model, we collect enormous quantities of data to help the machine learn better.
- If a good portion of the data collected is noise, then some of the columns of our dataset might not contribute significantly to the performance of our model.
- Further, having a lot of data can slow down the training process and cause the model to be slower. The model may also learn from this irrelevant data and be inaccurate.
- It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve.
- We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data



Example:

Consider a table which contains information on old cars. The model decides which cars must be crushed for spare parts. The model of the car, the year of manufacture, and the miles it has traveled are pretty important to find out if the car is old enough to be crushed or not.However, the name of the previous owner of the car does not decide if the car should be crushed or not. Hence we can drop the column



Need for Feature Engineering in Machine Learning

- In machine learning, the performance of the model depends **on data pre-processing and data handling.**
- But if we create a model without pre-processing or data handling, then it may ***not give good accuracy***.
- Whereas, if we apply feature engineering on the same model, then the accuracy of the model is enhanced.
- Hence, feature engineering in machine learning improves the model's performance. Below are some points that explain the need for feature engineering:
    - Better features mean **flexibility.**
    - Better features mean **simpler models.**

Better features mean **better results.**

Steps in Feature Engineering

Data Preparation:

- In this step, raw data acquired from different resources are prepared to make it in a suitable format so that it can be used in the ML model.
- The data preparation may contain **cleaning of data, delivery, data augmentation, fusion, ingestion, or loading.**

Exploratory Analysis:

- Exploratory analysis or Exploratory data analysis (EDA) is an important step of features engineering, which is mainly used by data scientists.
- This step involves analysis, investing data set, and **summarization of the main characteristics** of data.
- Different data visualization techniques are used to better understand the manipulation of data sources
- Understand data set variables and relationship between them to find the most **appropriate statistical technique for data analysis**-mean, std. deviation and Regression**to select the best features** for the data.
- **Benchmark**:
- Benchmarking is a process of setting a standard baseline for accuracy to compare all the variables from this baseline.
- The benchmarking process is used to improve the predictability of the model and reduce the error rate.

Feature Engineering Techniques

1 **Imputation**

Feature engineering deals with inappropriate data, missing values, human interruption, general errors, insufficient data sources, etc. Missing values within the dataset highly affect the performance of the algorithm, and to deal with them "Imputation" technique is used.

Imputation is responsible for handling irregularities within the dataset.

For example, removing the missing values from the complete row or complete column by a huge percentage of missing values. But at the same time, to maintain the data size, it is required to impute the missing data, which can be done as:

For **numerical data imputation**, a default value can be imputed in a column, and **missing values** can be filled with **means or medians** of the columns.

For **categorical data** imputation, missing values can be interchanged with the **maximum occurred value** in a column.

	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.5
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.6
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.0
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.6
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.0
5	NaN	22-10-2020 17:24:00	Thursday	3.5	2.0	NaN

	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.50
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.60
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.00
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.60
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.00
5	Sour Jelly	22-10-2020 17:24:00	Thursday	3.5	2.0	7.94

Handling Outliers

- Outliers are the deviated values or *data points* that are observed *too away from other data points* in such a way that they ***badly affect the performance of the model***.
- Outliers can be handled with this feature engineering technique. This technique first identifies the outliers and then remove them out.
- **Standard deviation** can be used to **identify** the outliers. For example, each value within a space has a definite to an average distance, but if a value is greater distant than a certain value, it can be considered as an outlier.
- **Z-score** can also be used to detect outliers.

Log transform

- Logarithm transformation or log transform is one of the commonly used mathematical techniques in machine learning.
- Log transform helps in handling the skewed data, and it makes the distribution more approximate to normal after transformation.
- It also reduces the effects of outliers on the data, as because of the normalization of magnitude differences, a model becomes much robust.

Note: Log transformation is only applicable for the positive values; else, it will give an error. To avoid this, we can add 1 to the data before transformation, which ensures transformation to be positive.

Binning

- In machine learning, overfitting is one of the main issues that degrade the performance of the model and which occurs due to a ***greater number of parameters and noisy data***. (model works fine with training data but poor performance on testing data)
- However, one of the popular techniques of feature engineering, **"binning",** can be used to *normalize the noisy data*. This process involves segmenting different features into bins.
- Binning is the process of transforming numerical variables into categorical counterparts.
- Binning improves accuracy of the predictive models by reducing the noise or non-linearity in the dataset. Finally, binning lets easy identification of outliers, invalid and missing values of numerical variables.

Example: #Numerical Binning Example
Value Bin
0-30 -> Low
31-70 -> Mid
71-100 -> High
#Categorical Binning Example
Value Bin
Spain -> Europe
Italy -> Europe
Chile -> South America
Brazil -> South America

Feature Split

- As the name suggests, feature split is the process of splitting features intimately into two or more parts and performing to make new features.
- **This technique helps the algorithms to better understand and learn the patterns in the dataset.**
- The feature splitting process enables the new features to be clustered and binned, which results in extracting useful information and improving the performance of the data models.

```
data.name
0  Luther N. Gonzalez
1    Charles M. Young
2       Terry Lawson
3      Kristen White
4     Thomas Logsdon
#Extracting first names
data.name.str.split(" ").map(lambda x: x[0])
0     Luther
1    Charles
2      Terry
3    Kristen
4     Thomas
```

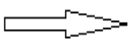
```
#Extracting last names
data.name.str.split(" ").map(lambda x: x[-1])
0    Gonzalez
1       Young
2      Lawson
3       White
4     Logsdon
```

Encoding-Categorical variables One hot encoding

- One hot encoding is the popular encoding technique in machine learning.
- It is a technique that ***converts the categorical data*** in a form so that they can be easily understood by machine learning algorithms and hence can make a good prediction.
- It enables group the of categorical data without losing any information.
- Here, categorical values are converted into simple numerical 1's and 0's without the loss of information.
- As with other techniques, OHE has its own disadvantages and has to be used sparingly. It could result in a dramatic increase in the number of features and result in the creation of highly correlated features.

	Candy Variety	Day	Type of Day	Weekend	Weekday
0	Chocolate Hearts	Sunday	Weekend	1	0
1	Sour Jelly	Saturday	Weekend	1	0
2	Candy Canes	Friday	Weekday	0	1
3	Sour Jelly	Sunday	Weekend	1	0
4	Fruit Drops	Sunday	Weekend	1	0
5	Sour Jelly	Thursday	Weekday	0	1

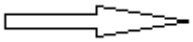
ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854



ID	Country_Japan	Country_U.S	Country_India	Country_China	Population
1	1	0	0	0	127185332
2	0	1	0	0	326766748
3	0	0	1	0	1354051854
4	0	0	0	1	1415045928
5	0	1	0	0	326766748
6	0	0	1	0	1354051854

One-hot encoding can lead to the multiple number of columns .When the column contains the less number of variation in the categorical variables then we can label encode them such that variants of same kind gets encoded to the same number.

ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854



ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

Curse of Dimensionality

- Curse of Dimensionality refers to a set of problems that arise when working with **high-dimensional data**.
- The dimension of a dataset corresponds to the **number of attributes/features** that exist in a dataset. A dataset with a large number of attributes, generally of the order of a hundred or more, is referred to as high dimensional data.
- Some of the difficulties that come with high dimensional data manifest during analyzing or visualizing the data to identify patterns, and some manifest while training machine learning models.
- The difficulties related to training machine learning models due to high dimensional data are referred to as the '***Curse of Dimensionality'.***
- In Machine Learning, a marginal increase in dimensionality also requires a large increase in the volume in the data in order to maintain the same level of performance. The curse of dimensionality is the by-product of a phenomenon which appears with high-dimensional data.

Cross Validation

Cross-validation is a technique for evaluating a machine learning model and testing its performance.

CV is commonly used in applied ML tasks. It helps to compare and select an appropriate model for the specific predictive modeling problem.

CV is easy to understand, and easy to implement. All of this makes cross-alidation a powerful tool for selecting the best model for a specific task.

Cross-validation Algorithm

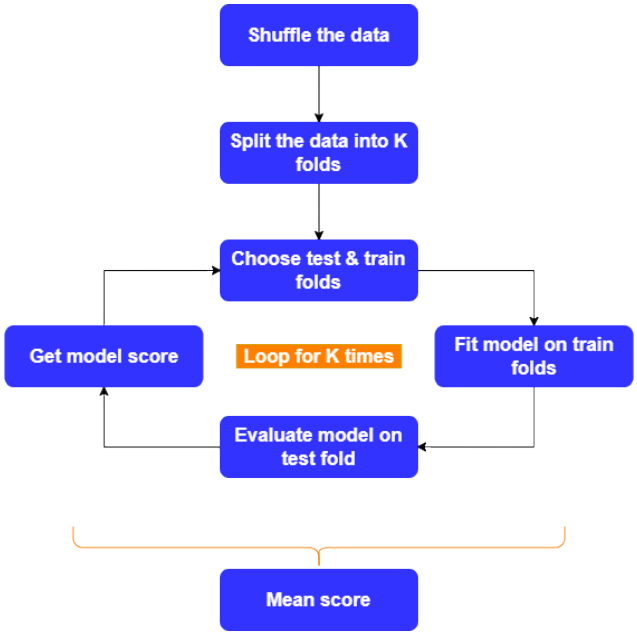
Divide the dataset into two parts: one for training, the other for testing Train the model on the training set Validate the model on the test set.

Repeat 1-3 steps a couple of times. This number depends on the CV method that you are using

- Hold-out
- **K-folds**
- Leave-one-out
- Leave-p-out
- Stratified K-folds
- Repeated K-folds
- Nested K-folds
- Time series CV

Inner Working Process of Cross-Validation

- Shuffle the dataset in order to remove any kind of order
- Split the data into K number of folds. K= 5 or 10 will work for most of the cases.
- Now keep one fold for testing and remaining all the folds for training.
- Train(fit) the model on the train set and test(evaluate) it on the test set and note down the results for that split
- Now repeat this process for all the folds, every time choosing a separate fold as
- test data
- So for every iteration our model gets trained and tested on different sets of data
- At the end sum up the scores from each split and get the mean score



K-fold cross validation

- In K Fold cross-validation input data is divided into 'K' number of folds, hence the name K Fold.
- Suppose we have divided data into 5 folds i.e. K=5.
- Now we have 5 sets of data to train and test our model.
- So the model will get trained and tested 5 times, but for every iteration we will use one fold as test data and rest all as training data.
- Note that for every iteration, data in training and test fold changes which adds to the effectiveness of this method.

Stratified K Fold Cross Validation

- Stratified K Fold used when just **random shuffling and splitting the data is not sufficient.**
- In case of regression problem folds are selected so that the mean response value is approximately equal in all the folds.
- In the case of classification problems folds are selected to have the same proportion of class labels.
- Stratified K Fold is more useful in the case of classification problems, where it is very important to have the **same percentage of labels in every fold.**

Advantages

- We end up using all the data for training and testing and this is very useful in case of small datasets.
    - It covers the variation of input data by validating the performance of the model on multiple folds.
    - Multiple folds also helps in case of unbalanced data.
    - Model performance analysis for every fold gives us more insights to fine tune the model.
    - Used for hyperparameter tuning.

Train Test Split

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. The procedure involves taking a dataset and dividing it into two subsets.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

The procedure has one main configuration parameter, which is the size of the train and test sets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

common split percentages include:

Train: 80%, Test: 20%

Train: 67%, Test: 33%

Train: 50%, Test: 50%

Code

 # split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

- **Repeatable Train-Test Splits**

- Rows are assigned to the train and test sets randomly.
- This is done to ensure that datasets are a representative sample (e.g. random sample) of the original dataset, which in turn, should be a representative sample of observations from the problem domain.
- This can be achieved by setting the "*random_state*" to an integer value

Code
 # split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)

Sample Questions

1. *How machine learning is different from general programming?*
2. *What is machine learning?*
3. *What is feature engineering? How does it affect the model's performance?*
4. *Why do we perform normalization?*
5. *What is Semi-supervised Machine Learning?*
6. *What is a confusion matrix, and how is it used?*
7. *Describe the difference between classification and regression*
8. *What is feature scaling, and why is it necessary?*
9. *What is feature selection, and what are some common techniques for it?*
10. *What is feature extraction, and how does it differ from feature selection?*
11. *What is cross-validation, and why is it used in machine learning?*
12. *Describe the concept of k-fold cross-validation and its advantages*
13. *Explain what is meant by "train-test split" and how it relates to cross-validation.*
14. *How does cross-validation help in assessing the generalization of a model?*

****
*

Module 2

- **Supervised Learning Regression and Classification Algorithms**
- **Regression:** Introduction to Regression- Regression Models- Linear Regression, Polynomial Regression
- **Decision Trees:** Introduction to Decision Trees-Tree Construction, Splitting Criteria, and Pruning-Handling Missing Values and Categorical Features- Gini Index-ID3-CART.

Regression

- Linear regression, as the name implies, is commonly used to estimate the ***linear relationship*** between independent variables - (x_1, x_2, …, x_n) and dependent variables- (y).
- You would use linear regression when your dependent variable is a continuous variable (value ranging between [-∞,+∞]).
- For example, predicting prices of houses, cars and stocks.

Univariate linear regression

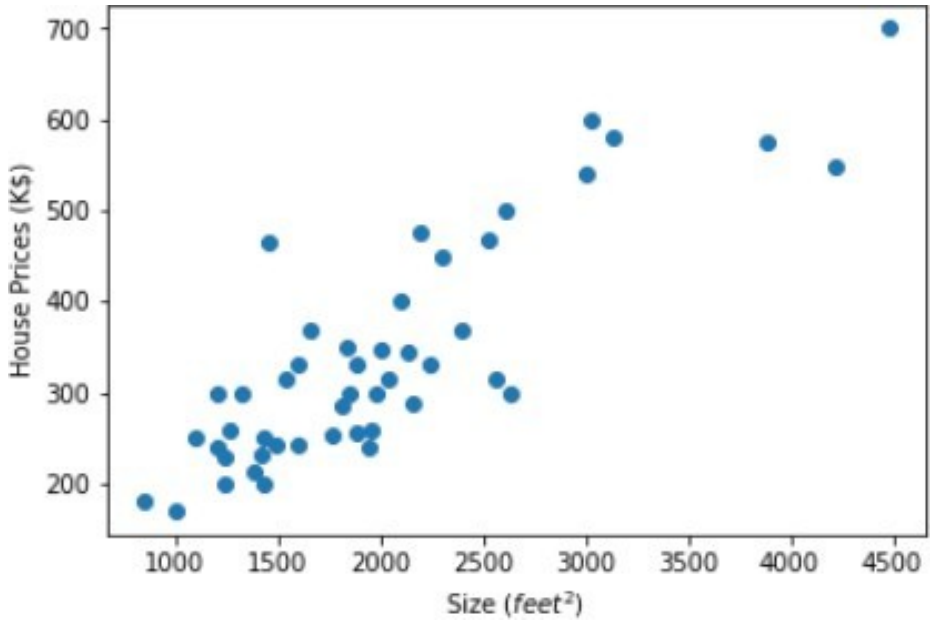
- Univariate linear regression focuses on determining relationship between ***one independent (explanatory variable) variable and one dependent variable.***
- For instance, estimating a person's weight (y) given his/her height (x).
- Independent variable: Input feature. (x_1, x_2, …, x_n)
- Dependent variable: The variable you are trying to predict. Sometimes also known as target variable. (y)

Example

- Here, we are interested in finding the relationship between a person's height and his/her weight.
- Intuitively, we would know that these two variables are positively correlated.
- In other words, taller people tends to be heavier than those that are shorter.
- For this example, height is our independent variable and weight is our dependent variable. To calculate the relationship between the two variables, we need some data.

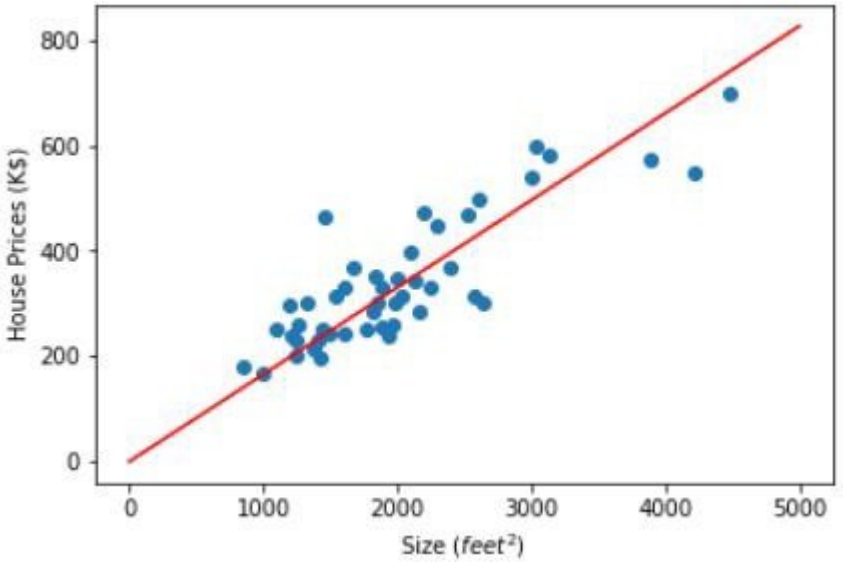
- If we have one input variable, we need to draw a line in the XY plane; if we have two input variables, we need to construct a plane in the three-dimensional coordinate system. More than two input variables are extremely tough to imagine.

Example

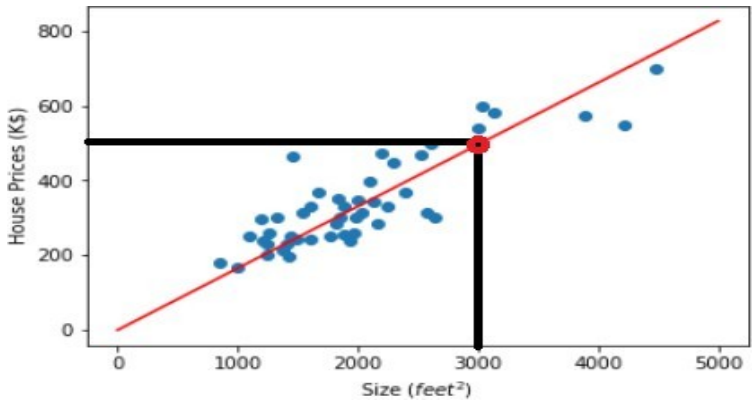


- In the graph above, the y-axis is the price (K$) of some houses, and the x-axis is the house's size (feet²).

- We want to draw a straight line on this graph (such as the image below) so that when we know the size of a new home not in this dataset, we can predict its price by using that line.

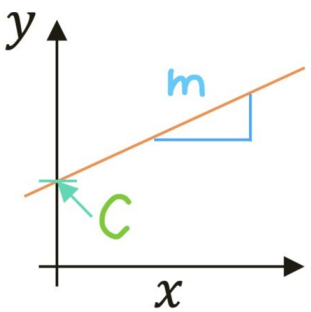


- For example, we can predict that the price of a 3000 feet² house is approximately 500 K$, as seen in the graph below.



- It is also obvious that we can fit a straight linear line to describe the relationship between these two variables.

$$y = mx + c$$



where m is the slope and c is the y- intercept of the line.

It is common to see these two parameters to be denoted as $c=\theta_0$ and $m=\theta_1$.

- In *linear regression*, we use a training set to come up with an algorithm that creates a function "*h*" that maps *x* to *y*.
- The function that we are trying to develop looks like this:
- For a univariate linear regression task, the hypothesis is of the form:

$$\hat{y} = h(x) = \theta_0 + \theta_1 x$$

where:

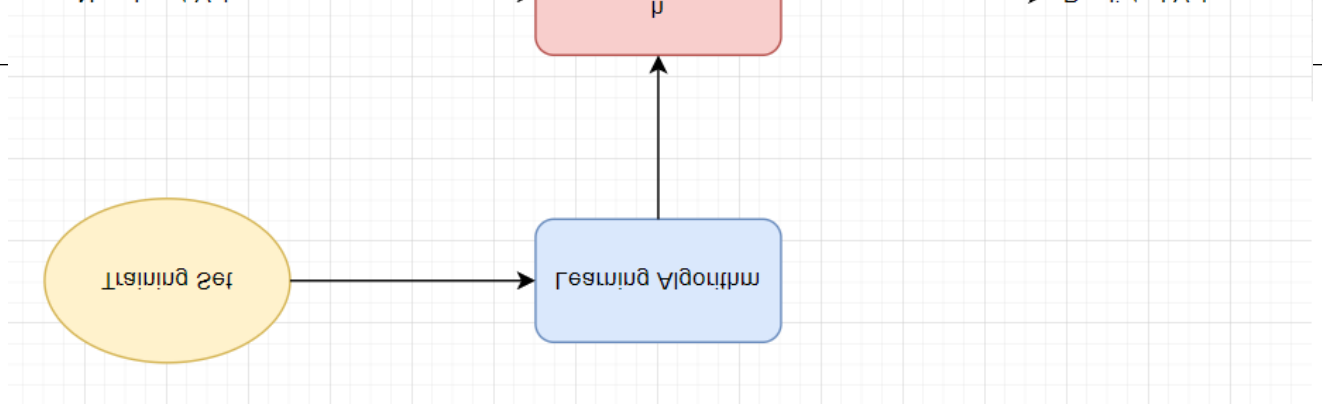
$\hat{y} = h(x)$: predicted target

x: input feature

θ_0: intercept

θ_1: slope

- A hypothesis is a potential model for a machine learning system



Regression Model Representation

- The algorithm finds the values for θ_0 and θ_1 that best fit the inputs and outputs given to the algorithm. This is called *univariate* linear regression because the θ parameters only go up to 1.
- The function that *multivariate* linear regression produces looks like this:

$$h_\theta(x_1, x_2, x_3, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

Least square method

- The **least square method** is the process of finding the best- fitting curve or line of best fit for a set of data points by *reducing the sum of the squares of the offsets (residual part) of the points from the curve.*
- During the process of finding the relation between two variables, the trend of outcomes are estimated quantitatively. This process is termed as **regression analysis**.

- The least square method is the process of finding the *best- fitting curve or line of best fit for a set of data points* by reducing the sum of the squares of the offsets (residual part) of the points from the curve.
- It is quite obvious that the fitting of curves for a particular data set are not always unique.
- Thus, it is required to *find a curve having a minimal deviation from all the measured data points*. This is known as the best-fitting curve and is found by using the least-squares method.

Least Square Method Definition

- The least-squares method is a crucial ***statistical method*** that is practiced to find a regression line or a best-fit line for the given pattern.
- This method is described by an equation with specific parameters. The method of least squares is generously used in evaluation and regression.
- The method of least squares actually defines the solution for the ***minimization of the sum of squares of deviations or the errors in the result of each equation.***
- Find the formula for sum of squares of errors, which help to find the variation in observed data.

Least Square Method Formula

- The least-square method states that the curve that best fits a given set of observations, is said to be a curve having a *minimum sum of the squared residuals* (or deviations or errors) from the given data points.
- Let us assume that the given points of data are (x_1, y_1), (x_2, y_2), (x_3, y_3), …, (x_n, y_n) in which all x's are independent variables, while all y's are dependent ones. Also, suppose that f(x) is the fitting curve and d represents error or deviation from each given point.
- Now, we can write:
- $d_1 = y_1 - f(x_1)$
- $d_2 = y_2 - f(x_2)$
- $d_3 = y_3 - f(x_3)$
- …..
- $d_n = y_n - f(x_n)$

- The least-squares explain that the curve that best fits is represented by the property that the ***sum of squares of all the deviations*** from given values must be minimum, i.e:

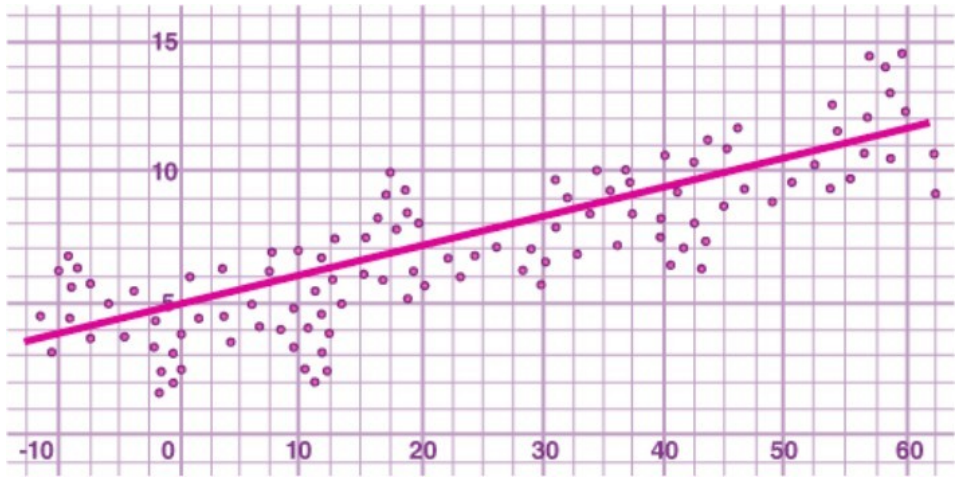
$$S = \sum_{i=1}^{n} {d_i}^2$$

$$S = \sum_{i=1}^{n} [y_i - f_{x_i}]^2$$

$$S = {d_1}^2 + {d_2}^2 + {d_3}^2 + \cdots + {d_n}^2$$

Least Square Method Graph

- In linear regression, the line of best fit is a straight line as shown in the following diagram:

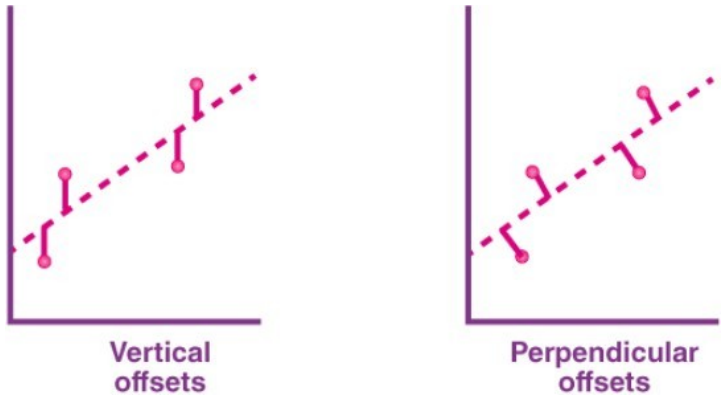


- The given data points are to be minimized by the method of reducing residuals or offsets of each point from the line.

- The given data points are to be minimized by the method of

  reducing residuals or offsets of each point from the line.

- The vertical offsets are generally used in surface, polynomial and hyperplane problems, while perpendicular offsets are utilized in common practice.



Sum = Minimum Quantity

- Suppose when we have to determine the equation of line of best fit for the given data, then we first use the following formula.
- The equation of least square line is given by $Y = a + bX$
- Normal equation for 'a':
- $\sum Y = na + b\sum X$
- Normal equation for 'b':
- $\sum XY = a\sum X + b\sum X^2$
- Solving these two normal equations we can get the required trend line equation.
- Thus, we can get the line of best fit with formula $y = a + bX$

Example

- The Least Squares Model for a set of data (x_1, y_1), (x_2, y_2), (x_3, y_3), …, (x_n, y_n) passes through the point (x_a, y_a), where x_a is the average of the x_i's and y_a is the average of the y_i's.
- The below example explains how to find the equation of a straight line or a least square line using the least square method.
- Consider the following data points

x_i	8	3	2	10	11	3	6	5	6	8
y_i	4	12	1	12	9	4	9	6	1	14

- Mean of x_i values = (8 + 3 + 2 + 10 + 11 + 3 + 6 + 5 + 6 + 8)/10 =

  62/10 = 6.2

- Mean of y_i values = (4 + 12 + 1 + 12 + 9 + 4 + 9 + 6 + 1 + 14)/10

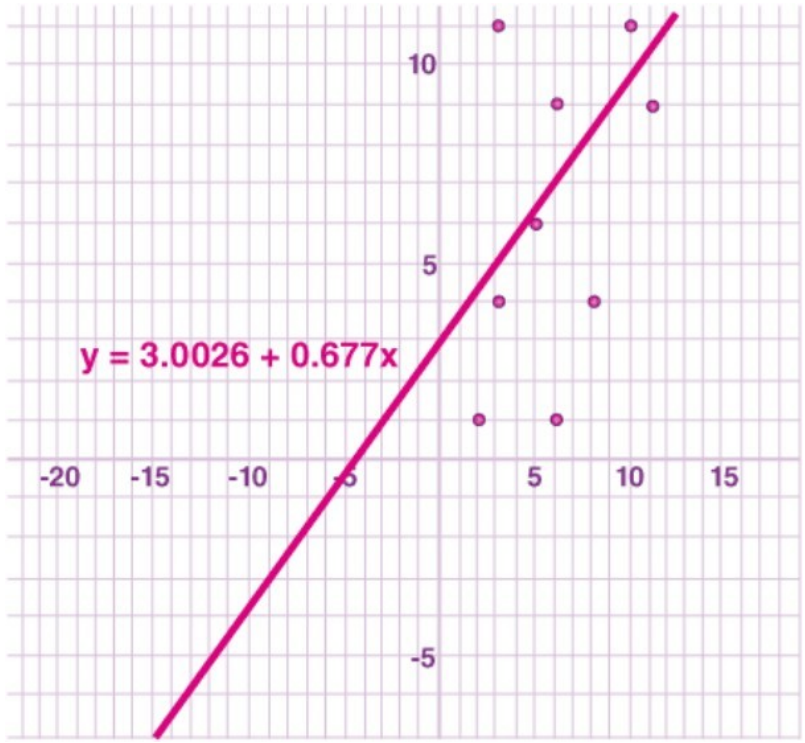
  = 72/10 = 7.2

- Straight line equation is $y = a + bx$.
- The normal equations are
- $\sum y = an + b\sum x$
- $\sum xy = a\sum x + b\sum x^2$

x	y	x^2	xy
8	4	64	32
3	12	9	36
2	1	4	2
10	12	100	120
11	9	121	99
3	4	9	12
6	9	36	54
5	6	25	30
6	1	36	6
8	14	64	112
$\sum x = 62$	$\sum y = 72$	$\sum x^2 = 468$	$\sum xy = 503$

- Substituting these values in the normal equations,
- $10a + 62b = 72$….(1)
- $62a + 468b = 503$….(2)
- (1) × 62 — (2) × 10,
- $620a + 3844b – (620a + 4680b) = 4464 – 5030$
- $-836b = -566$
- $b = 566/836$
- $b = 283/418$
- $b = 0.677$

- Substituting b = 0.677 in equation (1),
- $10a + 62(0.677) = 72$
- $10a + 41.974 = 72$
- $10a = 72 – 41.974$
- $10a = 30.026$
- $a = 30.026/10$
- $a = 3.0026$
- Therefore, the equation becomes,
- $y = a + bx$
- $y = 3.0026 + 0.677x$
-


y = 3.0026 + 0.677x
10
5
-5
-20
-15
-10
5
10
15

- Now, we can find the sum of squares of deviations from the obtained values as:
- $d_1 = [4 – (3.0026 + 0.677*8)] = (-4.4186)$
- $d_2 = [12 – (3.0026 + 0.677*3)] = (6.9664)$
- $d_3 = [1 – (3.0026 + 0.677*2)] = (-3.3566)$
- $d_4 = [12 – (3.0026 + 0.677*10)] = (2.2274)$
- $d_5 = [9 – (3.0026 + 0.677*11)] =(-1.4496)$
- $d_6 = [4 – (3.0026 + 0.677*3)] = (-1.0336)$
- $d_7 = [9 – (3.0026 + 0.677*6)] = (1.9354)$
- $d_8 = [6 – (3.0026 + 0.677*5)] = (-0.3876)$
- $d_9 = [1 – (3.0026 + 0.677*6)] = (-6.0646)$
- $d_{10} = [14 – (3.0026 + 0.677*8)] = (5.5814)$
- $\sum d^2 = (-4.4186)^2 + (6.9664)^2 + (-3.3566)^2 + (2.2274)^2 + (-1.4496)^2 + (-1.0336)^2 + (1.9354)^2 + (-0.3876)^2 + (-6.0646)^2 + (5.5814)^2 = 159.27990$

Multivariate Linear Regression

- Multivariate linear regression resembles simple linear regression except that in multivariate linear regression, *multiple independent variables* contribute to the dependent variables and so multiple coefficients are used in the computation.
- It is used to derive a *mathematical relationship* amongst *multiple* *random variables*. It explains how many multiple independent variables are associated with one dependent variable.

- The details of the *multiple independent variables* are used to make an accurate prediction of the influence they have on the outcome variable.
- Multivariate linear regression model generates a relationship in a linear form (a form of a straight line) with the best approximation of each data point.
- The equation of the Multivariate linear regression model is:

$$y = \beta_0 + \beta_1.x_1 + ... + \beta_n.x_n$$

Regression cost Function:

Regression models deal with predicting a continuous value for example salary of an employee, price of a car, loan prediction, etc.

- A cost function used in the regression problem is called "Regression Cost Function". They are calculated on the distance-based error as follows:
- Error = y-y'

- Where,

- Y – Actual Input
- Y’ – Predicted output

1 Mean Error (ME)

- In this cost function, the *error for each training data is calculated* and then the ***mean value of all these errors is derived***.
- Calculating the mean of the errors is the simplest and most intuitive way possible.
- The *errors can be both negative and positive*. So they can ***cancel***
  cancel each other out during summation giving zero mean error error for the model.
- Thus this is not a recommended cost function but it does lay the foundation for other cost functions of regression models.

2 Mean Squared Error (MSE)

- This improves the drawback we encountered in Mean Error above. Here a ***square of the difference between the actual and predicted value is calculated*** to avoid any possibility of negative error.
- It is measured as the average of the sum of squared differences between predictions and actual observations.
- MSE cost function
- MSE = (sum of squared errors)/n

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

- It is also known as **L2 loss**.
- In MSE, since each error is squared, it helps to penalize even small deviations in prediction when compared to MAE.
- But if our dataset has **outliers that contribute to larger prediction errors**, then squaring this error further will **magnify the error** many times more and lead to higher MSE error.
- Hence we can say that it ***is less robust to outliers***.

3 Mean Absolute Error (MAE)

- This cost function also addresses the shortcoming of mean error differently. Here an **absolute difference** between the actual and predicted value is calculated to **avoid any possibility of negative error**.
- So in this cost function, MAE is measured as the average of the sum of absolute differences between predictions and actual observations.
- MAE Cost function
- MAE = (sum of absolute errors)/n
- It is also known as L1 Loss.
- It is ***robust to outliers*** thus it will give better results even when our dataset

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

- **4. Root mean squared error (RMSE)**
- It is the square root of the mean of the square of all of the errors. Root Mean Square Error (RMSE) measures the error between two data sets. In other words, it compares an observed or known value and a predicted value.
- The lower the RMSE, the better a given model is able to "fit" a dataset.
- The formula to find the root mean square error, often abbreviated **RMSE**, is as follows:
- **RMSE =** $$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Regression Coefficients

- The aim of linear regression is to find the regression coefficients that produce the best-fitted line.
- The regression coefficients in linear regression help in predicting the value of an unknown variable using a known variable.
- **What are Regression Coefficients?**
- **Regression coefficients** can be defined as estimates of some unknown parameters to ***describe the relationship between a predictor variable and the corresponding response***.
- Linear regression is used to quantify how a unit change in an independent variable causes an effect in the dependent variable by determining the equation of the best-fitted straight line. This process is known as regression analysis.

Formula for Regression Coefficients

- The goal of linear regression is to find the equation of the straight line that best describes the relationship between two or more variables.
- Suppose the equation of the best-fitted line is given by Y = aX + b then, the regression coefficients formula is given as follows:

$$a = \frac{n(\sum xy)-(\sum x)(\sum y)}{n(\sum x^2)-(\sum x)^2}$$

$$b = \frac{(\sum y)(\sum x^2)-(\sum x)(\sum xy)}{n(\sum x^2)-(\sum x)^2}$$

- here, n refers to the number of data points in the given data sets.

How to Find Regression Coefficients?

- Before determining the regression coefficients to find the best-fitted line, it is necessary to check whether the variables follow a linear relationship or not.
- This can be done by using the correlation coefficient and interpreting the corresponding value. Given below are the steps to find the regression coefficients for regression analysis.
- To find the coefficient of X use the formula

$$a = \frac{n(\sum xy)-(\sum x)(\sum y)}{n(\sum x^2)-(\sum x)^2}.$$

$$b = \frac{(\sum y)(\sum x^2)-(\sum x)(\sum xy)}{n(\sum x^2)-(\sum x)^2}$$

- To find the constant term the formula is b
- Now input the regression coefficients in the equation Y = aX + b.
- A scatter plot can also be made so as to visually depict the regression line as shown below.

Why do we use Regression Analysis?

- Regression estimates the relationship between the target and the independent variable.
- It is used to find the trends in data.
- It helps to predict real/continuous values.
- By performing the regression, we can confidently determine the **most important factor, the least important factor, and how each factor is affecting the other factors**.

Polynomial Regression

- Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial. The Polynomial Regression equation is given below:

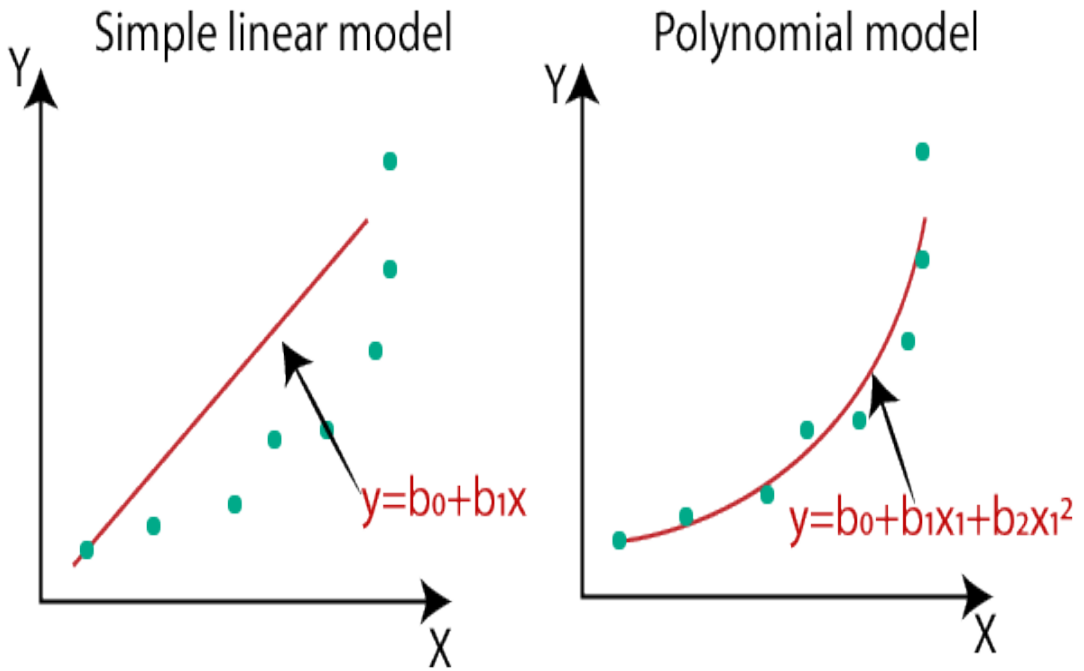
$$y= b_0+b_1x_1+ b_2x_1^2+ b_2x_1^3+.....b_nx_1^n$$

- The dataset used in Polynomial regression for training is of non-linear nature.
- It makes use of a linear regression model to fit the complicated and non- linear functions and datasets.
- ***In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model."***

Need for Polynomial Regression:

- If we apply a linear model on a **linear dataset**, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a **non-linear dataset**, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.

- So for such cases, **where data points are arranged in a non-linear fashion, we need the Polynomial Regression model**. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



Example

- There is a Human Resource company, which is going to hire a new candidate. The candidate has told his previous salary 160K per annum, and the HR have to check whether he is telling the truth or bluff.
- So to identify this, they only have a dataset of his previous company in which the salaries of the top 10 positions are mentioned with their levels.
- By checking the dataset available, we have found that there is a **non-linear relationship between the Position levels and the salaries**. Our goal is to build a **Bluffing detector regression** model, so HR can hire an honest candidate.

Position	Level(X-variable)	Salary(Y-Variable)
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Steps for Polynomial Regression:

- Data Pre-processing
- Build a Linear Regression model and fit it to the dataset
- Build a Polynomial Regression model and fit it to the dataset
- Visualize the result for Linear Regression and Polynomial Regression model.
- Predicting the output.

Example Code

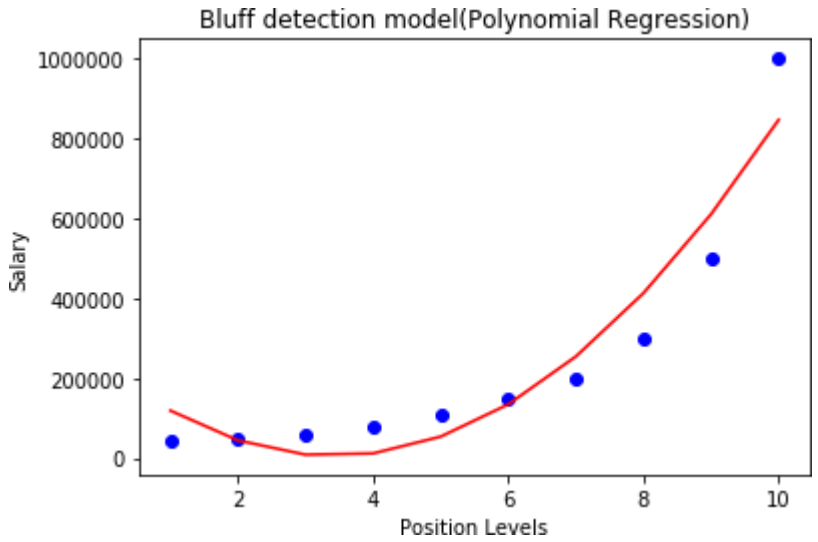
```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp


import pandas as pd #importing datasets
data_set= pd.read_csv('Position_Sala ries.csv')
```

- #Extracting Independent and depende nt Variable
- x= data_set.iloc[:, 1:2].values
- y= data_set.iloc[:, 2].values

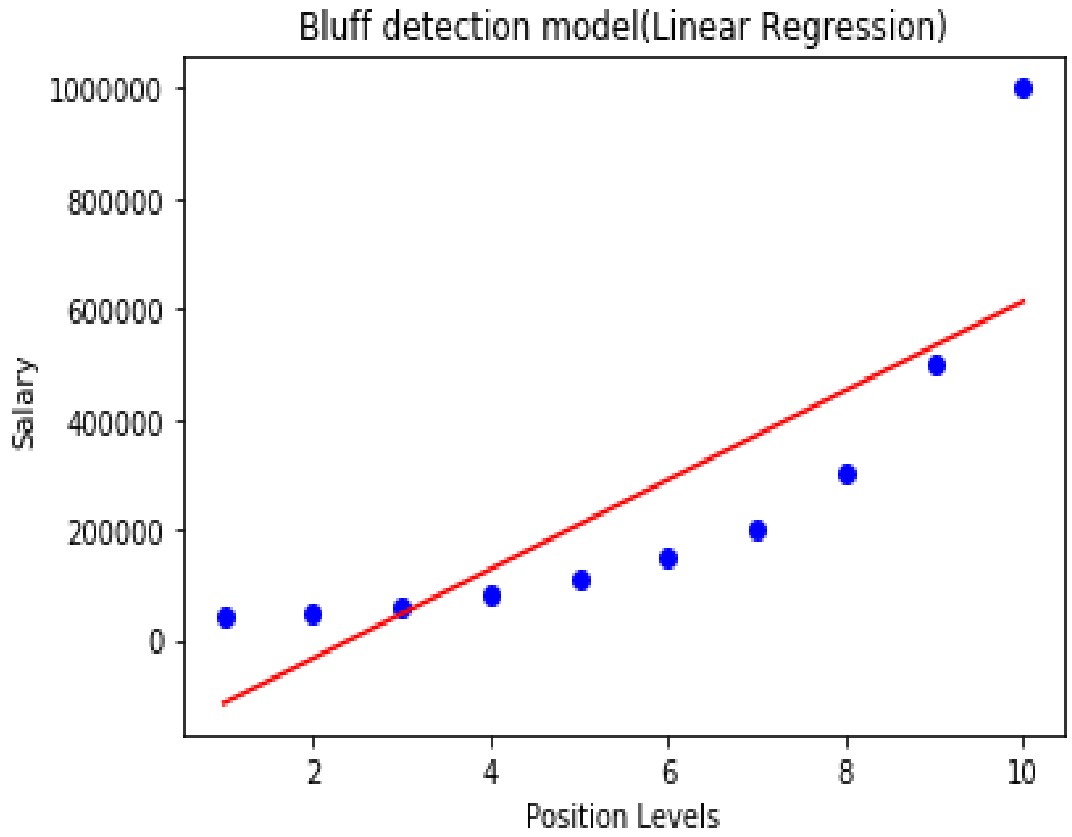
```
#Fitting the Linear Regression to the dat
aset
from sklearn.linear_model import Linea rRegression
lin_regs= LinearRegression() lin_regs.fit(x,y)
```

```
#Fitting the Polynomial regression to th e dataset
from sklearn.preprocessing import Poly
nomialFeatures
poly_regs= PolynomialFeatures(degree
= 2)
x_poly= poly_regs.fit_transform(x) lin_reg_2
=LinearRegression() lin_reg_2.fit(x_poly, y)
```

```
#Visulaizing the result for Polynomia l Regression
mtp.scatter(x,y,color="blue")
mtp.plot(x,  lin_reg_2.predict(poly_re gs.fit_transform(x)), color="red")
mtp.title("Bluff  detection  model(Poly nomial Regression)")
mtp.xlabel("Position Levels") mtp.ylabel("Salary")
mtp.show()
```



```
#Visulaizing the result for Linear Regression model
mtp.scatter(x,y,color="blue")
mtp.plot(x,lin_regs.predict(x), colo r="red")
mtp.title("Bluff detection model(Li near Regression)")
mtp.xlabel("Position Levels") mtp.ylabel("Salary")
mtp.show()
```



Decision Trees

Introduction

- Classification is a two-step process, ***learning step*** and

  prediction step, in machine learning.
- In the learning step, the model is developed based on given training data. In the prediction step, the model is used to the response for given data.
- Decision Tree is one of the easiest and popular classification algorithms to understand and interpret.
- The decision tree algorithm can be used for solving **regression and classification problems**

Decision Tree(CART)

- The ***goal*** of using a Decision Tree is to *create a training model* that can use to *predict the class* or value of the target variable by **learning simple decision rules** inferred from training data.
- In Decision Trees, for predicting a class label for a record we start from the **root** of the tree.
- We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Terminologies

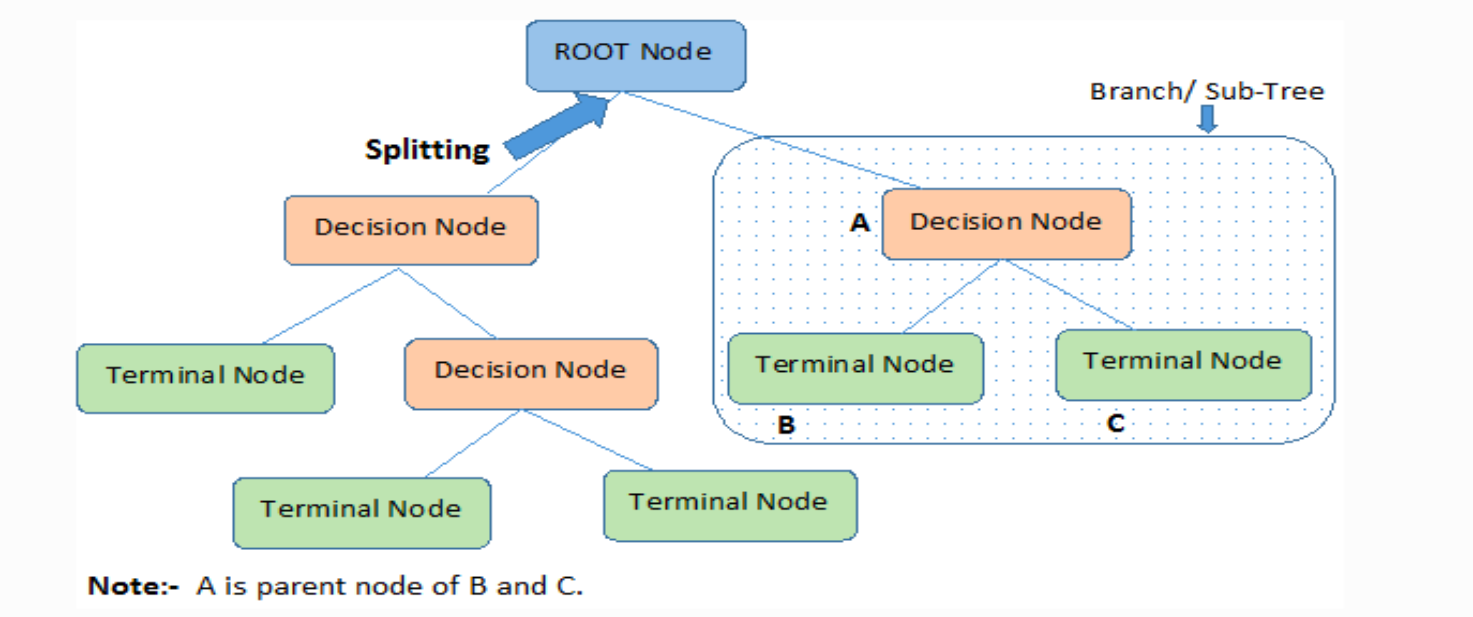
- **Important Terminology related to Decision Trees**

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning.

- By removing error-prone components, the classifier's performance may be improved

improved

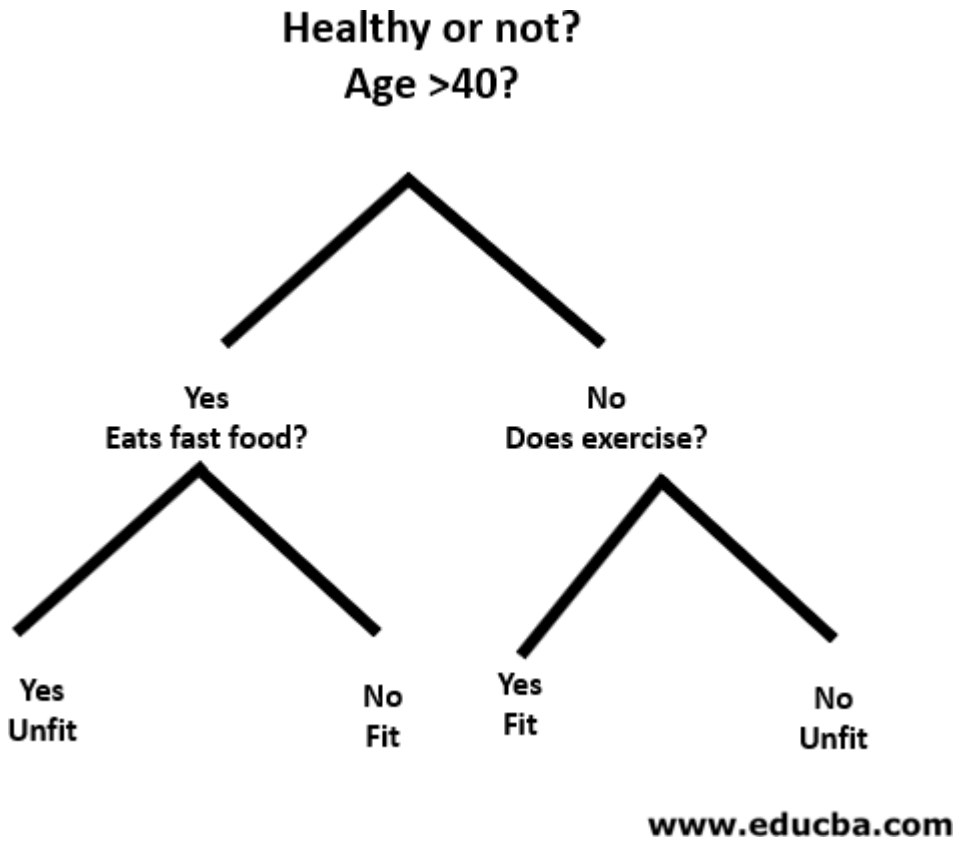
1. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
2. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



Decision Tree

- Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.
- **Each node in the tree acts as a test case** for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.
- The primary challenge in the decision tree implementation is to identify *which attributes do we need to consider as the root node and each level.*
- Handling this is to know as the **attributes selection**. We have different attributes selection measures to identify the attribute which can be considered as the root node at each level.

- Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.
- The creation of *sub-nodes increases* the homogeneity of resultant sub-nodes. In other words, we can say that the *purity of the node increases with respect to the target variable*. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

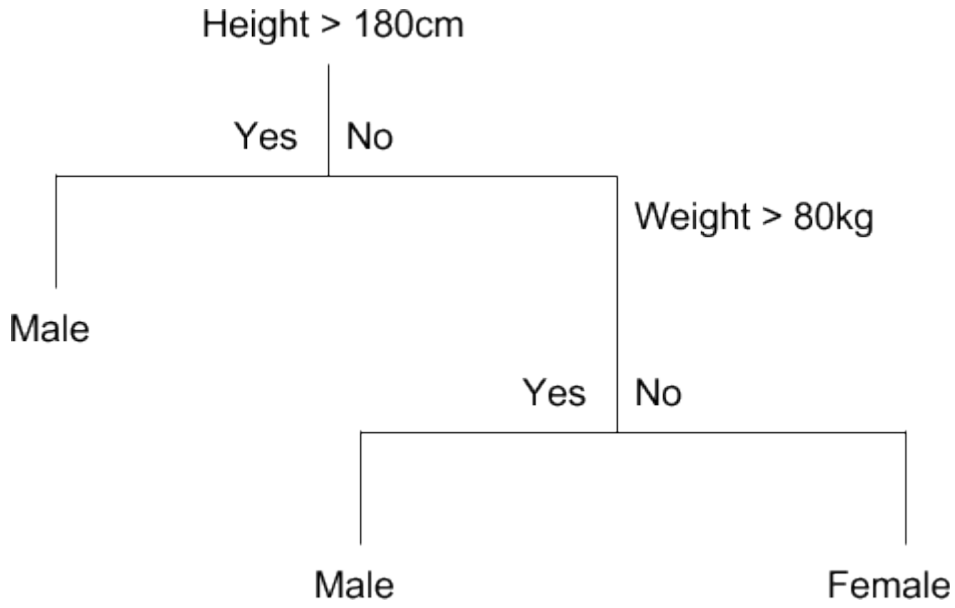


CART Model Representation

- The representation for the CART model is a *binary tree.*
- Each root node represents a single input variable (x) and a split point on that variable.
- The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.
- Given a dataset with two inputs (x) of height in centimeters and weight in kilograms the output of sex as male or female, below is a crude example of a binary decision tree (completely fictitious for demonstration purposes only).

- For example, given the input of [height = 160 cm, weight = 65 kg], we would traverse the above tree as follows:

- Height > 180 cm: No
- Weight > 80 kg: No
- Therefore: Female



- Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.
- A learned binary tree is actually a partitioning of the input space. You can think of each input variable as a dimension on a p- dimensional space. The decision tree split this up into rectangles (when p=2 input variables) or some kind of hyper- rectangles with more inputs.
- New data is filtered through the tree and lands in one of the rectangles and the output value for that rectangle is the prediction made by the model. This gives you some feeling for the type of decisions that a CART model is capable of making,
- For example, given the input of [height = 160 cm, weight = 65 kg], we would traverse the above tree as follows:

Learn a CART Model From Data

- Creating a CART model involves selecting input variables and split points on those variables until a suitable tree is constructed.
- *CART (Classification and Regression Trees) → uses Gini*

  Index(Classification) as metric.
- If all the data belong to a single class, then it can be called pure.
- Its Degree will be always between 0 and 1. ***If 0, means all data belongs to the single class/variable***. ***If 1, the data belong to the different class/field.***

GINI INDEX

- Gini Index is the metric for classification task in CART.
- Gini Index(Attribute=value) = $1-\sum_{i=1}(p_i)^2$
- Gini Index(Attribute)= $\sigma_{v=values}\ Pv \quad X \quad GI(v)$

Data set

There are 14 instances of golf playing decisions based on outlook, temperature, humidity and wind factors.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Gini index**
- Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.
    - Gini = $1 - \Sigma (Pi)^2$, for i=1 to number of classes

- **Outlook**
- Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = $1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$

Gini(Outlook=Overcast) = $1 - (4/4)^2 - (0/4)^2 = 0$

- Gini(Outlook=Rain) = $1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$
- Then, we will calculate weighted sum of gini indexes for outlook feature.
- Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342
-

- **Temperature**
- Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

- Gini(Temp=Hot) = $1 – (2/4)^2 – (2/4)^2 = 0.5$
- Gini(Temp=Cool) = $1 – (3/4)^2 – (1/4)^2 = 1 – 0.5625 – 0.0625 = 0.375$
- Gini(Temp=Mild) = $1 – (4/6)^2 – (2/6)^2 = 1 – 0.444 – 0.111 = 0.445$
- We'll calculate weighted sum of gini index for temperature feature
- Gini(Temp) = (4/14) x 0.5 + (4/14) x 0.375 + (6/14) x 0.445 = 0.142 + 0.107 + 0.190 = 0.439
- **Humidity**
- Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = $1 – (3/7)^2 – (4/7)^2 = 1 – 0.183 – 0.326 = 0.489$

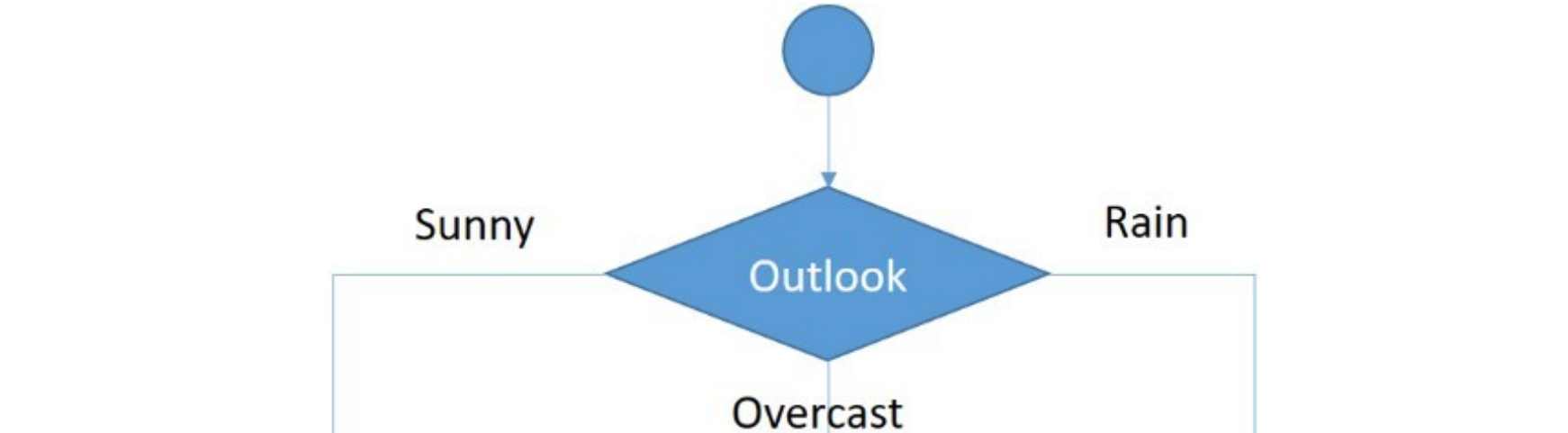
- Gini(Humidity=Normal) = $1 – (6/7)^2 – (1/7)^2 = 1 – 0.734 – 0.02 = 0.244$
- Weighted sum for humidity feature will be calculated next
- Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

- **Wind**
- Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

- Gini(Wind=Weak) = $1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$
- Gini(Wind=Strong) = $1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$
- Gini(Wind) = $(8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$
- **Time to decide**
- We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

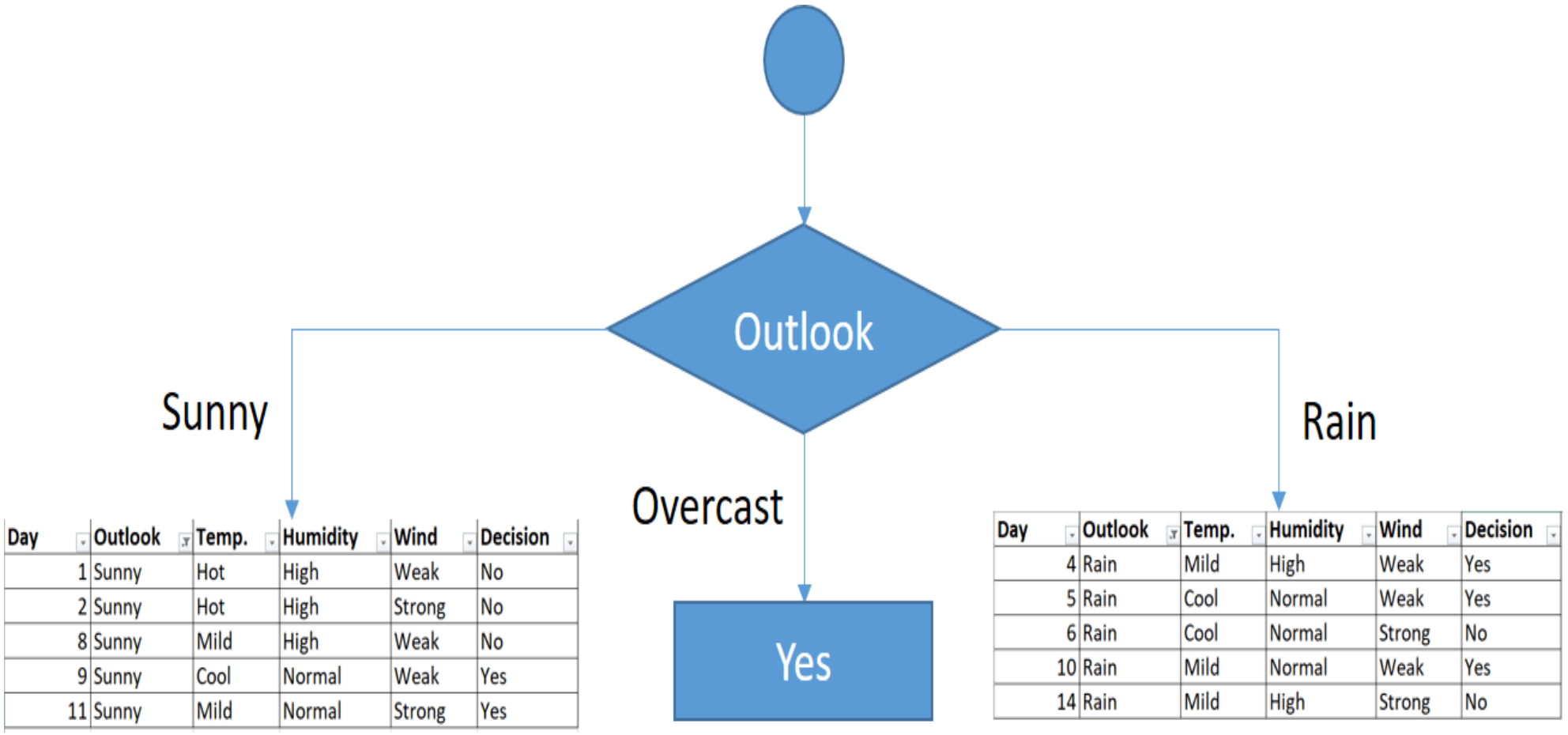
Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- Focus on the **sub dataset for sunny outlook**. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$ Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$ Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$ Gini(Outlook=Sunny and Temp.) = $(2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2$

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$

Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$

Gini(Outlook=Sunny and Humidity) = $(3/5)x0 + (2/5)x0 = 0$

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

$$\text{Gini(Outlook=Sunny and Wind=Weak)} = 1 - (1/3)^2 - (2/3)^2 = 0.266$$

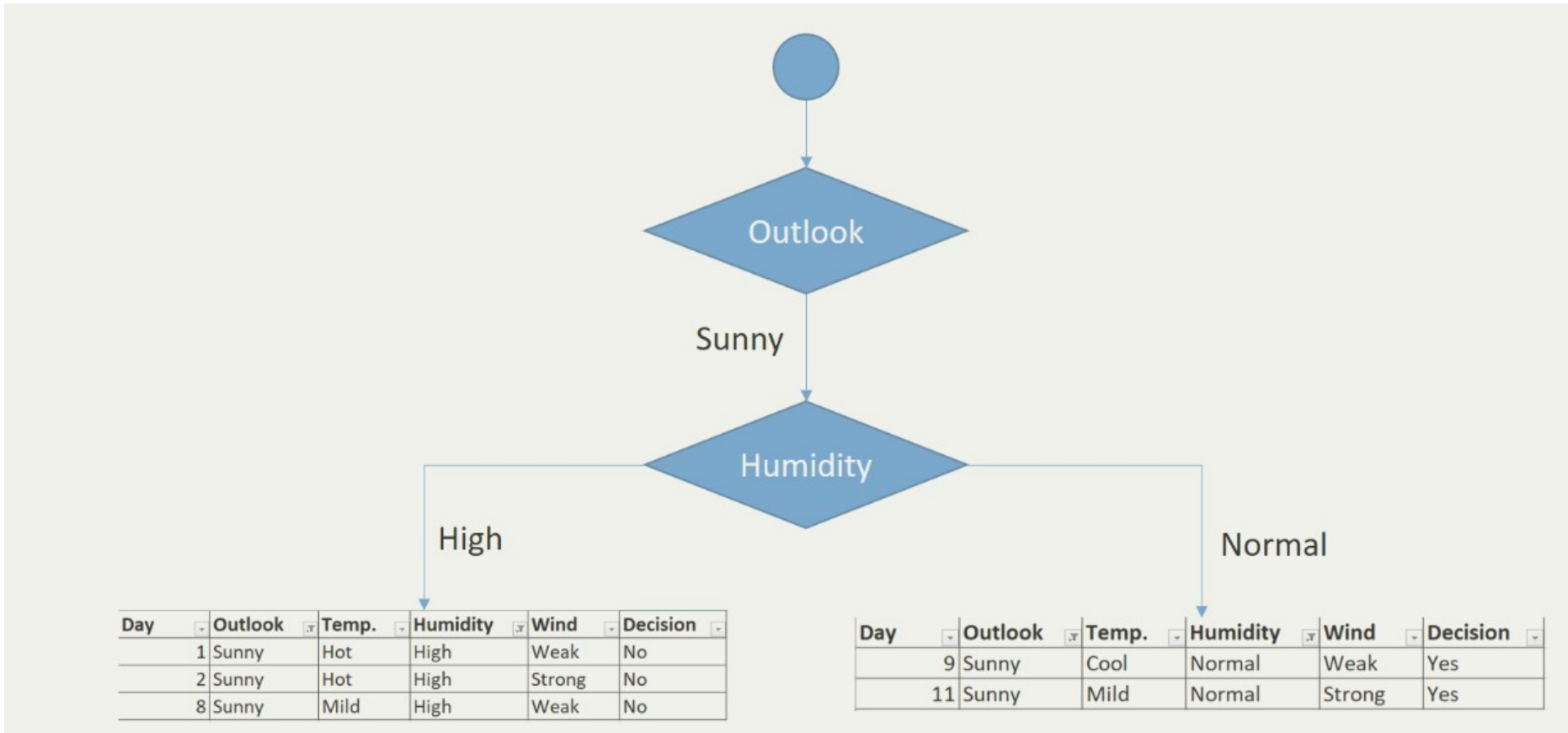
$$\text{Gini(Outlook=Sunny and Wind=Strong)} = 1 - (1/2)^2 - (1/2)^2 = 0.2$$

$$\text{Gini(Outlook=Sunny and Wind)} = (3/5)\times 0.266 + (2/5)\times 0.2 = 0.466$$

Decision for sunny outlook

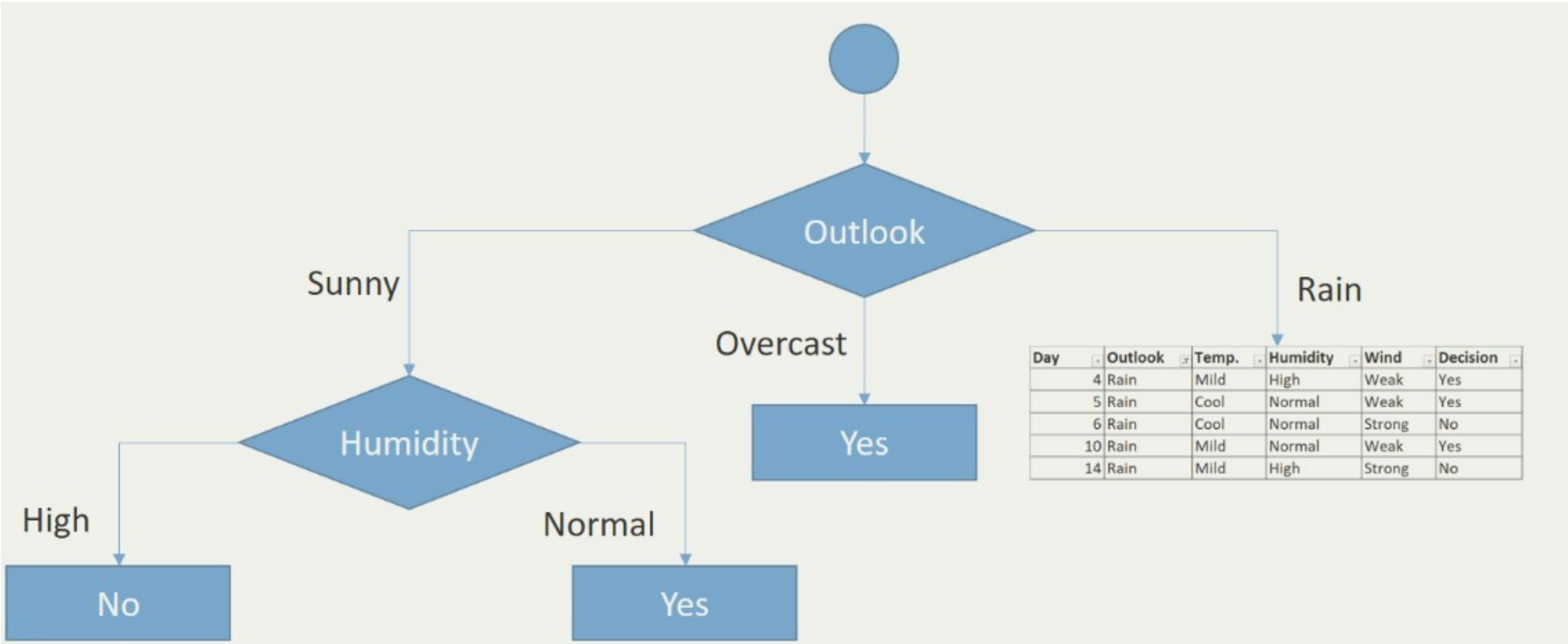
We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Temp.=Mild) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Temp.) = $(2/5)\times0.5 + (3/5)\times0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Humidity=Normal) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Humidity) = $(2/5)x0.5 + (3/5)x0.444 = 0.466$

Gini of wind for rain outlook

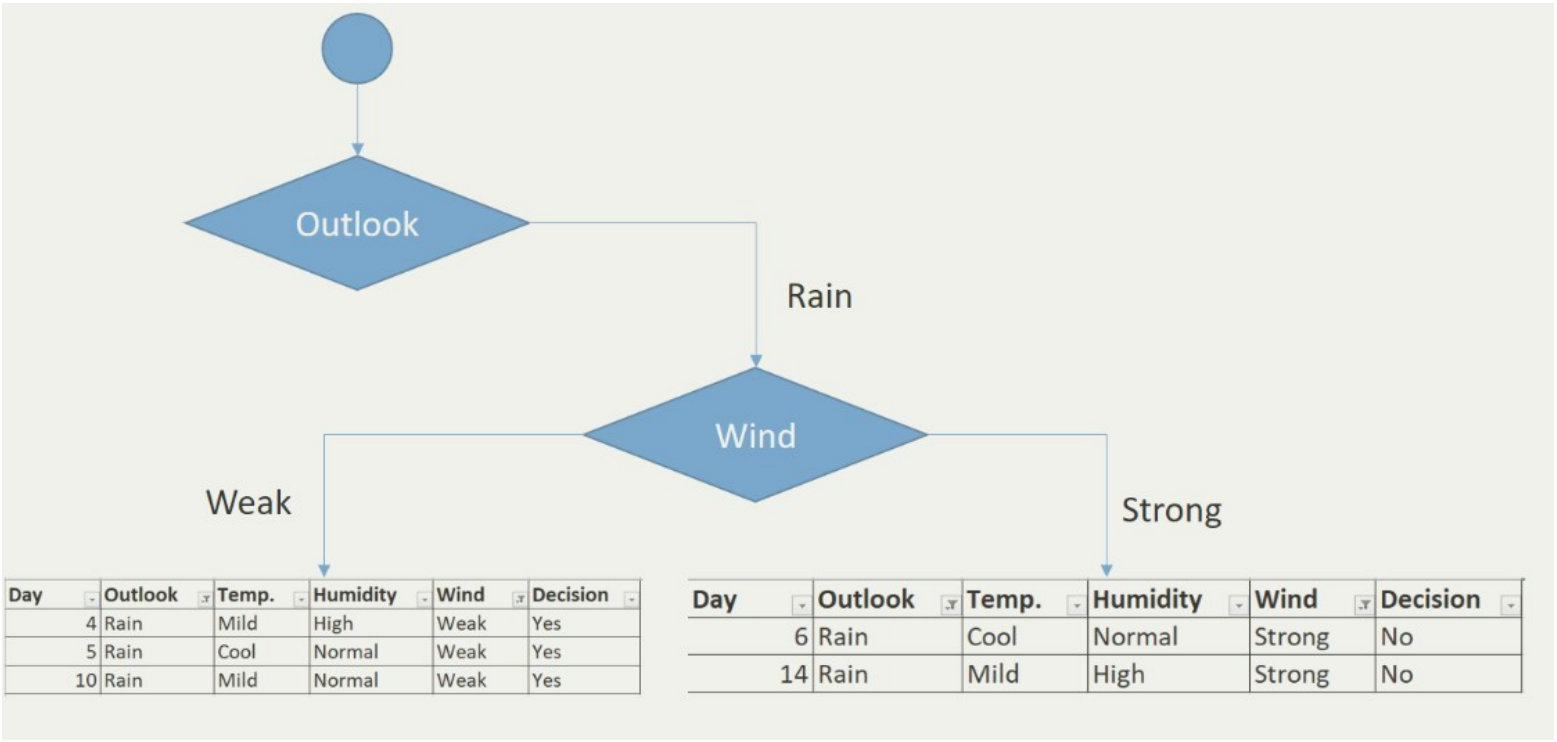
Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$

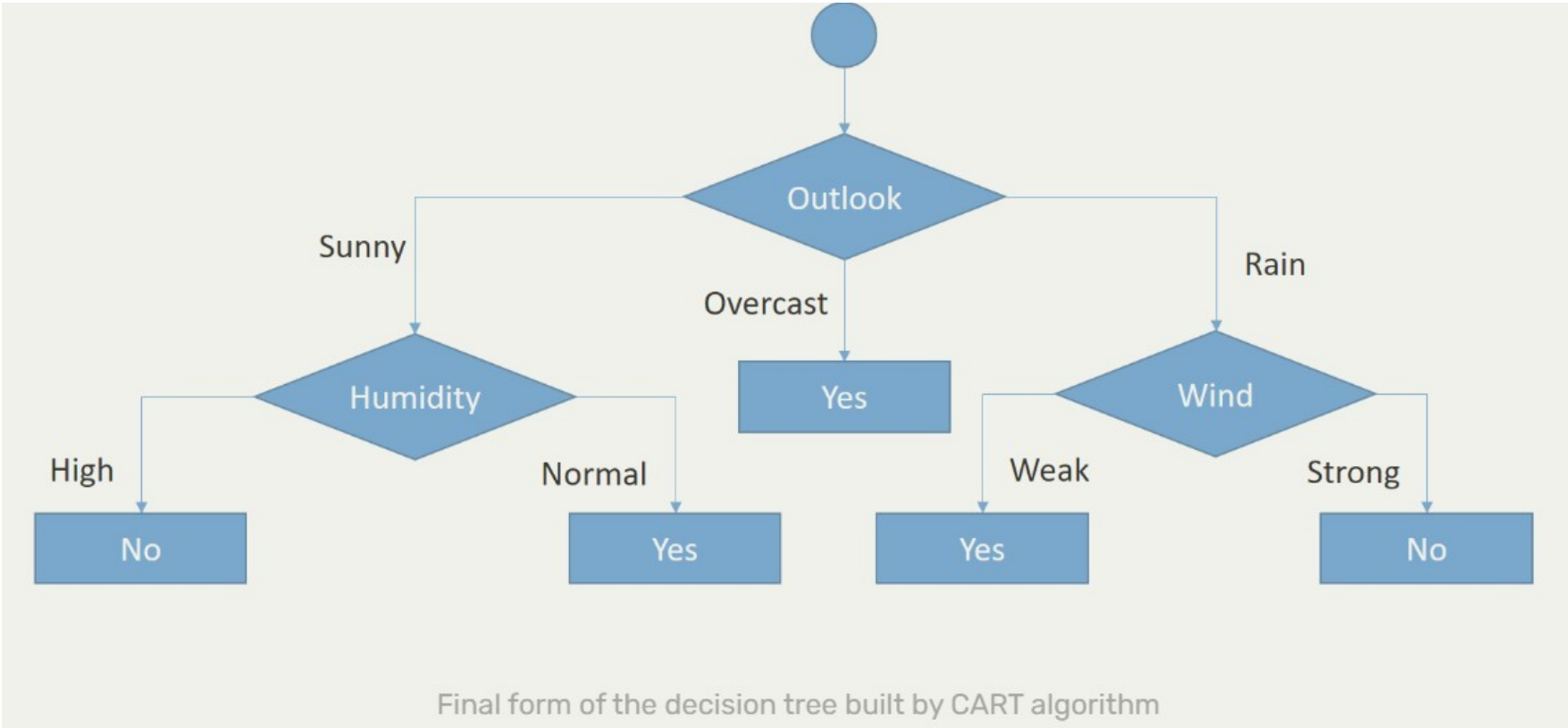
Gini(Outlook=Rain and Wind) = $(3/5)x0 + (2/5)x0 = 0$

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No



Final form of the decision tree built by CART algorithm

Entropy

- **Entropy** is a measure of disorder or *impurity in the given dataset.*
- In the decision tree, messy data are split based on values of the feature vector associated with each data point.
- With each *split, the data becomes more homogenous* which will decrease the entropy.
- The higher the entropy, the harder it is to draw any conclusion. When the tree finally reaches the terminal or leaf node maximum purity is added.For a dataset that has C classes and the probability of randomly choosing data from class, i is Pi. Then entropy E(S) can be mathematically represented as

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

- If we have a dataset of 10 observations belonging to two classes YES and NO. If 6 observations belong to the class, YES, and 4 observations belong to class NO, then entropy can be written as below.

$$E(S) = -(P_{yes}\log_2 P_{yes} + P_{no}\log_2 P_{no})$$

- *Pyes* is the probability of choosing Yes and *Pno* is the probability of choosing a No. Here *Pyes* is 6/10 and *Pno* is 4/10.

$$E(S) = -\ (6/10 * \log_2 * 6/10 + 4/10 * \log_2 * 4/10) \approx 0.971$$

If all the 10 observations belong to 1 class then entropy will be equal to zero. Which implies the node is a pure node.

$$E(S) = -\ (1 \log_2 1) = 0$$

If both classes YES and NO have an equal number of observations, then entropy will be equal to 1.

$$E = -\ (\frac{5}{10} * \log_2 \frac{5}{10} + \frac{5}{10} * \log_2 \frac{5}{10}) = -2(0.5 \log_2 0.5) = 1$$

Information Gain

- The Information Gain measures the expected reduction in entropy.
- Entropy measures impurity in the data and information gain measures ***reduction in impurity in the data***.
- The feature which has minimum impurity will be considered as the root node.
- Information gain is used to decide which feature to split on at each step in building the tree.

- Information gain of a parent node can be calculated as the entropy of the parent node is the subtracted entropy of the weighted average of the child node.
- As per the above example, the dataset has 10 observations belonging to two classes YES and NO. Where 6 observations belong to the class, YES, and 4 observations belong to class NO.

Observations	COLOR	Outcome
1	Red	Yes
2	Red	No
3	Yellow	Yes
4	Yellow	Yes
5	Red	Yes
6	Yellow	Yes
7	Red	No
8	Red	No
9	Red	Yes
10	Yellow	No

- Red color has 3 Yes outcome and 3 No outcome whereas yellow has 3 Yes outcome and 1 No outcome.
- E(S), we have already calculated and it is approximately equal to 0.971
-

$$E(S_{Red}) = -\left(\frac{3}{6} * \log_2 \frac{3}{6} + \frac{3}{6} * \log_2 \frac{3}{6}\right) = 1$$

$$E(S_{Yellow}) = -\left(\frac{3}{4} * \log_2 \frac{3}{4} + \frac{1}{4} * \log_2 \frac{1}{4}\right) \approx 0.811$$

$$\text{Weighted average} = \frac{6}{10} * E(S_{Red}) + \frac{4}{10} * E(S_{Yellow})$$

$$= \frac{6}{10} * 1 + \frac{4}{10} * 0.811$$

$$= 0.924$$

$$\text{Information Gain (S, Color)} = E(S) - \text{Weighted Average}$$

$$= 0.971 - 0.924 \approx -0.047$$

- For a dataset having many features, the information gain of each feature is calculated. The feature having maximum information gain will be the most important feature which will be the root node for the decision tree.

Decision Tree (ID3 Algorithm)

- ID3 or **Iterative Dichotomiser3 Algorithm** is used in machine learning for building decision trees from a given dataset. It was developed in **1986** by **Ross Quinlan**.
- It is a greedy algorithm that builds a decision tree by recursively partitioning the data set into smaller and smaller subsets until all data points in each subset belong to the same class.
- It employs a top-down approach, recursively selecting features to split the dataset based on information gain.
- used for both classification and regression tasks.ID3 deals primarily with categorical properties, which means that it can efficiently handle objects with a discrete set of values. This property is consistent with its suitability for problems where the input features are categorical rather han continuous

S teps of the ID3 A lg o rithm

1. **Determine entropy** for the overall the dataset using class distribution.

For each feature.

Calculate **Entropy for Categorical Values**.

Assess **information gain** for each unique categorical value of the feature.

Choose the feature that generates **highest information gain**.

Iteratively apply all above steps to build the decision tree structure.

```
def ID3(D, A):
  if D is pure or A is empty:
    return a leaf node with the majority class in D
  else:
    A_best = argmax(InformationGain(D, A)) root = Node(A_best)
    for v in values(A_best):
      D_v = subset(D, A_best, v) child = ID3(D_v, A -
      {A_best}) root.add_child(v, child)
    return root
```

Applications of ID3

1. **Fraud detection:** ID3 can be used to develop models that can detect fraudulent transactions or activities.
2. **Medical diagnosis:** ID3 can be used to develop models that can diagnose diseases or medical conditions.
3. **Customer segmentation:** ID3 can be used to segment customers into different groups based on their demographics, purchase history, or other factors.
4. **Risk assessment:** ID3 can be used to assess risk in a variety of different areas, such as insurance, finance, and healthcare.
5. **Recommendation systems:** ID3 can be used to develop recommendation systems that can recommend products, services, or content to users based on their past behavior or preferences.

Example:

- Build a **decision tree** using ID3 algorithm for the given training data in the table (Buy Computer data), and predict the class of the following new example: **age<=30, income=medium, student=yes, credit-rating=fair**

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

- First, check which attribute provides the highest Information Gain in order to split the training set based on that attribute. We need to calculate the expected information to classify the set and the entropy of each attribute.
- The mutual information of the two classes,
- Entropy(S)= E(9,5)= $-9/14 \log_2(9/14) - 5/14 \log_2(5/14)=0.94$
- **Now Consider the Age attribute**
- For Age, we have three values $age_{<=30}$ (2 yes and 3 no), $age_{31..40}$ (4 yes and 0 no), and $age_{>40}$ (3 yes and 2 no)
- Entropy(age) = $5/14 (-2/5 \log_2(2/5)-3/5\log_2(3/5)) + 4/14 (0) + 5/14 (-3/5\log_2(3/5)-2/5\log_2(2/5))$
- $= 5/14(0.9709) + 0 + 5/14(0.9709) = 0.6935$
- Gain(age) = 0.94 – 0.6935 = 0.2465
- **Next, consider Income Attribute**
- For Income, we have three values $income_{high}$ (2 yes and 2 no), $income_{medium}$ (4 yes and 2 no), and $income_{low}$ (3 yes 1 no)

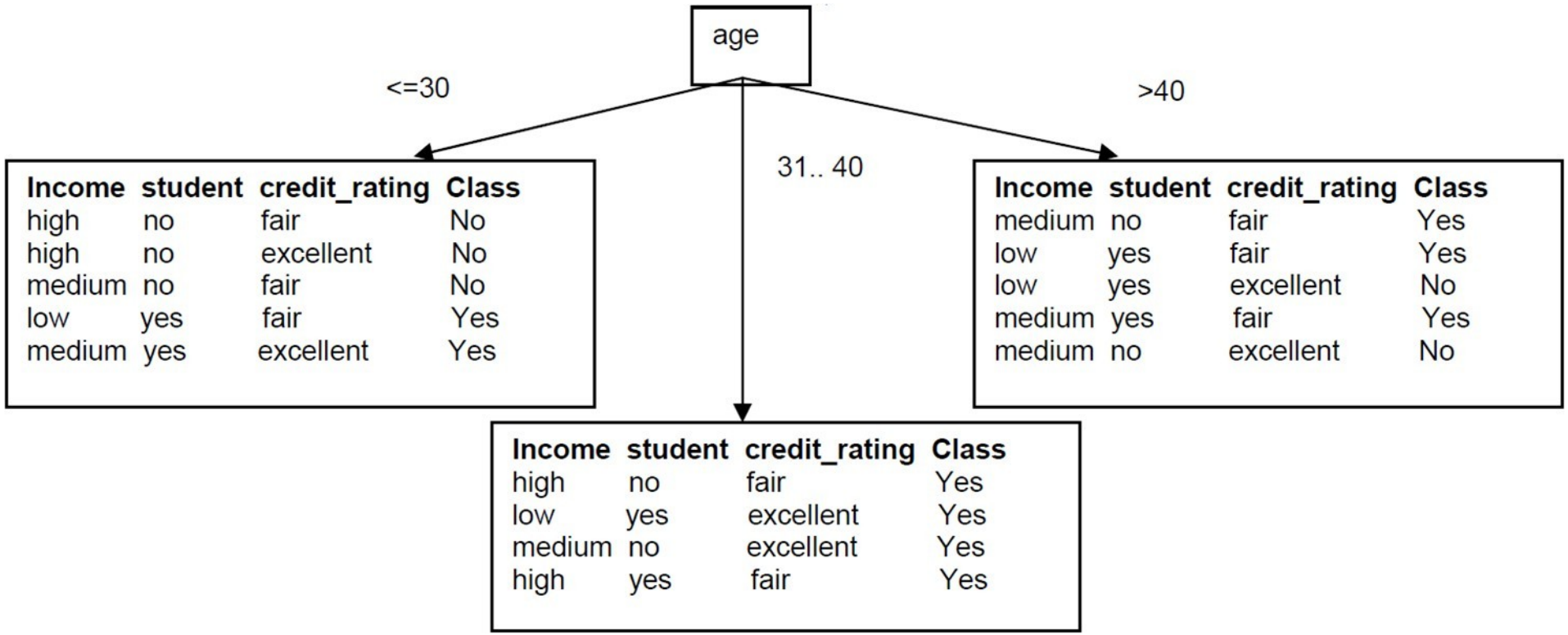
- Entropy(income) = $4/14(-2/4\log_2(2/4)-2/4\log_2(2/4)) + 6/14 (- 4/6\log_2(4/6)-2/6\log_2(2/6)) + 4/14 (-3/4\log2(3/4)-1/4\log2(1/4))$
- = 4/14 (1) + 6/14 (0.918) + 4/14 (0.811)
- = 0.285714 + 0.393428 + 0.231714 = 0.9108
- Gain(income) = 0.94 – 0.9108 = 0.0292
- **Next, consider Student Attribute**
- For Student, we have two values student_{yes} (6 yes and 1 no) and student_{no} (3 yes 4 no)

Entropy(student) = $7/14(-6/7\log_2(6/7)-1/7\log_2(1/7)) + 7/14(-3/7\log_2(3/7)-4/7\log_2(4/7)$
= 7/14(0.5916) + 7/14(0.9852)
= 0.2958 + 0.4926 = 0.7884
Gain (student) = 0.94 – 0.7884 = 0.1516
Finally, consider Credit_Rating Attribute
For Credit_Rating we have two values credit_ratingfair (6 yes and 2 no) and credit_ratingexcellent (3 yes 3 no)
Entropy(credit_rating) = $8/14(-6/8\log_2(6/8)-2/8\log_2(2/8)) + 6/14(-3/6\log_2(3/6)- 3/6\log_2(3/6))$
= 8/14(0.8112) + 6/14(1)

= 0.4635 + 0.4285 = 0.8920

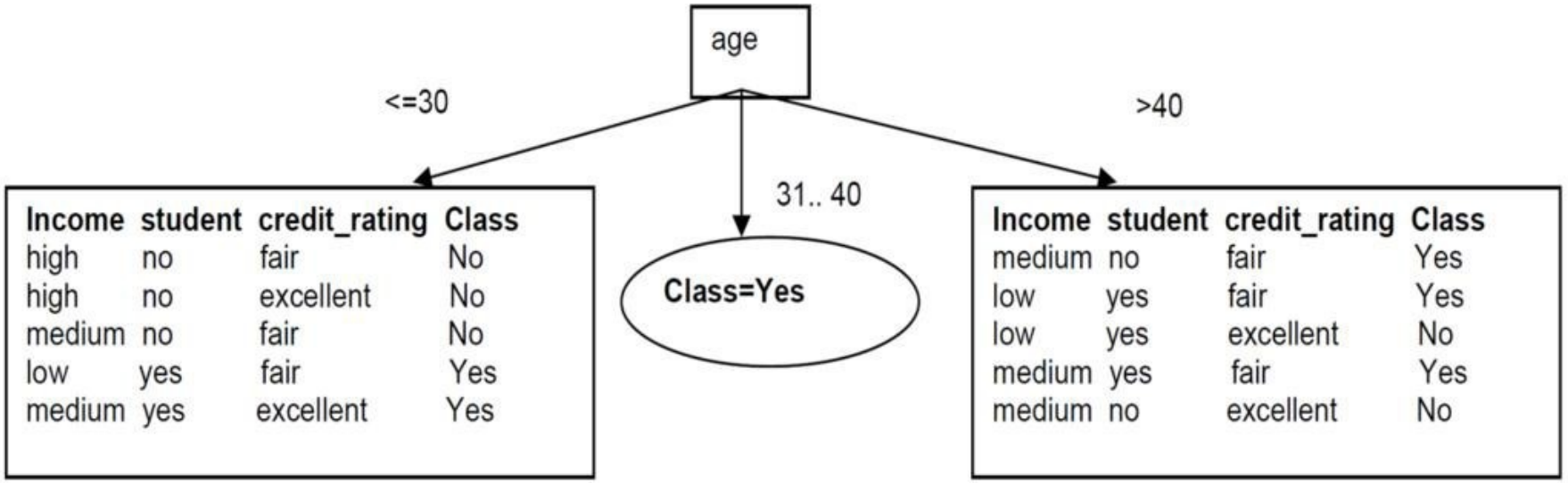
- Gain(credit_rating) = 0.94 – 0.8920 = 0.479
- **Since Age has the highest Information Gain we start splitting the dataset using the age attribute.**

Decision Tree after step 1



Decision Tree after step 1_1

- Since all records under the branch age31..40 are all of the class, Yes, we can replace the leaf with Class=Yes



Income	student	credit_rating	Class
high	no	fair	No
high	no	excellent	No
medium	no	fair	No
low	yes	fair	Yes
medium	yes	excellent	Yes

Income	student	credit_rating	Class
medium	no	fair	Yes
low	yes	fair	Yes
low	yes	excellent	No
medium	yes	fair	Yes
medium	no	excellent	No

- **Now build the decision tree for the left subtree**
- **The same process of splitting has to happen for the two remaining branches.**

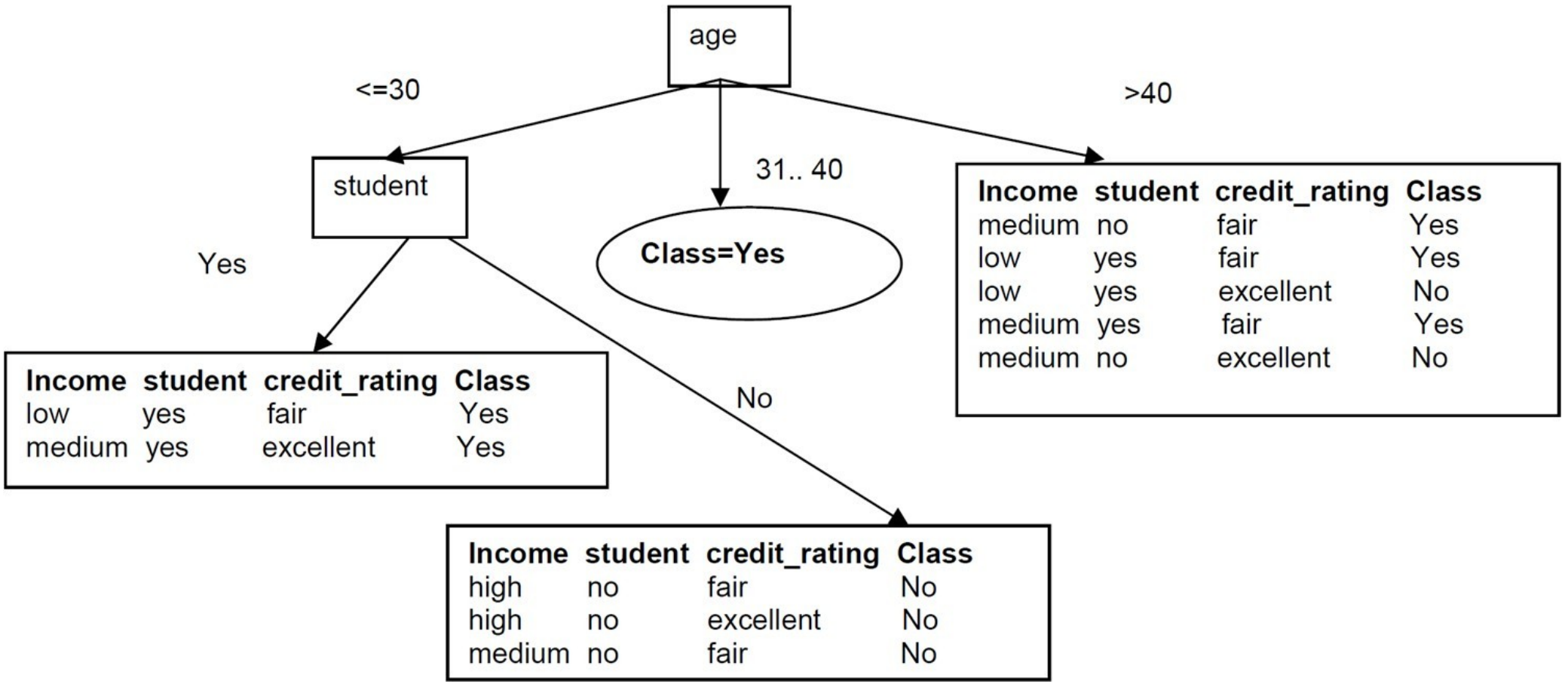
Income	student	credit_rating	Class
high	no	fair	No
high	no	excellent	No
medium	no	fair	No
low	yes	fair	Yes
medium	yes	excellent	Yes

- For branch age<=30 we still have attributes income, student, and credit_rating. Which one should be used to split the partition?
- The mutual information is $E(S_{age<=30})= E(2,3)= -2/5 \log_2(2/5) – 3/5 \log_2(3/5)=0.97$
- For **Income,** we have three values $income_{high}$ (0 yes and 2 no), $income_{medium}$ (1 yes and 1 no) and $income_{low}$ (1 yes and 0 no)
- Entropy(income) = $2/5(0) + 2/5 (-1/2\log_2(1/2)-1/2\log_2(1/2)) + 1/5 (0)$

  $= 2/5 (1) = 0.4$

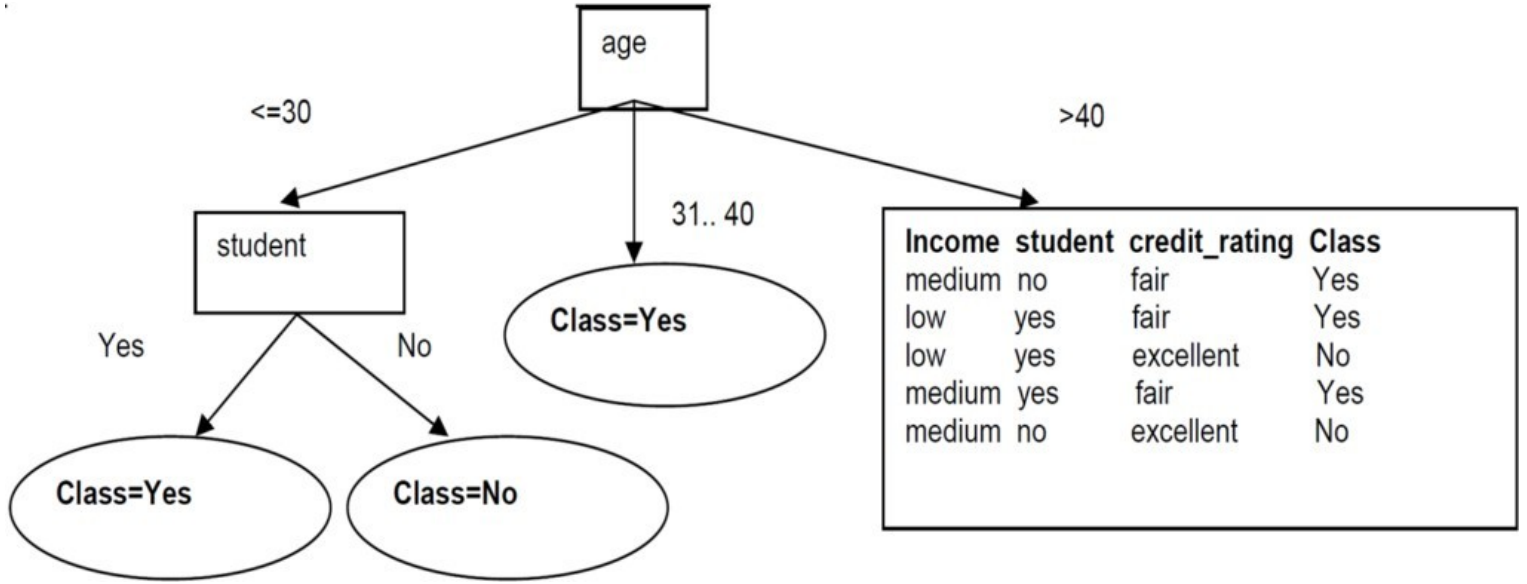
- Gain(income) = 0.97 – 0.4 = 0.57
- For Student, we have two values studentyes (2 yes and 0 no) and studentno (0 yes 3 no)
- Entropy(student) = 2/5(0) + 3/5(0) = 0
- Gain (student) = 0.97 – 0 = 0.97
- We can then safely split on attribute student without checking the other attributes since the information gain is maximized.

Decision Tree after step 2



Decision Tree after step 2_2

- Since these two new branches are from distinct classes, we make them into leaf nodes with their respective class as label:



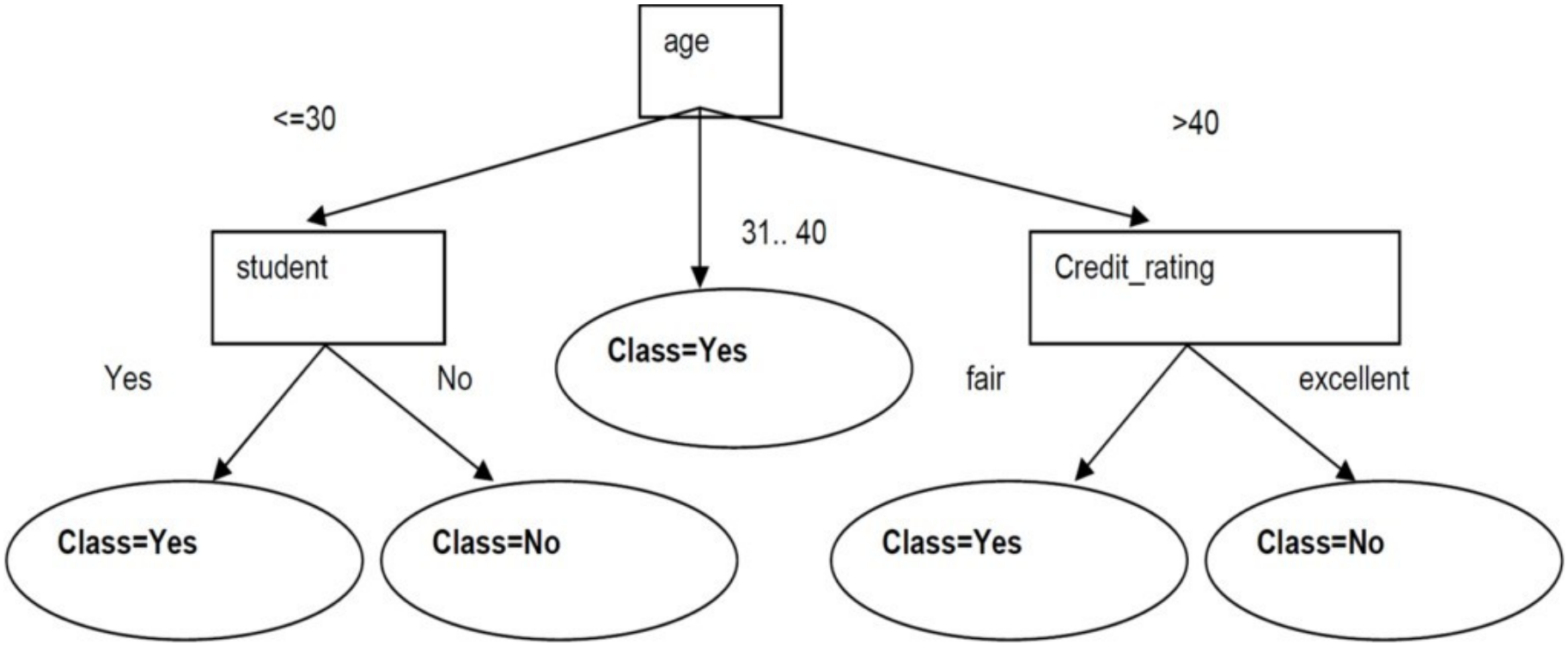
Right sub-branch

- **Now build**

Income	student	credit_rating	Class
medium	no	fair	Yes
low	yes	fair	Yes
low	yes	excellent	No
medium	yes	fair	Yes
medium	no	excellent	No

- he mutual information is Entropy($S_{age>40}$)= I(3,2)= -3/5 $\log_2$(3/5) – 2/5 $\log_2$(2/5)=0.97
- For **Income**, we have two values $income_{medium}$ (2 yes and 1 no) and $income_{low}$ (1 yes and 1 no)
- Entropy(income) = 3/5(-2/3log2(2/3)-1/3log2(1/3)) + 2/5 (-1/2log2(1/2)-1/2log2(1/2))
- = 3/5(0.9182)+2/5 (1) = 0.55+0. 4= 0.95
- Gain(income) = 0.97 – 0.95 = 0.02
- For **Student**, we have two values $student_{yes}$ (2 yes and 1 no) and $student_{no}$ (1 yes and 1 no)
- Entropy(student) = 3/5(-2/3$\log_2$(2/3)-1/3$\log_2$(1/3)) + 2/5(-1/2$\log_2$(1/2)-1/2$\log_2$(1/2)) = 0.95
- Gain (student) = 0.97 – 0.95 = 0.02
- For **Credit_Rating**, we have two values credit_ratingfair (3 yes and 0 no) and credit_ratingexcellent (0 yes and 2 no)
- Entropy(credit_rating) = 0
- Gain(credit_rating) = 0.97 – 0 = 0.97

- We then split based on credit_rating. These splits give partitions each with records from the same class. We just need to make these into leaf nodes with their class label attached:



- New example: age<=30, income=medium, student=yes, credit- rating=fair
- Follow branch(age<=30) then student=yes we predict Class=yes
- **Buys_computer = yes**

Module 3

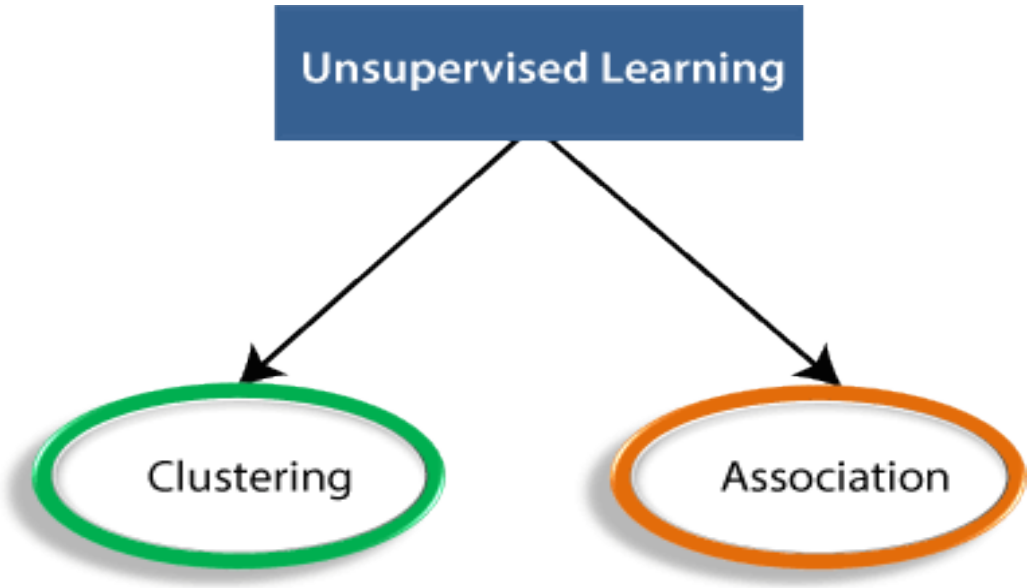
Similarity based Models

k-Nearest Neighbors (k-NN): Introduction to k-Nearest Neighbors Algorithm-Distance Metrics and Choosing 'k' -k-NN for Classification. Logistic Regression- Multiclass Classification with Logistic Regression.

Unsupervised learning Model

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to ***analyze and cluster unlabeled datasets***.
- These algorithms *discover hidden patterns or data groupings*

  without the need for human intervention.
- Its ability to *discover similarities and differences in information* make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.



Clustering

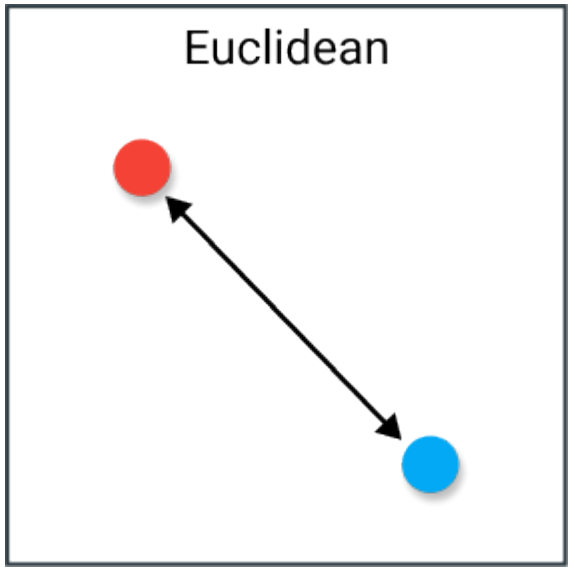
- Unsupervised learning models are utilized for three main tasks—*clustering, association, and dimensionality reduction.*
- **Clustering**
- Clustering is a data mining technique which groups unlabeled data based on their similarities or differences.
- Clustering algorithms are used to process *raw, unclassified data objects into groups* represented by structures or patterns in the information.

Distance Metrics

- Choosing a good distance metric becomes really important here. The *distance metric helps algorithms to recognize similarities between the contents.*
- *A distance function provides distance between the elements of a set. If the* ***distance is zero then elements are equivalent else they are different from each other.***

- **Euclidean Distance**
- It is a distance measure which can be explained as the length of a segment connecting two points.
- The distance is calculated from the cartesian coordinates of the points using the Pythagorean theorem.

$$D(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2)}$$



- # calculating euclidean distance between vectors
- from scipy.spatial.distance import euclidean
- # define data
- row1 = [10, 20, 15, 10, 5]
- row2 = [12, 24, 18, 8, 7]
- # calculate distance
- dist = euclidean(row1, row2)
- print(dist)
- 6.082762530298219

Disadvantages

1. *Euclidean distance is not scale in-variant* which means that distances computed might be skewed depending on the units of the features.

- Typically, one needs to normalize the data before using this distance measure.

2. As the dimensionality increases of your data, the less useful Euclidean distance becomes.

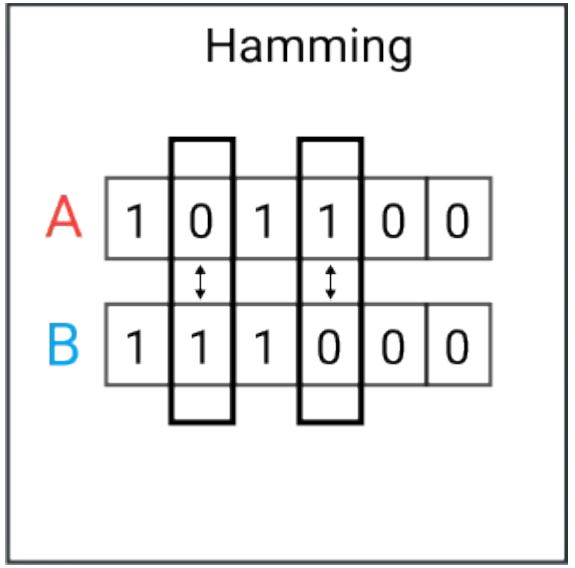
- This has to do with the curse of dimensionality which relates to the notion that higher-dimensional space does not act as we would, intuitively, expect from 2- or 3-dimensional space.

Advantages

- Euclidean distance works great when you have low-dimensional data and the magnitude of the vectors is important to be measured.
- It is incredibly intuitive to use, *simple to implement and shows great results* in many use-cases.

Hamming Distance

- Hamming distance is the number of values that are different between two vectors.
- It is typically used to compare two binary strings of equal length.
- It can also be used for strings to compare how similar they are to each other by calculating the number of characters that are different from each other.



- The Hamming distance between two strings, a and b is denoted as d(a,b).
- In order to calculate the Hamming distance between two strings, and, we perform their XOR operation, (a⊕ b), and then count the total number of 1s in the resultant string.
- Suppose there are two strings 11011001 and 10011101.
- 11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.

Disadvantages

- As you might expect, *hamming distance is difficult to use when two vectors are not of equal length*.
- You would want to compare same-length vectors with each other in order to understand which positions do not match.
- It does not take the actual value into account as long as they are different or equal.
- Therefore, *it is not advised to use this distance measure when the magnitude is an important measure*.
- For bitstrings that may have many 1 bits, it is more common to calculate the average number of bit differences to give a hamming distance score between 0 (identical) and 1 (all different).
- HammingDistance = (sum for i to N abs(v1[i] – v2[i])) / N
- # calculating hamming distance between bit strings
- from scipy.spatial.distance import hamming
- # define data
- row1 = [0, 0, 0, 0, 0, 1]
- row2 = [0, 0, 0, 0, 1, 0]
- # calculate distance
- dist = hamming(row1, row2)
- print(dist)

```
1  0.3333333333333333
```

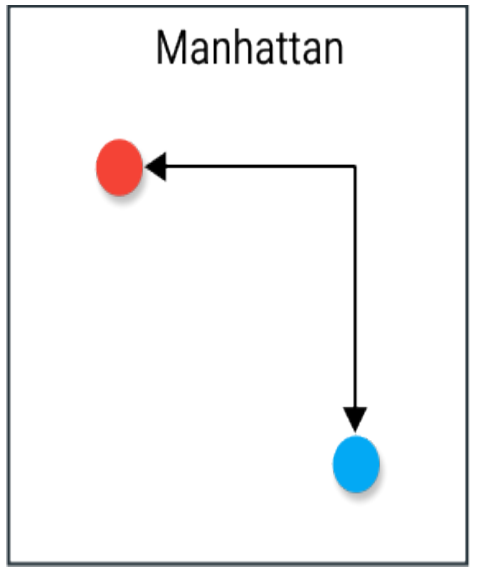
Uses

- Typical use cases include **error correction/detection** when data is transmitted over computer networks.
- It can be used to determine the **number of distorted bits** in a binary word as a way to estimate error.
- Moreover, you can also use Hamming distance to measure the ***distance between categorical variables***.

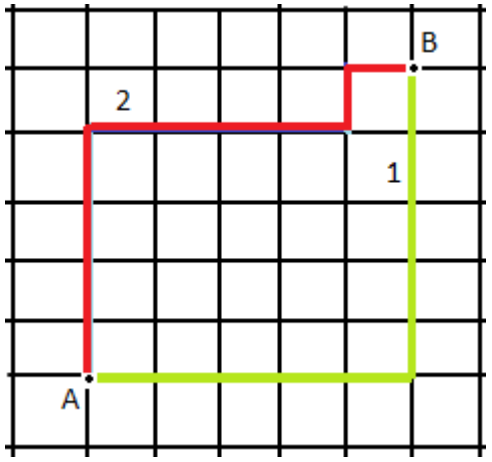
Manhattan Distance

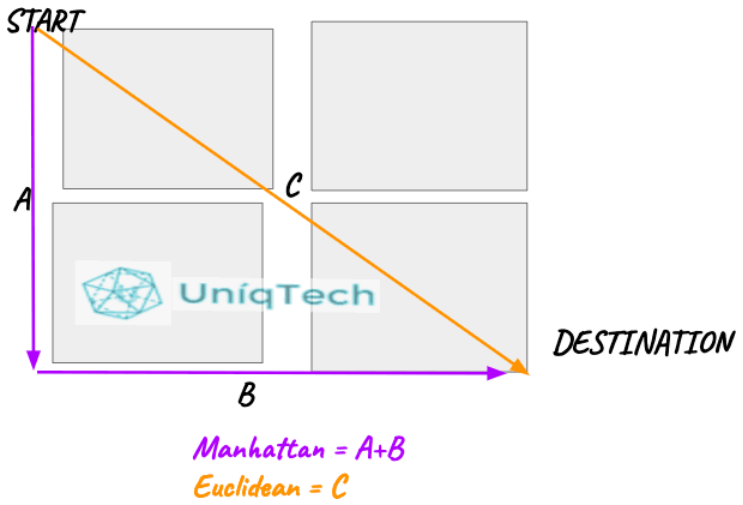
- The Manhattan distance, often called Taxicab distance or City Block distance, calculates the distance between real-valued vectors.
- Imagine vectors that describe objects on a uniform grid such as a chessboard. Manhattan distance then refers to the distance between two vectors if they could only move right angles.

$$D(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$



- In the given picture, imagine each cell to be a building, and the grid lines to be roads.
- Now if I want to travel from Point A to Point B marked in the image and follow the red or the yellow path.
- We see that the path is not straight and there are turns. In this case, we use the Manhattan distance metric to calculate the distance walked.





```
# calculating manhattan distance between vectors
from scipy.spatial.distance import cityblock
# define data
row1 = [10, 20, 15, 10, 5]
row2 = [12, 24, 18, 8, 7]
# calculate distance
dist = cityblock(row1, row2)
print(dist)
```

Output:

13

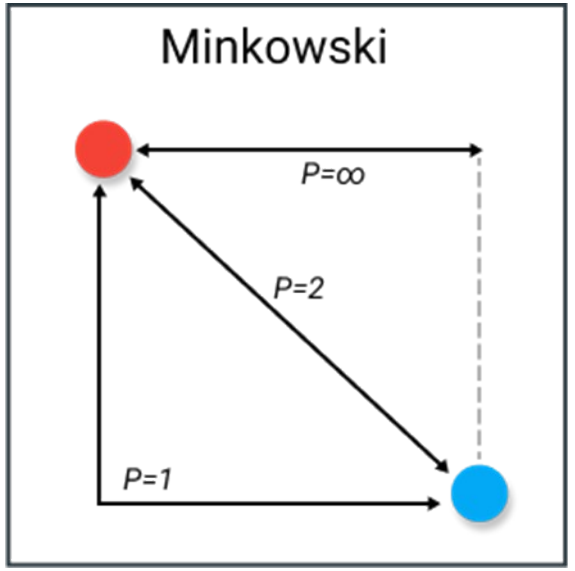
- **Disadvantages**

  It is more likely to give a higher distance value than Euclidean distance since it does not use the shortest path possible.

- **Advantages**
- When your dataset has discrete and/or binary attributes, Manhattan seems to work quite well *since it takes into account the paths that realistically could be taken* within values of those attributes.
- Take Euclidean distance, for example, would create a straight line between two vectors when in reality this might not actually be possible.

Minkowski

- Minkowski distance is a distance measured between two points in N- dimensional space.
- It is basically a generalization of the Euclidean distance and the Manhattan distance.
- It is widely used in the field of Machine learning, especially in the concept to find the optimal correlation or classification of data.
- Minkowski distance is used in certain algorithms also like K-Nearest Neighbors, Self-Organizing Map (SOM), and K-Means Clustering.
- This measure has three requirements:
- **Zero Vector** — The zero vector has a length of zero whereas every other vector has a positive length.
    - For example, if we travel from one place to another, then that distance is always positive. However, if we travel from one place to itself, then that distance is zero.
- **Scalar Factor** — When you multiple the vector with a positive number its length is changed whilst keeping its direction.
    - For example, if we go a certain distance in one direction and add the same distance, the direction does not change.
- **Triangle Inequality** — The shortest distance between two points is a straight line.



- The formula for the Minkowski distance is shown below:

$$D(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- Most interestingly about this distance measure is the use of parameter p. We can use this parameter to manipulate the distance metrics to closely resemble others.

- Common values of p are:

- p=1 — Manhattan distance
- p=2 — Euclidean distance

- **Disadvantages:**
- Minkowski has the same disadvantages as the distance measures they represent, so a good understanding of metrics like Manhattan, Euclidean, and Chebyshev distance is extremely important.
- **Advantages:**
- Find the distance measure that works best for your use case.
- It allows you a huge amount of flexibility over your distance metric, which can be a huge benefit if you are closely familiar with p and many distance measures.

It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the “order” or “p“, that allows different distance measures to be calculated.

```
# calculating minkowski distance between vectors
from scipy.spatial import minkowski_distance
# define data
row1 = [10, 20, 15, 10, 5]
row2 = [12, 24, 18, 8, 7]
# calculate distance (p=1)
dist = minkowski_distance(row1, row2, 1)
print(dist)
# calculate distance (p=2)
dist = minkowski_distance(row1, row2, 2)
print(dist)
```

OUTPUT:

```
13.0
6.082762530298219
```

K-Means Clustering Algorithm

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning.
- It groups the *unlabeled dataset into different clusters*.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It allows us to *cluster the data into different groups* and a convenient way to discover the categories of groups in the unlabeled dataset on its own *without the need for any training*.

- The k-means clustering algorithm mainly performs two tasks:
  - Determines the best value for K center points or centroids by an iterative process.
  - Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-01:
Choose the number of clusters K.

Step-02:
Randomly select any K data points as cluster centers.Select cluster centers in such a way that they are as farther as possible from each other.

Step-03:
Calculate the distance between each data point and each cluster center.

The distance may be calculated either by using given distance function or by using Euclidean distance formula.K-Nearest Neighbor(KNN) Algorithm for Machine Learning

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on *Supervised Learning technique.*

K-NN algorithm assumes the s*imilarity between the new case/data and available cases* and put the new case into the category that is most similar to the available categories.

K-NN algorithm ***stores all the available data and classifies a new data point based on the similarity.*** This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

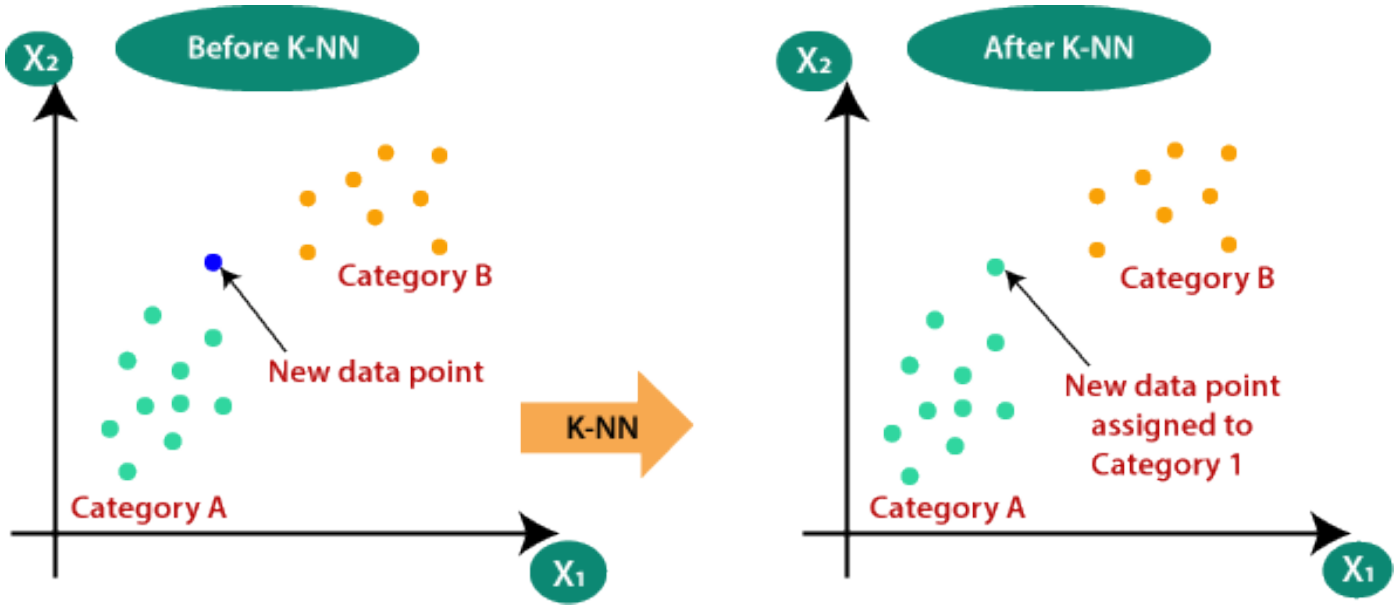
K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

•

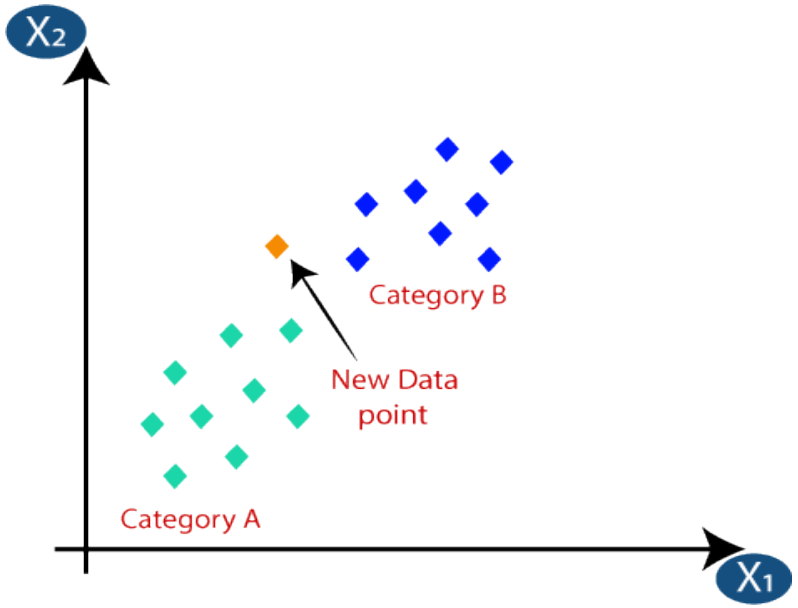
Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.

To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

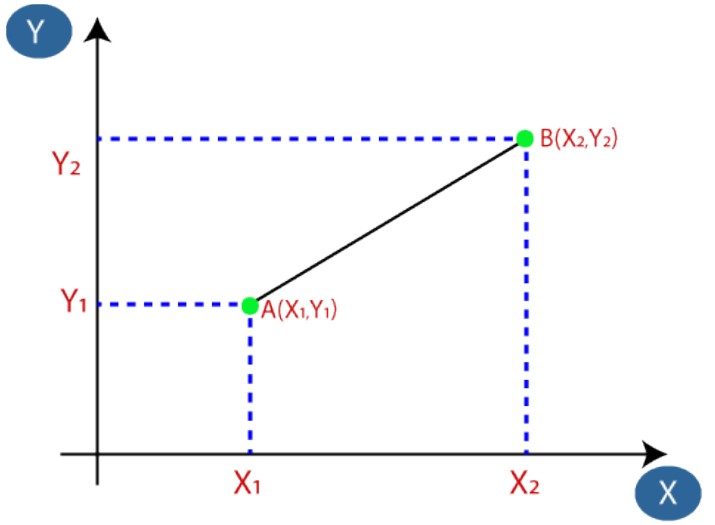


How does K-NN work?

- The K-NN working can be explained on the basis of the below algorithm:
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

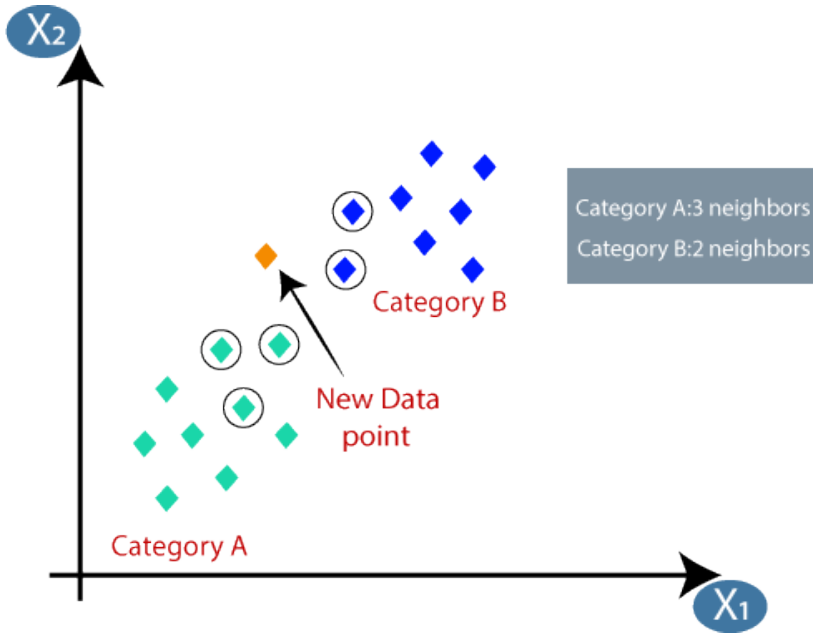


• Firstly, we will choose the number of neighbors, so we will choose the k=5.
• Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- Advantages of KNN Algorithm:
- It is **simple** to implement.
- It is ***robust to the noisy*** training data
- It can be more effective if the **training data is large**.
- Disadvantages of KNN Algorithm:
- Always needs to **determine the value of K** which may be complex some time.
- The ***computation cost is high*** because of calculating the distance between the data points for all the training samples.

Example

We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7. Without another expensive survey, can we guess what the classification of this new tissue is?

1. Determine parameter K = number of nearest neighbors

Suppose use K = 3

2. Calculate the distance between the query-instance and all the training samples

Coordinate of query instance is (3, 7), instead of calculating the distance we compute square distance which is faster to calculate (without square root)

3. *Sort the distance and determine nearest neighbors based on the K-th minimum distance*

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance (3, 7)	**Rank minimum distance**	**Is it included in 3-Nearest neighbors?**
7	7	$(7-3)^2+(7-7)^2=16$	**3**	**Yes**
7	4	$(7-3)^2+(4-7)^2=25$	**4**	**No**
3	4	$(3-3)^2+(4-7)^2=9$	**1**	**Yes**
1	4	$(1-3)^2+(4-7)^2=13$	**2**	**Yes**

4. Gather the category Y *of the nearest neighbors.* Notice in the second row last column that the category of nearest neighbor (Y) is not included because the rank of this data is more than 3 (=K).

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance (3, 7)	Rank minimum distance	Is it included in 3-Nearest neighbors?	**Y = Category of nearest Neighbor**
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes	**Bad**
7	4	$(7-3)^2+(4-7)^2=25$	4	No	**-**
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes	**Good**
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes	**Good**

5. *Use simple majority of the category of nearest neighbors as the prediction value of the query instance*

We have 2 good and 1 bad, since 2>1 then we conclude that a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7 is included in **Good** category.

k in k-Means

a. **The Elbow Method**
b. **The Silhouette Method**

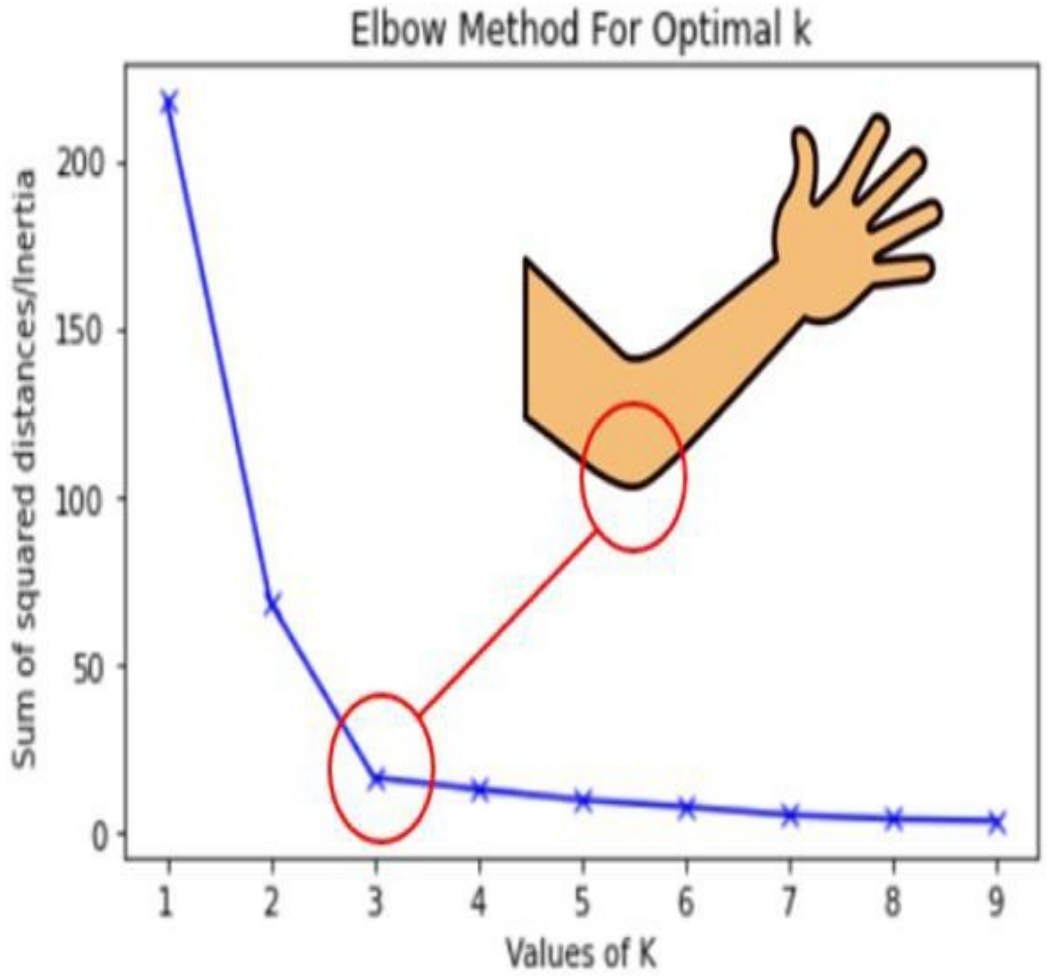
The Elbow Method:

*Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of k**, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an **elbow.***

- Perform K-means clustering with all these different values of K. For each of the K values, we calculate average distances to the centroid across all data points.
- Plot these points and find the point where the average distance from the centroid falls suddenly ("Elbow").
- At first, clusters will give a lot of information (about variance), but at some point, the marginal gain will drop, giving an angle in the graph.
  The number of clusters is chosen at this point, hence the "elbow criterion". This "elbow" can't always be unambiguously identified.
- **Inertia:** Sum of squared distances of samples to their closest cluster center.

Elbow Method

```
Sum_of_squared_distances = [] K = range(1,10)
 for num_clusters in K : kmeans =
KMeans(n_clusters=num_clusters) kmeans.fit(data_frame)
Sum_of_squared_distances.append(kmeans.   inertia_)
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
 plt.title('Elbow Method For Optimal k')
plt.show()
```



Line plot between K and inertia

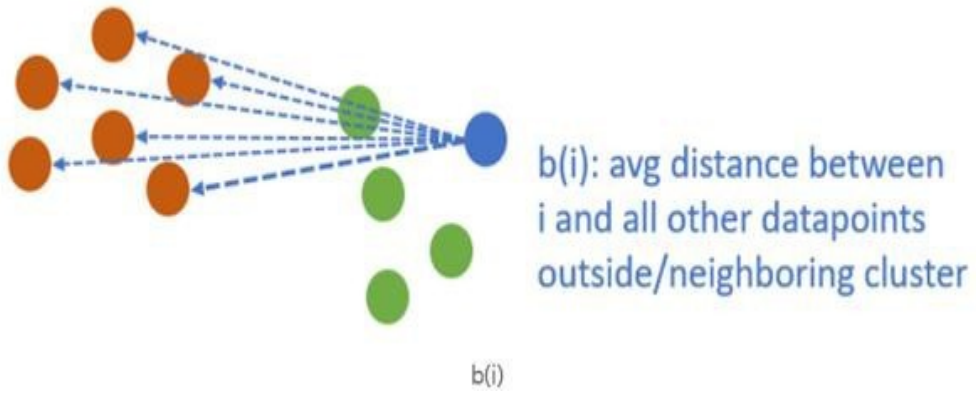
Silhouette Analysis

- The silhouette coefficient or v in k-means clustering measures the similarity of a data point within its cluster (cohesion) compared to other clusters (separation).
- The equation for calculating the silhouette coefficient for a particular data

  point:

- S(i) is the silhouette coefficient of the data p $S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$
- a(i) is the average distance between i and all the other data points in the cluster to which i belongs.
- b(i) is the average distance from i to all clusters to which i does not belong.

Silhouette Analysis



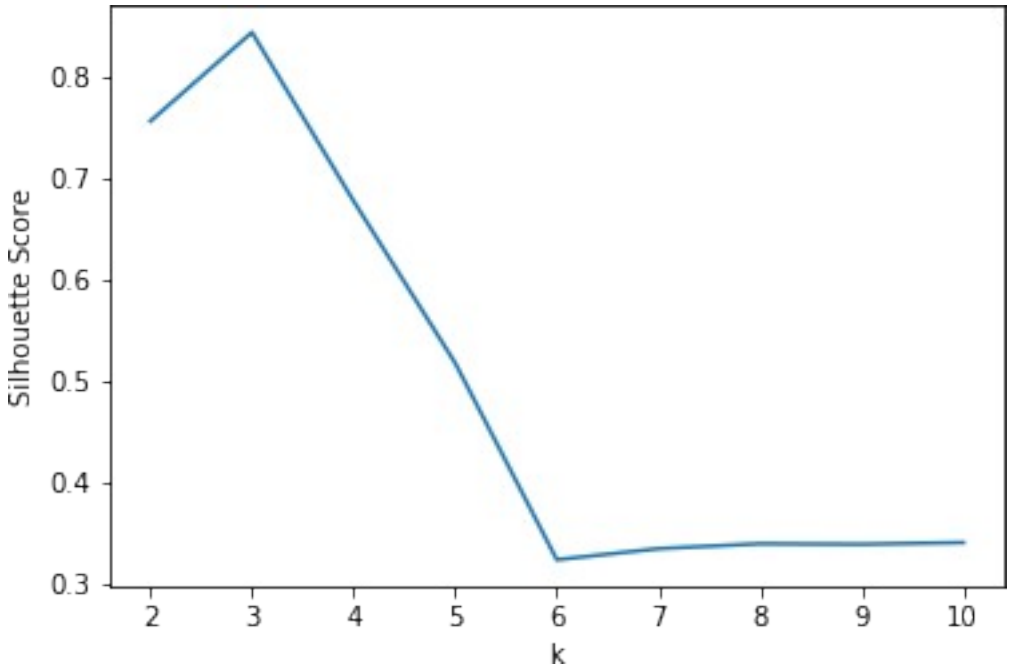
We will then calculate the average_silhouette for every k. Then plot the graph between average_silhouette and K.

- The value of the silhouette analysis coefficient is between [-1, 1].
- A score of 1 denotes the best, meaning that the data point i is very compact within the cluster to which it belongs and far away from the other clusters.
- The worst value is -1. Values near 0 denote overlapping clusters.

Silhouette Analysis

- *from sklearn.metrics import silhouette_score*
- *sil = []*
- *kmax = 10*
- *# dissimilarity would not be defined for a single cluster, thus, minimum number of clusters should be 2*
- *for k in range(2, kmax+1):*
- *kmeans = KMeans(n_clusters = k).fit(x)*
- *labels = kmeans.labels_*
- *sil.append(silhouette_score(x, labels, metric = 'euclidean'))*

- There is a clear peak at k = 3. Hence, it is optimal.

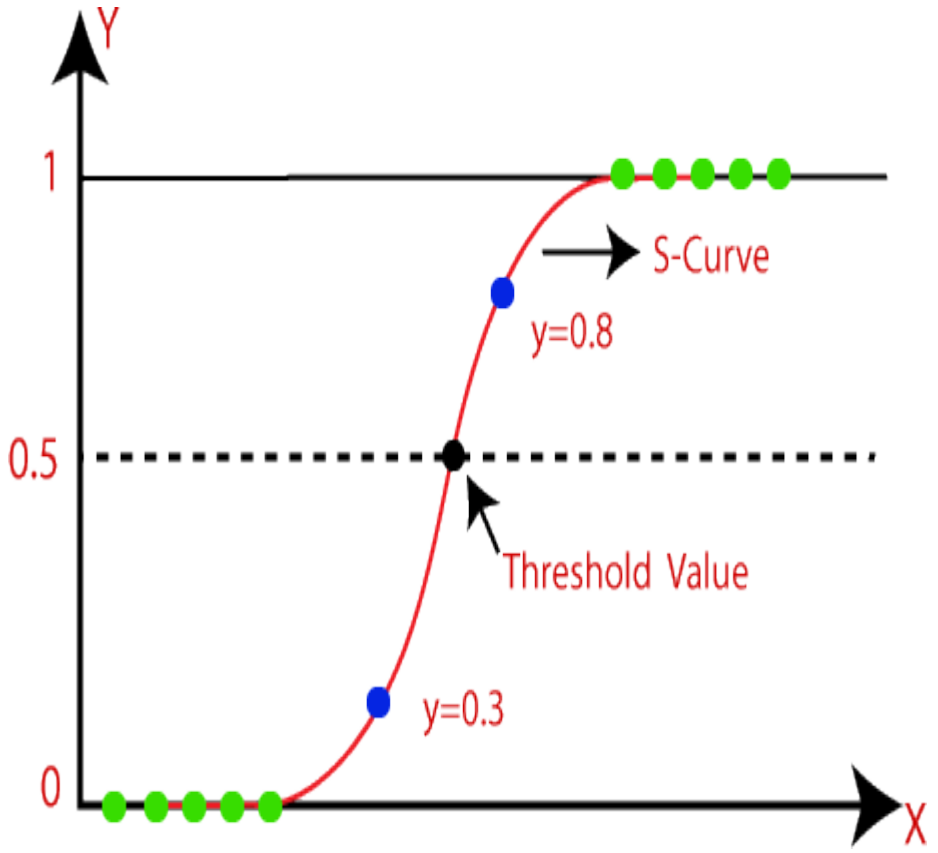


Logistic Regression

- Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary).
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- Eg:cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.



Logistic Regression Equation:

Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Types of Logistic Regression

[1] **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

[2] **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”

[3] **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

Assumptions of Logistic Regression

1. **Independent observations**: Each observation is independent of the other. meaning there is no correlation between any input variables.
2. **Binary dependent variables**: It takes the assumption that the dependent variable must be binary or dichotomous, meaning it can take only two values. For more than two categories SoftMax functions are used.
3. **Linearity relationship** between independent variables and log odds: The relationship between the independent variables and the log odds of the dependent variable should be linear.
4. **No outliers**: There should be no outliers in the dataset.
5. **Large sample size**: The sample size is sufficiently large

Example

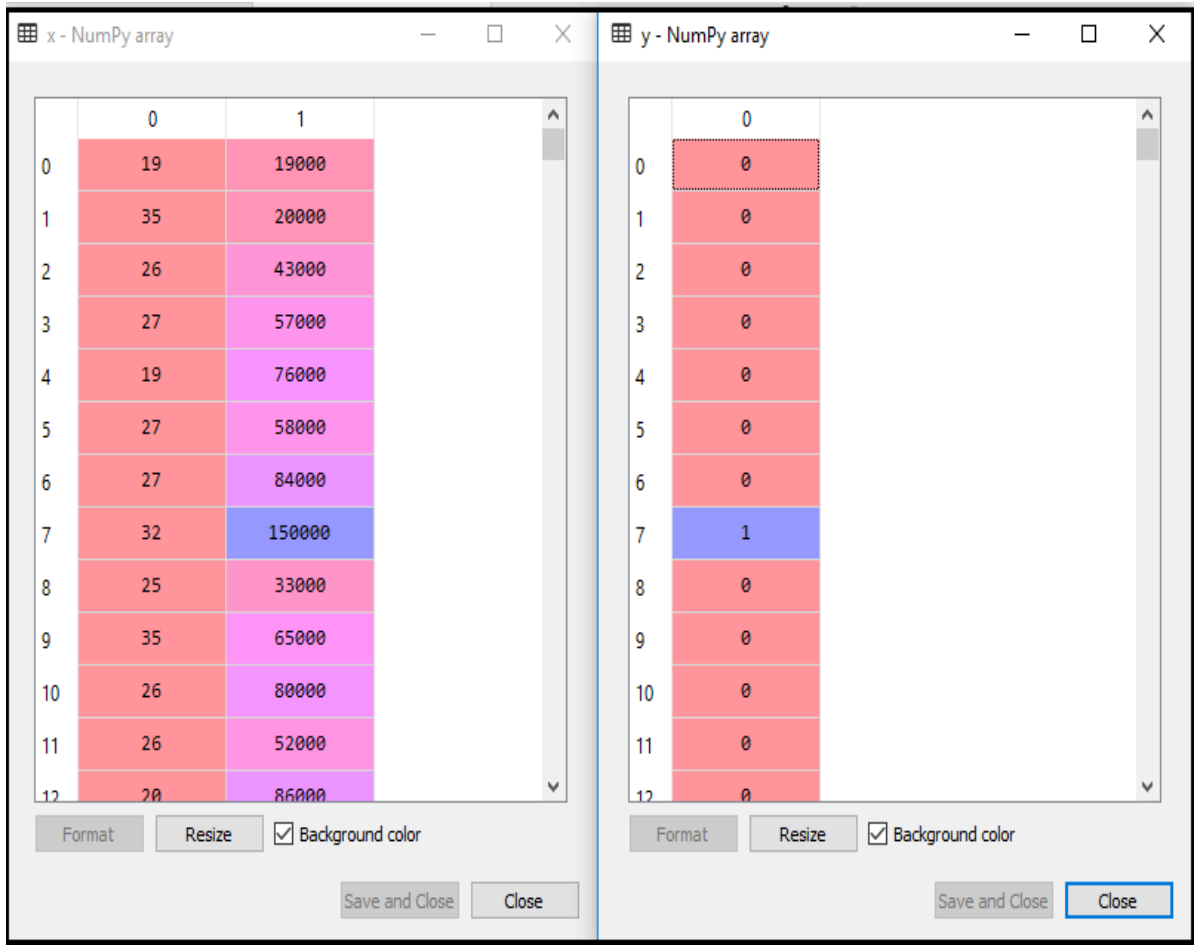
- There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So the company wanted to check how many users from the dataset, wants to purchase the car. The dataset is shown in the below image. In this problem, we will predict the **purchased variable (Dependent Variable)** by using **age and salary (Independent variables)**.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

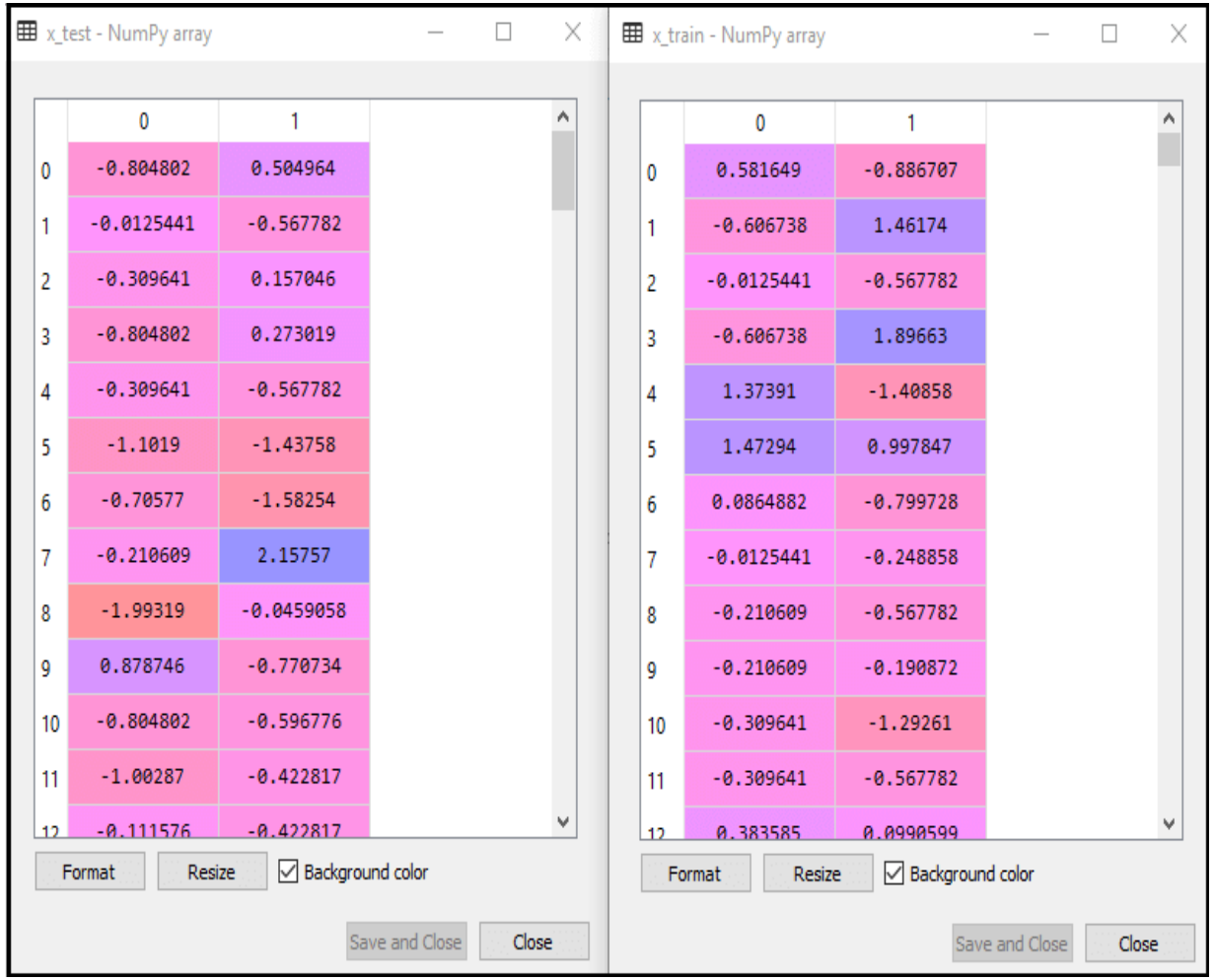
Example

#Data Pre-procesing Step # importing libraries ***import*** *numpy as nm*
import *matplotlib.pyplot as mtp* ***import*** *pandas as pd*
#importing datasets
data_set= pd.read_csv('user_data.csv')

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



```
# Splitting the dataset into training and t est set.
from sklearn.model_selection import tra
in_test_split
x_train, x_test, y_train, y_test= train_te st_split(x, y, test_size= 0.25, random_sta te=0
#feature Scaling
from sklearn.preprocessing import Stand ardScaler
st_x= StandardScaler()
x_train=          st_x.fit_transform(x_train)          x_test=
st_x.transform(x_test)
```



Example

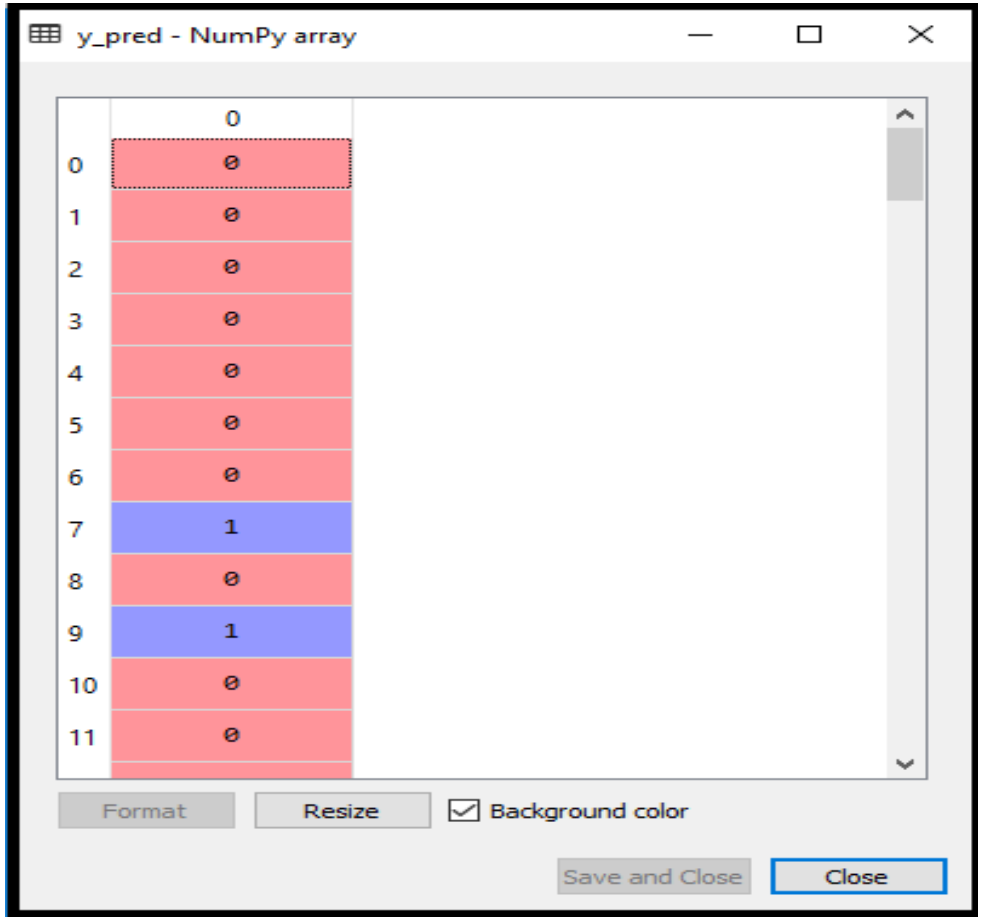
#Fitting Logistic Regression to the training set from sklearn.linear_model

import *LogisticRegress*
ion

classifier= LogisticRegression(random_state=0) classifier.fit(x_train, y_train) LogisticRegression(C=1.0, class_weight=None)

#Predicting the test set result

y_pred= classifier.predict(x_test)



Multiclass Classification with Logistic Regression

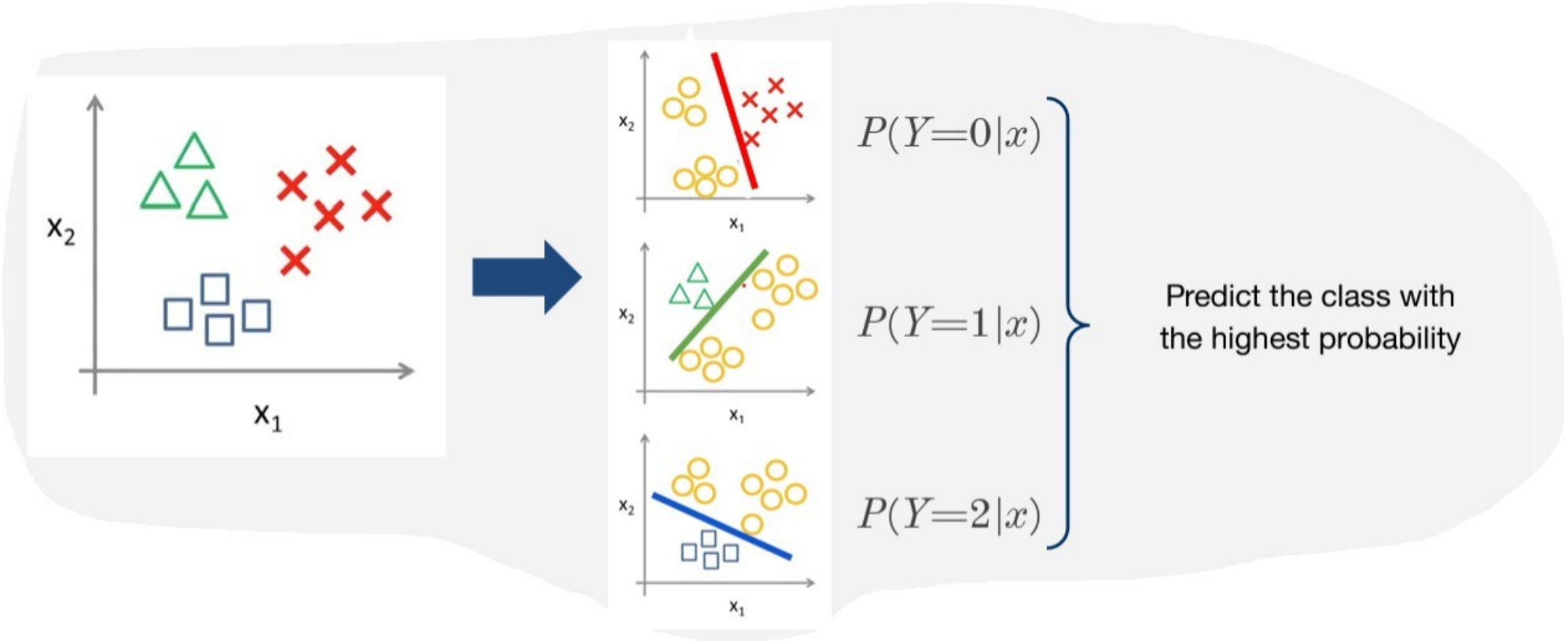
Multiclass logistic regression is a machine learning technique that uses logistic regression to predict multiple outcomes. It's also known as multinomial logistic regression or softmax regression. •

Common Approaches

- 1. One-vs-Rest (One-vs-All)
- 2. Softmax Regression (Multinomial Logistic Regression)

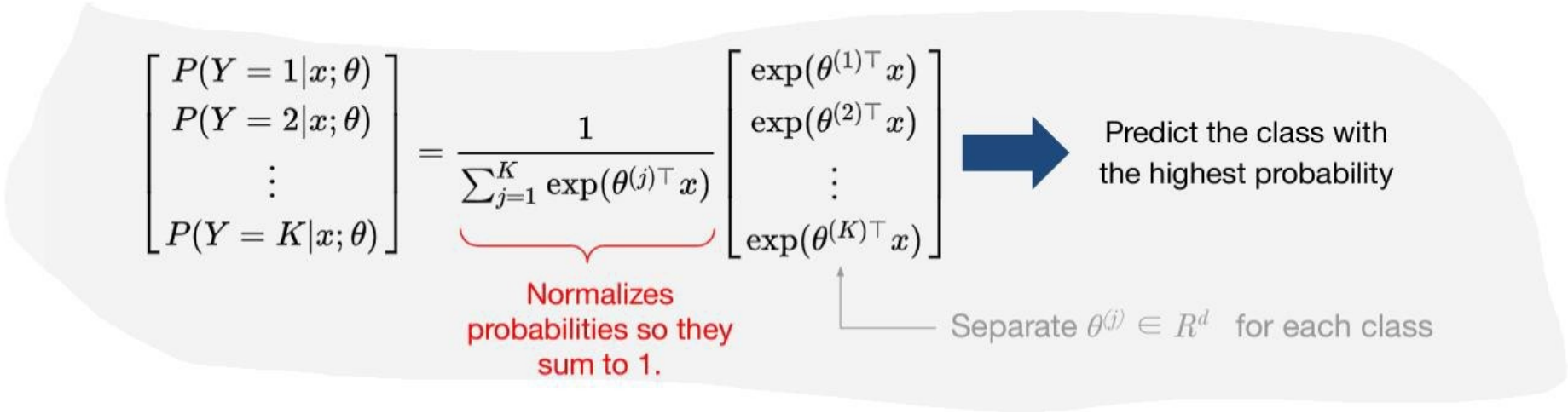
One-vs-Rest (One-vs-All)

- For each class, build a logistic regression to find the probability the observation belongs to that class. For each data point, predict the class with the highest probability.



Softmax Regression (Multinomial Logistic Regression)

- A multinomial logistic regression (or multinomial regression for short) is used when the outcome variable being predicted is nominal and has more than two categories that do not have a given rank or order. This model can be used with any number of independent variables that are categorical or continuous.



- Given a test input x , we want our hypothesis to estimate the probability that P(y=k|x) for each value of k=1,…,K . I.e., we want to estimate the probability of the class label taking on each of the K different possible values. Thus, our hypothesis will output a K - dimensional vector (whose elements sum to 1) giving us our K estimated probabilities. Concretely, our hypothesis hθ(x) takes the form:

Normalizes the input so that it sums to one

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

Code

- *LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None, penalty='l2', random_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)*

- There is an argument called 'multi_class' which is 'auto' by default. Here 'auto' does automatic selection of binary and multinomial. If the data is binary classification 'auto' does binary classification, and if the data is multi classification, 'auto' does multi classification.
- We can also change 'auto' to 'ovr'. Here 'ovr' does only binary classificaion.

Reference

- http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegres sion/

Module 4

Probabilistic Based Models

- **Naive Bayes: Introduction to Naive Bayes Classifier-Bayes' Theorem and Conditional Probability- Gaussian,Multinomial, and Bernoulli Naive Bayes. Bayesian Belief Network-EM algorithm.**

Probabilistic Model

- **A Probabilistic model in machine learning is a mathematical representation of a real-world process that incorporates uncertain or random variables.**
- **These are the models that incorporate random variables and probability distributions.**
- **Random variables represent the potential outcomes of an uncertain event**
- **Probability distribution assign probabilities to the various potential outcomes.**
- **Probabilistic models are used in practice because realistic decision making often necessitates recognizing uncertainty.**

What is probabilistic model?????

- **Let's consider a classification problem with N classes. If the classification model (classifier) is probabilistic, for a given input, it will provide probabilities for each class (of the N classes) as the output. In other words, a probabilistic classifier will provide a probability distribution over the N classes. Usually, the class with the highest probability is then selected as the Class for which the input data instance belongs.**

Probabilistic ModelVs Non probabilistic

-An example….

- **Take the task of classifying an image of an animal into five classes —**

  {Dog, Cat, Deer, Lion, Rabbit} as the problem. As input, we have an image (of a dog). For this example, let's consider that the classifier works well and provides correct/ acceptable results for the particular input we are discussing. When the image is provided as the input to the probabilistic classifier, it will provide an output such as (Dog (0.6), Cat (0.2), Deer(0.1), Lion(0.04), Rabbit(0.06)). But, if the classifier is non-probabilistic, it will only output "Dog".

Examples

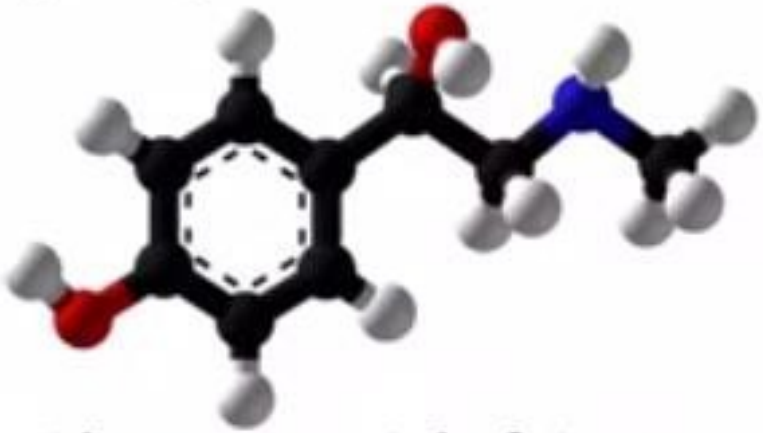
Oil prices

If you run an energy intensive business, an airline for example, then the price of oil is a key determinant of your profitability

For medium or long-term investment planning (buying new planes) the future price of oil is an important consideration

But who knows the price of oil in ten years? No-one. But we may be able to put a probability distribution around the future price and incorporate the uncertainty into the decision making process

Valuing a drug development company



- A company has 10 drugs in a development portfolio
- Given a drug has been approved, you have predicted its revenue
- But whether a drug is approved or not is an uncertain future event (a random variable). You have estimated the probability of approval
- You only wish to invest in the company if the company's expected total revenue for the portfolio is over $10B in 5 years time
- You need to calculate the ***probability distribution*** of the total revenue to understand the investment risk

Categories Of Probabilistic Models

These models can be classified into the following categories:

- **Generative models**
- **Discriminative models.**
- **Graphical models**
- **Generative models aim to model the joint distribution of the input and output variables. These models generate new data based on the probability distribution of the original dataset.**
- **The discriminative model aims to model the conditional distribution of the output variable given the input variable. They learn a decision boundary that separates the different classes of the output variable.**
- **Graphical models use graphical representations to show the conditional dependence between variables. They are commonly used for tasks such as image recognition, natural language processing, and causal inference.**

Advantages

- **Ability to take into account uncertainty and variability in data. This allows for more accurate predictions and decision-making, particularly in complex and unpredictable situations.**
- **Provide insights into how different factors influence outcomes and can help identify patterns and relationships within data.**

Naïve Bayes Classifier

- **A Naive Bayes classifier is a probabilistic machine learning model that is used for classification task**
- **It is based on Baye's Theorem**

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. The assumption made here is that the predictors/features are independent.

Baye's Theorem

- **Bayes theorem works on the principle of Conditional Probability.**
- **The general statement of Bayes' theorem is "The conditional probability of an event A, given the occurrence of another event B, is equal to the product of the event of B, given A and the probability of A divided by the probability of event B.**

- **where,**
- **P(A) and P(B) are the probabilities of events A and B**
- **P(A|B) is the probability of event A when event B happens**
- **P(B|A) is the probability of event B when A happens**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes Theorem Derivation

The proof of Bayes' Theorem is given as, according to the conditional probability formula,

$$P(E_i|A) = P(E_i \cap A) / P(A)\ldots..(i)$$

Then, by using the multiplication rule of probability, we get

$$P(E_i \cap A) = P(E_i)P(A|E_i)\ldots\ldots(ii)$$

Now, by the total probability theorem,

$$P(A) = \sum P(E_k)P(A|E_k)\ldots..(iii)$$

Substituting the value of $P(E_i \cap A)$ and P(A) from eq (ii) and eq(iii) in eq(i) we get,

$$P(E_i|A) = P(E_i)P(A|E_i) / \sum P(E_k)P(A|E_k)$$

- **Bayes' theorem is also known as the formula for the probability**

  of "causes". As we know, the E_i's are a partition of the
  sample space S, and at any given time only one of the events E_i occurs. Thus we conclude that the Bayes' theorem formula gives the probability of a particular E_i, given the event A has occurred.

Terms Related to Bayes Theorem

- **Hypotheses: Events happening in the sample space E_1, E_2,… E_n is called the hypotheses**
- **Priori Probability: Priori Probability is the initial probability of an event occurring before any new data is taken into account. $P(E_i)$ is the priori probability of hypothesis E_i.**
- **Posterior Probability: Posterior Probability is the updated probability of an event after considering new information. Probability $P(E_i|A)$ is considered as the posterior probability of hypothesis E_i.**

- ***Conditional Probability***
- **The probability of an event A based on the occurrence of another event B is termed conditional Probability.**
- **It is denoted as P(A|B) and represents the probability of A when event B has already happened**
- ***Joint Probability***
- **When the probability of two more events occurring together and at the same time is measured it is marked as Joint Probability. For two events A and B, it is denoted by joint probability is denoted as, $P(A \cap B)$.**
- ***Random Variables***
- **Real-valued variables whose possible values are determined by random experiments are called random variables. The probability of finding such variables is the experimental probability.**

Difference Between Conditional Probability and Bayes Theorem

Bayes' Theorem	Conditional Probability
Bayes' Theorem is derived using the definition of conditional probability. It is used to find the reverse probability.	Conditional Probability is the probability of event A when event B has already occurred.
Formula: $P(A\|B) = [P(B\|A)P(A)] / P(B)$	Formula: $P(A\|B) = P(A \cap B) / P(B)$

Bayes Theorem Examples

- **Example 1: A person has undertaken a job. The probabilities of completion of the job on time with and without rain are 0.44 and 0.95 respectively. If the probability that it will rain is 0.45, then determine the probability that the job will be completed on time.**
- **Solution:**

Let E_1 be the event that the mining job will be completed on time and E_2 be the event that it rains. We have, P(A) = 0.45,

P(no rain) = P(B) = 1 − P(A) = 1 − 0.45 = 0.55

By multiplication law of probability,

$P(E_1)$ = 0.44, and $P(E_2)$ = 0.95

Since, events A and B form partitions of the sample space S, by total probability theorem, we have P(E) = P(A) $P(E_1)$ +

P(B) $P(E_2)$

⇒ P(E) = 0.45 × 0.44 + 0.55 × 0.95

⇒ P(E) = 0.198 + 0.5225 = 0.7205

So, the probability that the job will be completed on time is 0.7205

- **Example 2: There are three urns containing 3 white and 2 black balls; 2 white and 3 black balls; 1 black and 4 white balls respectively. There is an equal probability of each urn being chosen. One ball is equal probability chosen at random. what is the probability that a white ball is drawn?**
- **Let E_1, E_2, and E_3 be the events of choosing the first, second, and third urn respectively. Then,**
- **$P(E_1) = P(E_2) = P(E_3) = 1/3$**
- **Let E be the event that a white ball is drawn. Then,**
- **$P(E/E_1) = 3/5$, $P(E/E_2) = 2/5$, $P(E/E_3) = 4/5$**
- **By theorem of total probability, we have**
- **$P(E) = P(E/E_1) \cdot P(E_1) + P(E/E_2) \cdot P(E_2) + P(E/E_3) \cdot P(E_3)$**
- **$\Rightarrow P(E) = (3/5 \times 1/3) + (2/5 \times 1/3) + (4/5 \times 1/3)$**
- **$\Rightarrow P(E) = 9/15 = 3/5$**

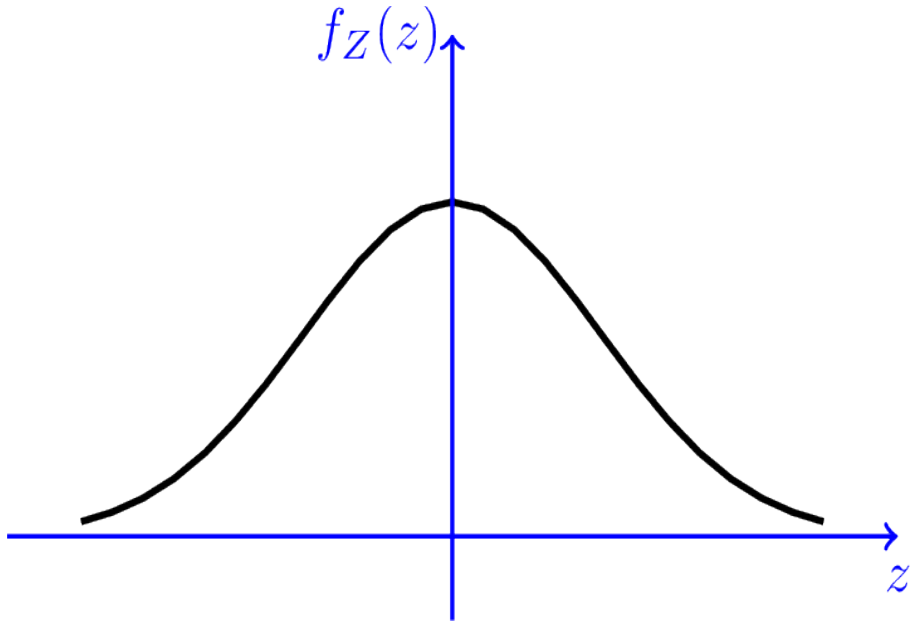
- **How is Bayes' theorem different from conditional probability?**
- **Bayes' theorem is used to define the probability of an event based on the previous conditions of the event. Whereas, Bayes' Theorem uses conditional probability to find the reverse probability of the event.**

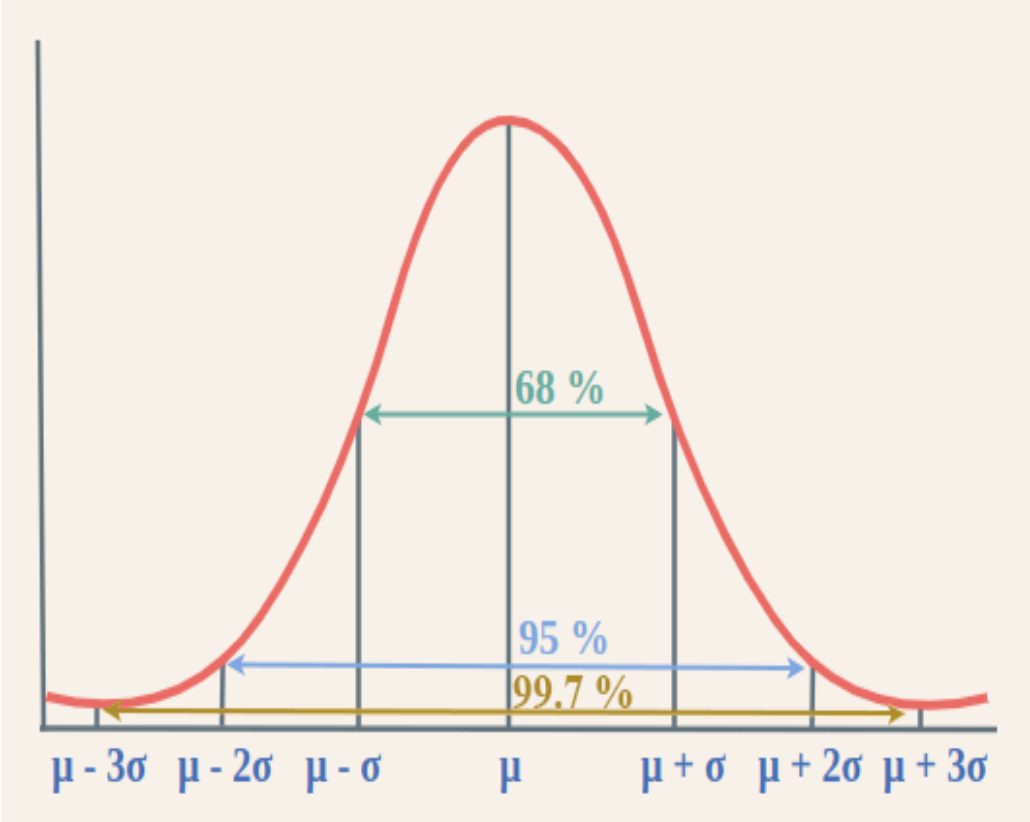
Gaussian Distribution

- **The Gaussian distribution, also known as the normal distribution, is a fundamental concept in statistics and probability theory. It describes how the values of a variable are distributed.**
- **The formula for the Gaussian distribution is:**

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- **where:**
- **x is a value in the distribution.**
- **μ is the mean.**
- **σ is the standard deviation.**
- **Exp denotes the exponential function.**





- **The standard deviations are used to subdivide the area under the normal curve. Each subdivided section defines the percentage of data, which falls into the specific region of a graph.**
- **Analysis : A smaller standard deviation results in a narrower and taller bell curve, indicating that data points are clustered closely around the mean. Conversely, a larger standard deviation leads to a wider and shorter bell curve, suggesting that data points are more spread out from the mean.**
- **The Empirical Rule, also known as the 68-95-99.7 rule, quantifies the proportion of data falling within certain intervals around the mean in a normal distribution. It provides a quick way to estimate the spread of data without performing detailed calculations.**

Multinomial Distribution

- **The multinomial distribution is a generalization of the binomial distribution. While the binomial distribution deals with scenarios where there are only two possible outcomes (success or failure), the multinomial distribution handles situations where there are more than tw**
- **The multinomial distribution applies to experiments in which the following conditions are true:**
- **Repeated: The experiment consists of repeated trials, such as rolling a die five times instead of just once.**
- **Independent: Each trial must be independent of the others. For example, if you roll two dice, the outcome of one die does not impact the outcome of the other die.**
- **Same probability: The probability of each outcome must be the same across each instance of the experiment. For example, if a fair, six-sided die is used, then there must be a one-in-six chance of each number being given on each roll.**
- **Specific outcome: Each trial must produce a specific outcome, such as a number between two and 12 if rolling two six-sided dice.**

Multinomial Naive Bayes Classifier

- **This algorithm is particularly effective for text classification tasks such as spam detection or sentiment analysis.**
- **Multinomial Naive Bayes classifier assumes that the features (e.g., words in text data) follow a multinomial distribution given the class label. In this case, the probability of observing a particular word in a document is modeled using a multinomial distribution**
- **During training, the model estimates the probabilities of each feature (word) occurring in each class, and these probabilities are used to predict the class of new documents.**
- **For a given class c, the probability of a document with features x1,x2,x3…xn s given by:**

$$P(c \mid x_1, x_2, \ldots, x_n) \propto P(c) \prod_{i=1}^{n} P(x_i \mid c)$$

where $P(x_i \mid c)$ follows a multinomial distribution.

Step-by-Step with BNB

- **Data Preparation: Begin with a set of binary data. Each row signifies a data sample while columns represent features.**

DATA PREPARATION

Feature 1	Feature 2	Class
1	0	0
0	1	0
1	1	0
0	0	0
1	0	1
0	1	1
0	0	1
1	1	1
1	0	0
0	1	0

- Each row in the table represents a data sample
- each column represents a binary feature.
- For instance, "Feature 1" represents the presence (1) or absence (0)

Training Phase

- **Training Phase — Class Priors: Compute the prior probabilities of each class based on your training data.**
- **Calculate the prior probabilities for each class based on the training data:**
- **Total samples = 10**
- **Class 0 samples = 6**
- **Class 1 samples = 4**
- **Thus:**
- **P (Class 0) = 6/10 = 0.6**
- **P (Class 1) = 4/10 = 0.4**

TRAINING PHASE – CLASS PRIORS

Calculate the **prior probabilities** of each class (Class A and Class B) based on the training data

$$P(\text{Class 0}) = \frac{\text{Number of Class 0 samples}}{\text{Total number of samples}} = \frac{6}{10} = 0.6$$

$$P(\text{Class 1}) = \frac{\text{Number of Class 1 samples}}{\text{Total number of samples}} = \frac{4}{10} = 0.4$$

- **Training Phase — Feature Priors: Calculate conditional probabilities for each feature based on their presence (1) or absence (0).**
- **Next, compute the conditional probabilities for each feature based on its presence (1) or absence (0):**
- **For Feature 1:**
-

With Class 0: 3 out of 6 samples have Feature 1 = 1

P(Feature 1=1|Class 0) = 3 /6 =0.5

2. With Class 1: 3 out of 6 samples have Feature 1 = 1

P (Feature 1=1|Class 1)= 2/4 =0.5

TRAINING PHASE - FEATURE PRIORS

- For each feature (Feature 1 and Feature 2), calculate conditional probabilities for each class based on the presence (1) or absence (0) of the feature
- For feature 1

$$P(\text{Feature 1} = 1 | \text{Class 0}) = \frac{\text{Number of Class 0 samples with Feature 1} = 1}{\text{Total number of Class 0 samples}} = \frac{3}{6}$$

$$P(\text{Feature 1} = 1 | \text{Class 1}) = \frac{\text{Number of Class 1 samples with Feature 1} = 1}{\text{Total number of Class 1 samples}} = \frac{2}{4}$$

- **For Feature 2:**

With Class 0: 3 out of 6 samples have Feature 2= 1

P(Feature 2=1|Class 0) = 3 /6 =0.5

2. With Class 1: 1out of 4samples have Feature 2= 1

P (Feature 2=1|Class 1)= 1/4 =0.25

TRAINING PHASE - FEATURE PRIORS

- For feature 2

$$P(\text{Feature 2 = 1}|\text{Class 0}) = \frac{\text{Number of Class 0 samples with Feature 2 = 1}}{\text{Total number of Class 0 samples}} = \frac{3}{6} = 0.5$$

$$P(\text{Feature 2 = 1}|\text{Class 1}) = \frac{\text{Number of Class 1 samples with Feature 2 = 1}}{\text{Total number of Class 1 samples}} = \frac{2}{4} = 0.5$$

- **4. Prediction: Using the trained model, classify new samples.**
- **Calculate the likelihood of each class for the given features and compute the unnormalized posterior probability.**
- **For Class 0:**
  P(Feature 1=1, Feature 2=0 | Class 0) = P(Feature 1=1 | Class 0) × P(Feature2=0 | Class 0)

  = 0.5 × 0.5

  = 0.25
- **Unnormalized Posterior: P(Class 0) × Likelihood = 0.6 × 0.25 = 0.15**

CALCULATE POSTERIORS

- **Class 0**

$$P(Class|Features) = \frac{P(Features|Class) * P(Class)}{P(Features)}$$

- 1. Calculate the likelihood

P(Feature 1=1 / Class 0) × *P(Feature 2=0 / Class 0)* = 0.5 × (1 − 0.5) = 0.25

- 2. Calculate the unnormalized posterior probability

P(Class 0) × *likelihood* = 0.6 × 0.25 = 0.15

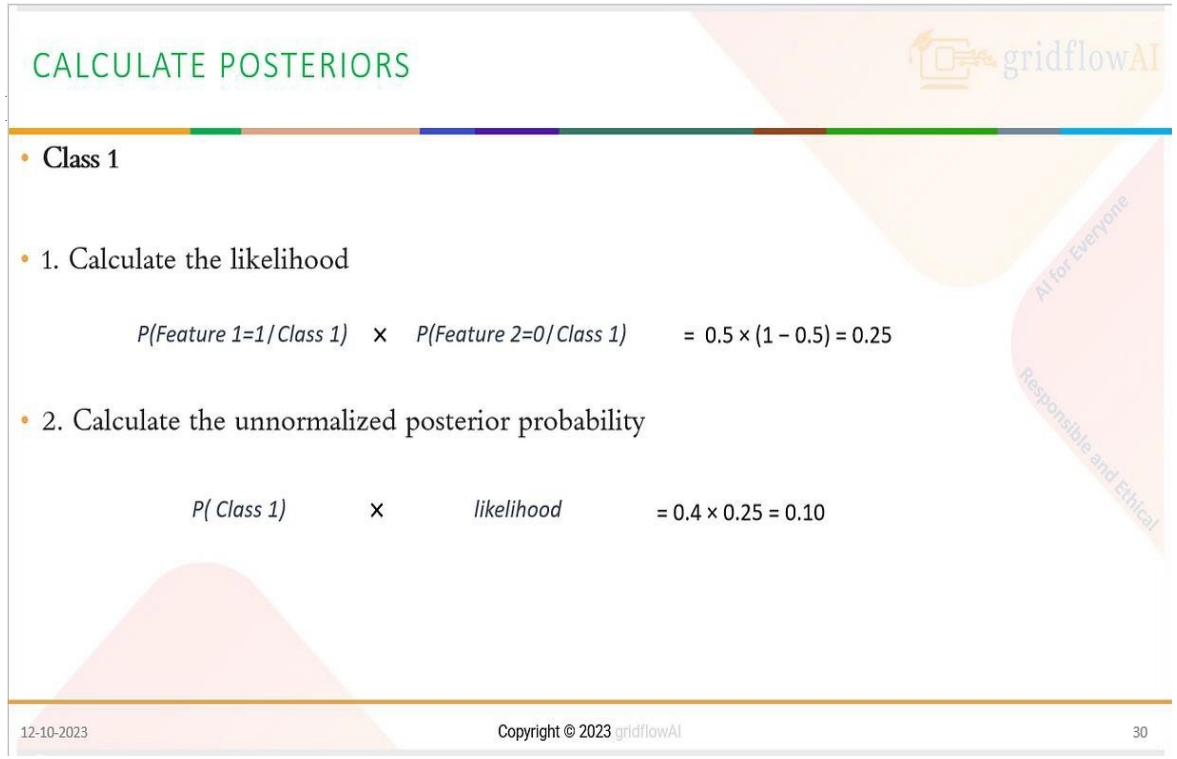
- **For Class 1:**
  P (Feature 1=1, Feature 2=0 | Class 1) = P (Feature 1=1 | Class 1) × P (Feature 2=0 | Class 1)

  = 0.5 × 0.75

  = 0.375

- **Unnormalized Posterior: P (Class 1)**

  × Likelihood = 0.4 × 0.375 = 0.15

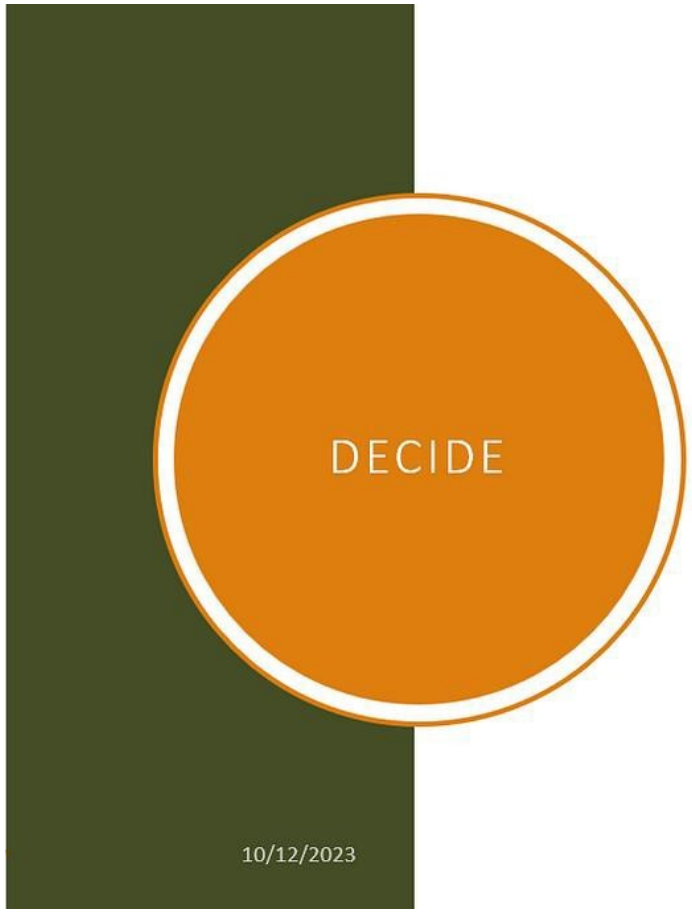


- **Normalize these probabilities**
- **Z = 0.15 (Class 0) + 0.15 (Class 1) = 0.3**
- **P (Class 0|Features) = 0.25 / 0.3 = 0.8**
- **P (Class 1|Features) = 0.15 / 0.3 = 0.2**

NORMALIZE THE POSTERIORS

- Calculate the normalization factor (sum of unnormalized posteriors):
  - Z = 0.15 + 0.10 = 0.25
- Normalize the posteriors:
  - P(Class 0|Features) = 0.15 / 0.25= 0.6
  - P(Class 1|Features) = 0.10 / 0.25 = 0.4

Making the prediction



- Since
  - P(Class 0|Features) =0.6 > P(Class 1|Features) = 0.4
    - the prediction is Class 0.

Bayesian Belief Network

- **A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."**
- **It is also called a Bayes network, belief network, decision network, or Bayesian model.**
- **Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.**

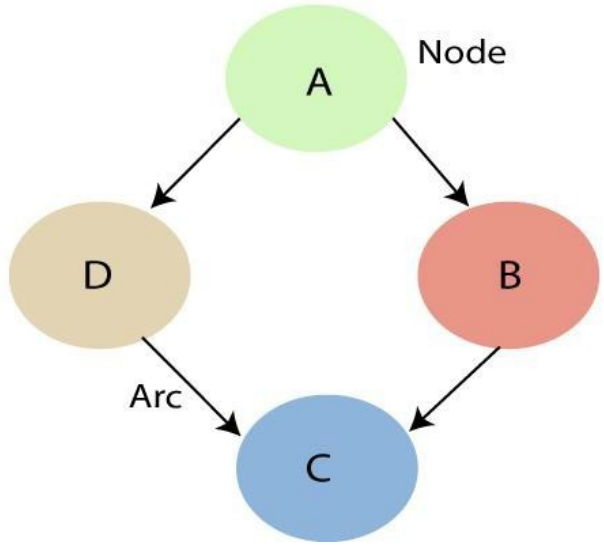
- **A Bayesian network graph is made up of nodes and Arcs (directed links), where:**

Each node corresponds to the random variables, and a variable can be continuous

or discrete.

Arc or directed arrows represent the causal relationship or conditional probabilities betw links or arrows connect the pair of nodes in the graph.

These links represent that one node directly influence the other node, and if there is n independent with each other.



- **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
- **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
- **Node C is independent of node A.**
- **The Bayesian network has mainly two components:**
- **Causal Component**
- **Actual numbers**
- **Each node in the Bayesian network has condition probability distribution $P(X_i \mid Parent(X_i))$, which determines the effect of the parent on that node.**

- **The Bayesian network has mainly two components:**
- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution $P(X_i \mid Parent(X_i))$, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

- **Joint probability distribution:**
- **If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.**
- **$P[x_1, x_2, x_3,....., x_n]$, it can be written as the following way in terms of the joint probability distribution.**
- **$= P[x_1| x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$**
- **$= P[x_1| x_2, x_3,....., x_n]P[x_2|x_3,....., x_n]$.....$P[x_{n-1}|x_n]P[x_n]$.**
- **In general for each variable Xi, we can write the equation as:**

$P(X_i|X_{i-1},....., X_1) = P(X_i |Parents(X_i))$

- Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.
- Problem:
- Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.

The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.

The conditional distributions for each node are given as conditional probabilities table or CPT.

Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.

In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

Burglary (B)

Earthquake(E)

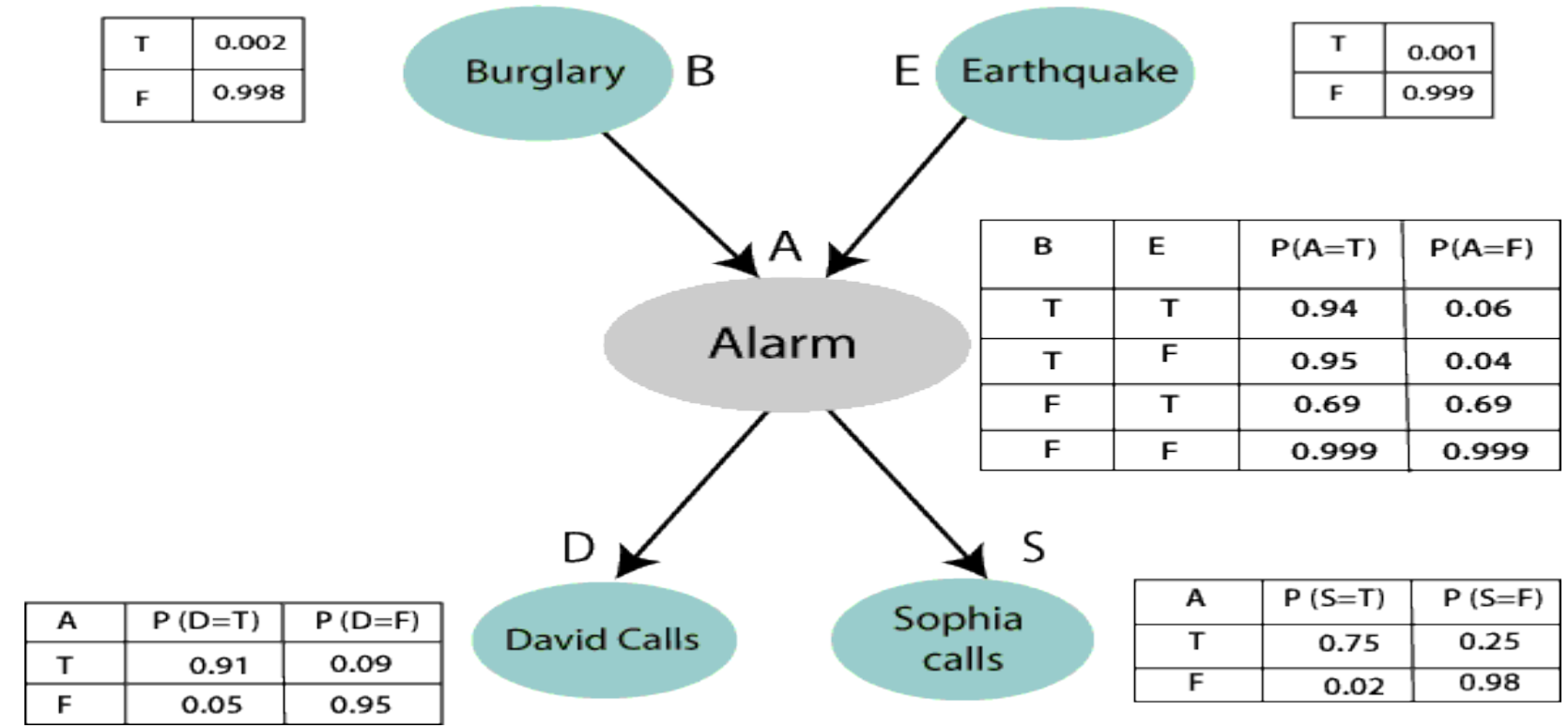
Alarm(A)

David Calls(D)

Sophia calls(S)

- **We can write the events of problem statement in the form of probability: P[D, S, A, B, E], can rewrite the above probability statement using joint probability distribution:**
- **P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]**
- **=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]**
- **= P [D| A]. P [S| A, B, E]. P[A, B, E]**
- **= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]**
- **= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]**
-

Example



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

- **Let's take the observed probability for the Burglary and earthquake component:**
- **P(B= True) = 0.002, which is the probability of burglary.**
- **P(B= False)= 0.998, which is the probability of no burglary.**
- **P(E= True)= 0.001, which is the probability of a minor earthquake**
- **P(E= False)= 0.999, Which is the probability that an earthquake not occurred.**

- **We can provide the conditional probabilities as per the below tables:**
- **Conditional probability table for Alarm A:**
- **The Conditional probability of Alarm A depends on Burglar and earthquake:**

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

- **Conditional probability table for David Calls:The Conditional probability of David that he will call depends on the probability of Alarm.**

- **Conditional probability table for Sophia**

  Calls:

- **The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm.“**

A	P(S= True)	P(S= False)
True	0.75	0.25

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

From the formula of joint distribution, we can write the problemstatement in the form of probability distribution:

- **P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).**
- **= 0.75* 0.91* 0.001* 0.998*0.999**
- **= 0.00068045.**

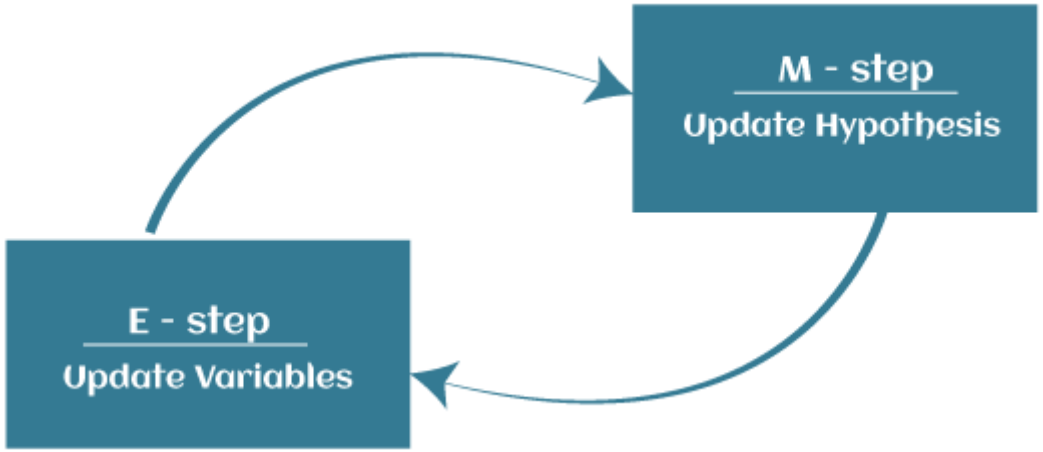
Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

EM Algorithm

- **The Expectation-Maximization (EM) algorithm is defined as the combination of various unsupervised machine learning algorithms, which is used to determine the local maximum likelihood estimates (MLE) or maximum a posteriori estimates (MAP) for unobservable variables in statistical models.**
- **It is a technique to find maximum likelihood estimation when the latent variables are present. It is also referred to as the latent variable model.**
- **A latent variable model consists of both observable and unobservable variables where observable can be predicted while unobserved are inferred from the observed variable. These unobservable variables are known as latent variables.**

- **The EM algorithm is the combination of various unsupervised ML algorithms, such as the k-means clustering algorithm. Being an iterative approach, it consists of two modes. In the first mode, we estimate the missing or latent variables. Hence it is referred to as the Expectation/estimation step (E-step). Further, the other mode is used to optimize the parameters of the models so that it can explain the data more clearly. The second mode is known as**

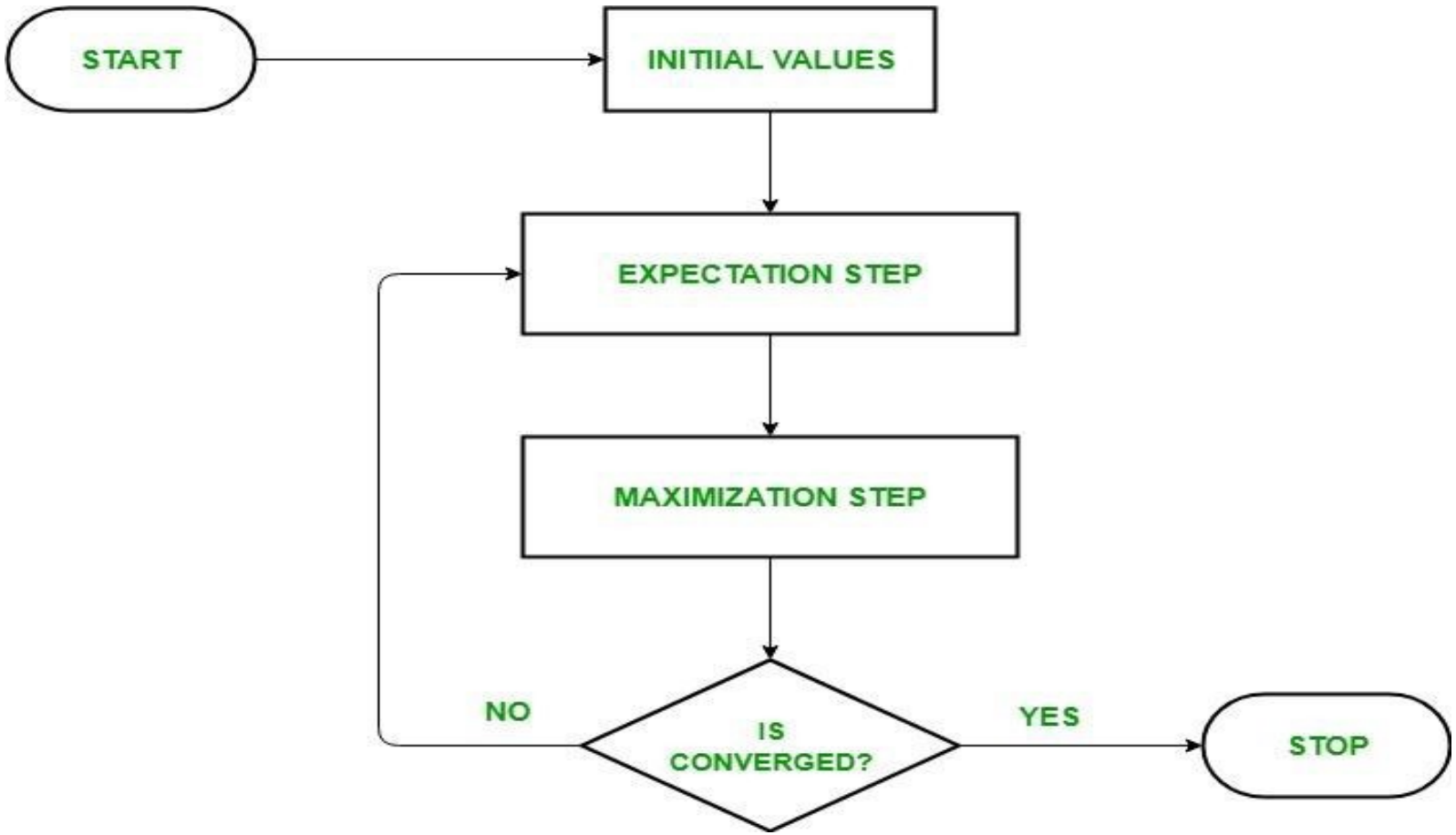
  the maximization-step or M-step.



- **Expectation step (E - step): It involves the estimation (guess) of all missing values in the dataset so that after completing this step, there should not be any missing value.**
- **Maximization step (M - step): This step involves the use of estimated data in the E-step and updating the parameters.**
- **Repeat E-step and M-step until the convergence of the values occurs.**

Convergence in the EM algorithm

- **Convergence refers to the condition when the EM algorithm has reached a stable solution. It is typically determined by checking if the change in the log-likelihood or the parameter estimates falls below a predefined threshold.**
- **The essence of the Expectation-Maximization algorithm is to use the available observed data of the dataset to estimate the missing data and then use that data to update the values of the parameters.**


START
INITIAL VALUES
EXPECTATION STEP
MAXIMIZATION STEP
IS CONVERGED?
NO
YES
STOP

- **Initialization: A set of initial values of the parameters are considered. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.**

2.E-Step (Expectation Step): we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables.

- **Compute the posterior probability or responsibility of each latent variable given the observed data and current parameter estimates.**
- **Estimate the missing or incomplete data values using the current parameter estimates.**
- **Compute the log-likelihood of the observed data based on the current parameter estimates and estimated missing data.**

3.M-step (Maximization Step): In this step, we use the complete data generated in the preceding "Expectation" – step in order to update the values of the parameters. It is basically used to update the hypothesis.

3. **Update the parameters of the model by maximizing the expected complete data log-likelihood obtained from the E-step.**
4. **This typically involves solving optimization problems to find the parameter values that maximize the log-likelihood.**
5. **The specific optimization technique used depends on the nature of the problem and the model being used.**

-

4.Convergence: In this step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat step-2 and step-3 i.e. “Expectation” – step and “Maximization” –

step until the convergence occurs.

4. **Check for convergence by comparing the change in log-likelihood or the parameter values between iterations.**
5. **If the change is below a predefined threshold, stop and consider the algorithm converged.**
6. **Otherwise, go back to the E-step and repeat the process until convergence is achieved.**

Code

```
import numpy as np import
seaborn as sns
from scipy.stats import norm
from scipy.stats import gaussian_kde import
matplotlib.pyplot as plt
# Generate a dataset with two Gaussian components
mu1, sigma1 = 2, 1
mu2, sigma2 = -1, 0.8
X1 = np.random.normal(mu1, sigma1, size=200) X2 =
np.random.normal(mu2, sigma2, size=600) X =
np.concatenate([X1, X2])
```

```
# Plot the density estimation using seaborn
sns.kdeplot(X) plt.xlabel('X')
plt.ylabel('Density')
plt.title('Density Estimation of X') plt.show()
```

- `# Initialize parameters`

```
mu1_hat, sigma1_hat = np.mean(X1), np.std(X1) mu2_hat,
sigma2_hat = np.mean(X2), np.std(X2) pi1_hat, pi2_hat =
len(X1) / len(X), len(X2) / len(X)
```

```
# Perform EM algorithm for 20 epochs num_epochs = 20
log_likelihoods = []
for epoch in range(num_epochs):
    # E-step: Compute responsibilities
    gamma1 = pi1_hat * norm.pdf(X, mu1_hat, sigma1_hat)
    gamma2 = pi2_hat * norm.pdf(X, mu2_hat,
sigma2_hat)
    total = gamma1 + gamma2 gamma1 /= total
    gamma2 /= total
```

```
# M-step: Update parameters
  mu1_hat = np.sum(gamma1 * X) /
  np.sum(gamma1) mu2_hat = np.sum(gamma2 *
  X) / np.sum(gamma2)
  sigma1_hat = np.sqrt(np.sum(gamma1 * (X - mu1_hat)**2) /
np.sum(gamma1))
  sigma2_hat = np.sqrt(np.sum(gamma2 * (X - mu2_hat)**2) /
np.sum(gamma2))
  pi1_hat =
  np.mean(gamma1) pi2_hat
  = np.mean(gamma2) #
  Compute log-likelihood
  log_likelihood = np.sum(np.log(pi1_hat * norm.pdf(X, mu1_hat,
sigma1_hat)
                + pi2_hat * norm.pdf(X, mu2_hat, sigma2_hat)))
  log_likelihoods.append(log_likelihood)
```

```
# Plot log-likelihood values over epochs
plt.plot(range(1, num_epochs+1), log_likelihoods)
plt.xlabel('Epoch') plt.ylabel('Log-Likelihood')
plt.title('Log-Likelihood vs.Epoch') plt.show()
```

NEW HORIZON
COLLEGE OF ENGINEERING

Module 5

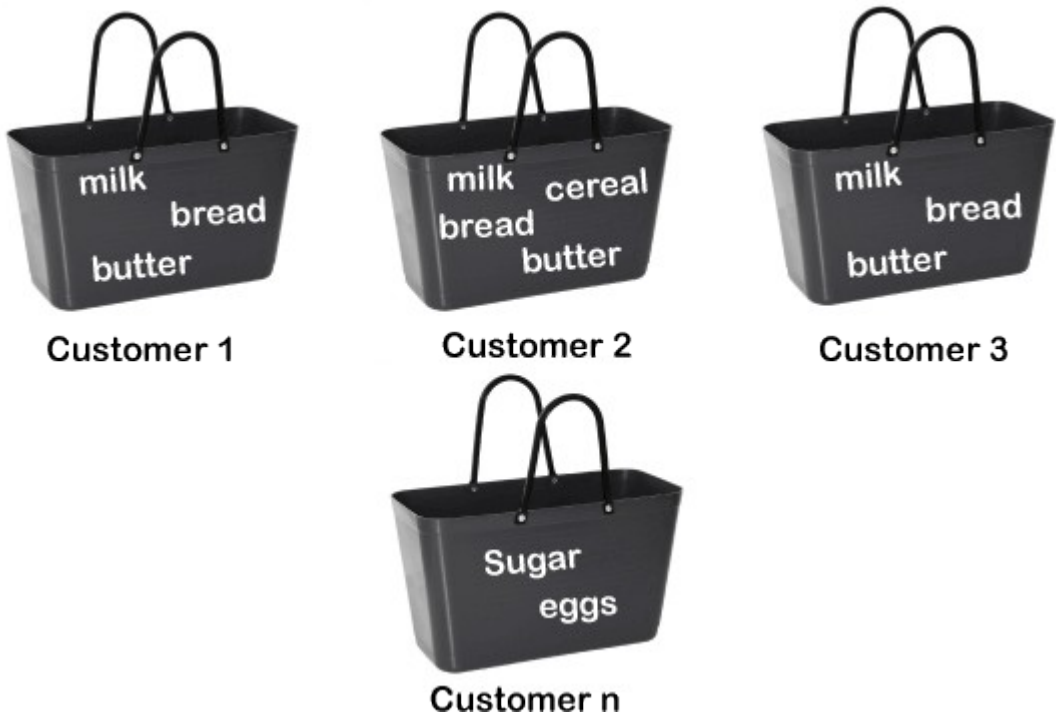
Unsupervised Algorithms

- Introduction to Rule Based learning and Association rules- Apriori Algorithm, FP-Growth. Introduction to unsupervised Algorithms- types of clustering algorithms- k-means clustering algorithms-DBSCAN clustering algorithm.

Introduction to Rule Based learning and Association rules

- Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable.
- It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.
- Association rule mining finds interesting associations and relationships among large sets of data items.
- This rule shows how frequently a itemset occurs in a transaction. A typical example is a ***Market Based Analysis.***

- For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby.



Association Rule Learning

- Association rule learning works on the concept of If and Else Statement, such as if A then B.

Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known as single cardinality.

Association Rule Learning

To measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- ***Support***
- ***Confidence***
- ***Lift***

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as

$$\text{Supp}(X) = \frac{Freq(X)}{T}$$

- Confidence
- Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

- Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

- It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:
- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

- Association rule learning can be divided into three types of algorithms:

1.Apriori

2.Eclat

3.F-P Growth Algorithm

- **The Apriori Algorithm: Basics**

- The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

-

- Key Concepts :

- Frequent Itemsets: The sets of item which has minimumsupport (denoted by Li for ith-Itemset).
- Apriori Property: Any subset of frequent itemset must befrequent.
- Join Operation: To find Lk , a set of candidate k-itemsetsis generated by joining Lk-1 with itself.

The Apriori Algorithm in aNutshell

- Find the *frequent itemsets*: the sets of items that haveminimum support
  - **A subset of a frequent itemset must also be afrequent itemset**
    - i.e., if $\{AB\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$should be a frequent itemset
  - **Iteratively find frequent itemsets with cardinalityfrom 1 to *k* (*k*-itemset)**
- Use the frequent itemsets to generate association rules

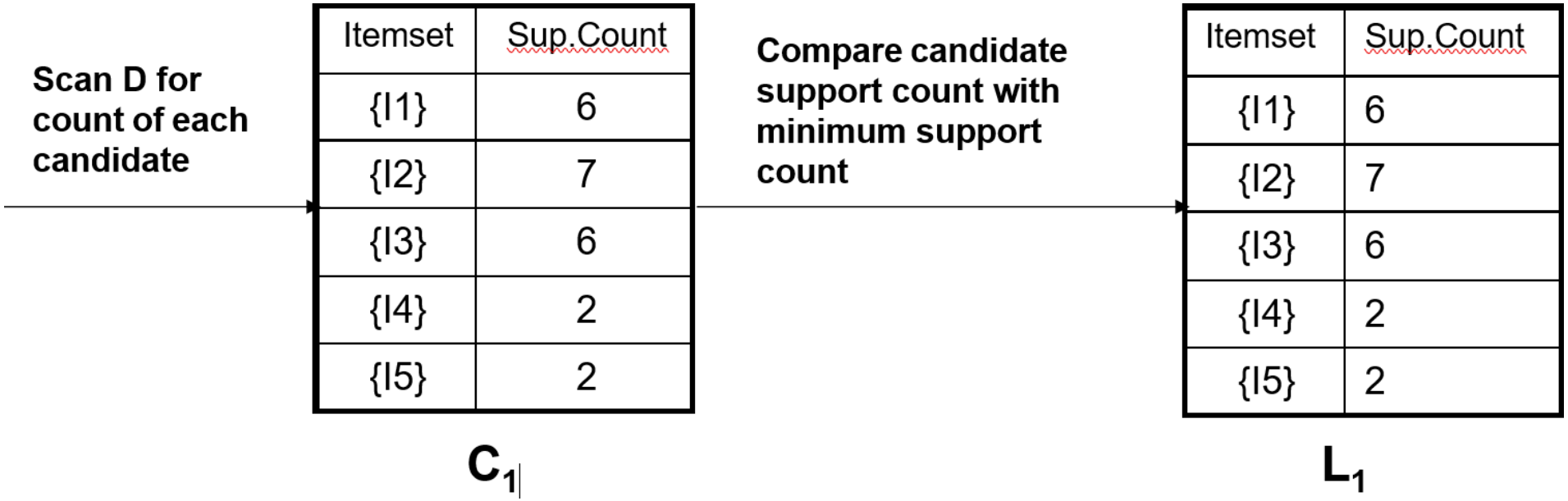
The Apriori Algorithm : Pseudocode

- Join Step: Ck is generated by joining Lk-1with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be asubset of a frequent k-itemset
- Pseudo-code:
    - *Ck*: Candidate itemset of size k
    - *Lk* : frequent itemset of size k
    - L_1 = {frequent items};
    - **for** (*k* = 1; *Lk* !=Æ; *k*++) **do begin**
        - *Ck+1* = candidates generated from *Lk*;
        - **for each** transaction *t* in database do
            - increment the count of all candidates in *Ck+1*
            - that are contained in *t*
        - *Lk+1* = candidates in *Ck+1* with min_support
        - **end return** È*k* *Lk*;

The Apriori Algorithm: Example

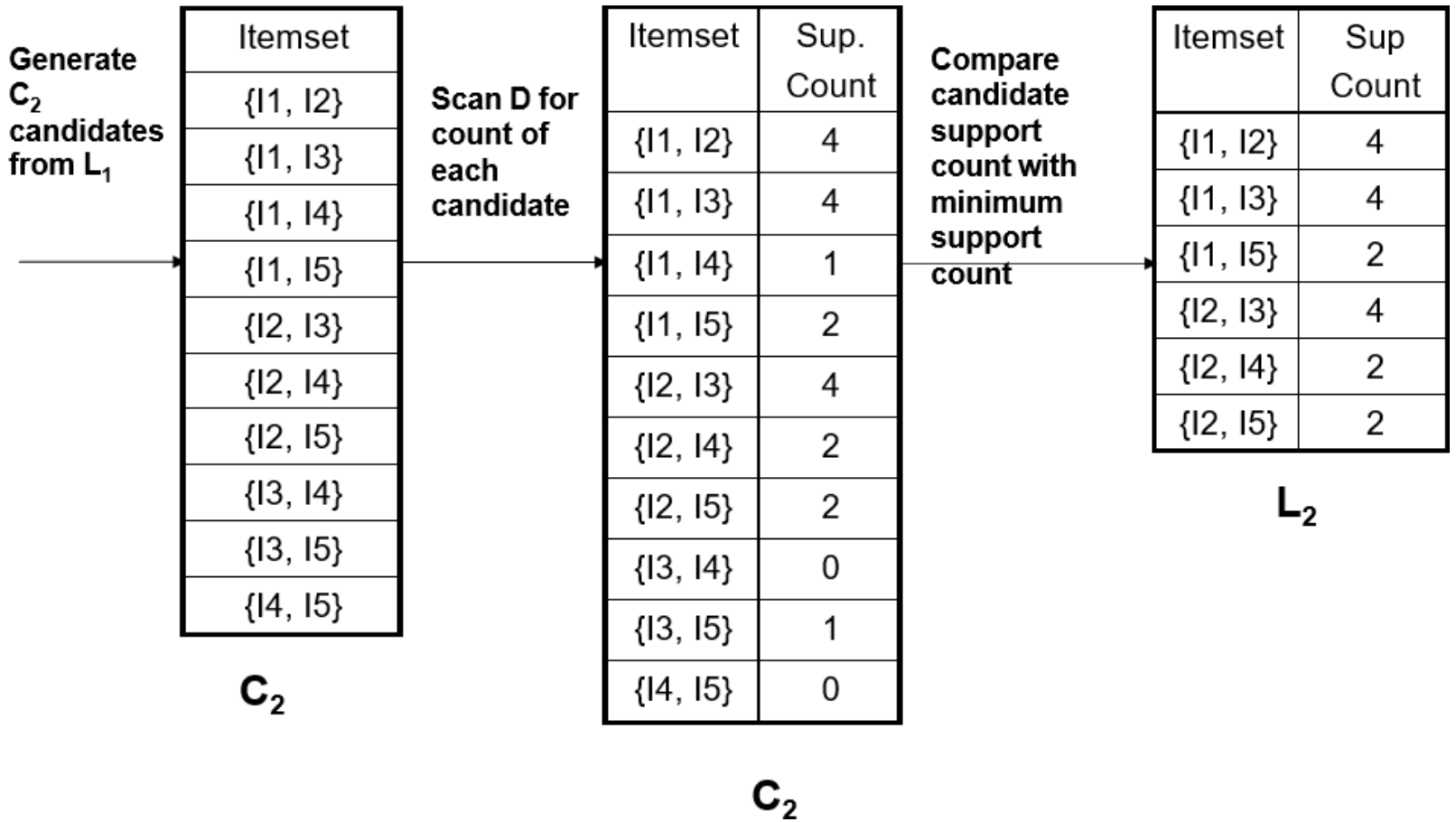
- Consider a database, D , consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3



- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the setof candidate.

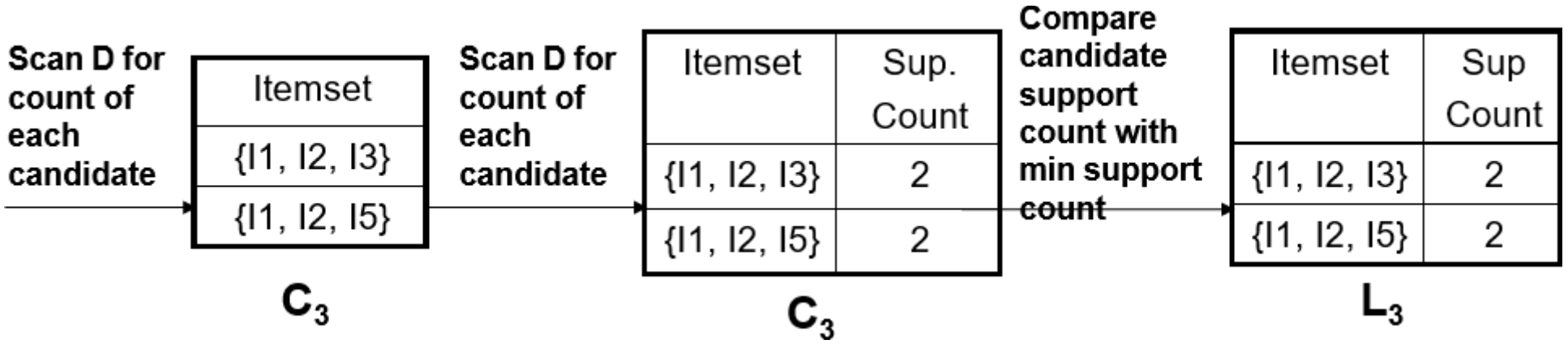
Step 2: Generating 2-itemset Frequent Pattern



Step 2: Generating 2-itemset Frequent Pattern

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses L_1 *Join* L_1 to generate a candidate set of2-itemsets, C_2.
- Next, the transactions in D are scanned and the supportcount for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined,consisting of those candidate 2- itemsets in C_2 having minimum support.
- Note: We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern



- The generation of the set of candidate 3-itemsets, C_3 , involves use ofthe Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3,I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 3: Generating 3-itemset Frequent Pattern

- Based on the Apriori property that all subsets of a frequent itemset mustalso be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example , lets take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, Wewill keep {I1, I2, I3} in C_3.
- Lets take another example of {I2, I3, I5} which shows how the pruning isperformed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L_2 and hence it is not frequent violatingApriori Property. Thus We will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members ofresult of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L_3, consistingof those candidates 3-itemsets in C_3 having minimum support.

Step 4: Generating 4-itemset Frequent

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4- itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}}is not frequent.
- Thus, $C_4 = \varphi$, and algorithm terminates, having foundall of the frequent items. This completes our Apriori Algorithm.
- What's Next ?
  - These frequent itemsets will be used to generate strong association rules (where strong association rules satisfyboth minimum support & minimum confidence).

Step 5: Generating Association Rules from FrequentItemsets

- Procedure:
  - For each frequent itemset ***"l"***, generate all nonempty subsetsof ***l***.
  - For every nonempty subset ***s*** of ***l***, output the rule **"s ⟍ (l-s)"** if
    - **support_count(l) / support_count(s) >= min_conf** wheremin_conf is minimum confidence threshold.
-

- Back To Example:
  - We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3},
  - {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

– Lets take ***l*** = {I1,I2,I5}.

– Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules fromFrequent

- **Let minimum confidence threshold is , say 70%.**
- The resulting association rules are shown below,each listed with its confidence.
  - – R1: I1 ^ I2 ⌊ I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - – R2: I1 ^ I5 ⌊ I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - – R3: I2 ^ I5 ⌊ I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

Step 5: Generating Association Rules fromFrequent Itemsets

- **R4: I1 ⌊ I2 ^ I5**
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- – **R5: I2 ⌊ I1 ^ I5**
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- – **R6: I5 ⌊ I1 ^ I2**
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.

•In this way, We have found three strong association rules.

Methods to Improve Apriori's

- Hash-based itemset counting: A *k*-itemset whose correspondinghashing bucket count is below the threshold cannot be frequent.
- Transaction reduction: A transaction that does not contain anyfrequent k-itemset is useless in subsequent scans.
- Partitioning: Any itemset that is potentially frequent in DB must befrequent in at least one of the partitions of DB.
- Sampling: mining on a subset of given data, lower support threshold
  - + a method to determine the completeness.
- Dynamic itemset counting: add new candidate itemsets only whenall of their subsets are estimated to be frequent.

Mining Frequent Patterns Without CandidateGeneration

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure

– highly condensed, but complete for frequent patternmining
– avoid costly database scans

- Develop an efficient, FP-tree-based frequent patternmining method

– **A divide-and-conquer methodology: decomposemining tasks into smaller ones**
– Avoid candidate generation: sub-database testonly!

FP-Growth Method : An

Consider the same previous example of a database, D , consisting of 9 transactions.

–Suppose min. up = 2/9 = 22 %)

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

–The first scan of database is same as Apriori, which derives the set of 1- itemsets & their support counts.

–The set of frequent items is sorted in the order of descending support count.

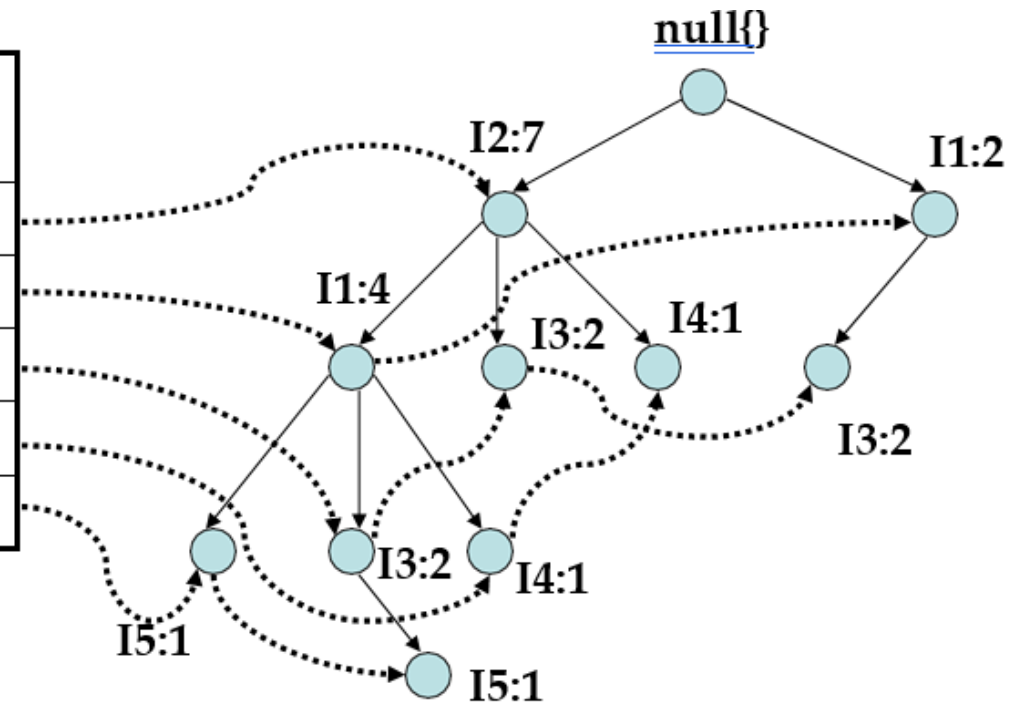
–The resulting set is denoted as L = {I2:7, I1:6, I3:6, I4:2, I5:2}

FP-Growth Method: Construction of FP-

- First, create the root of the tree, labeled with “null”.
- Scan the database D a second time. (First time we scanned it tocreate 1-itemset and then L).
- The items in each transaction are processed in L order (i.e. sortedorder).
- A branch is created for each transaction with items having theirsupport count separated by colon.
- Whenever the same node is encountered in another transaction, wejust increment the support count of the common node or Prefix.
- To facilitate tree traversal, an item header table is built so that eachitem points to its occurrences in the tree via a chain of node-links.
- Now, The problem of mining frequent patterns in database istransformed to that of mining the FP-Tree.

FP-Growth Method: Construction of FP-Tree

Item Id	Sup Count	Node-link
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



An FP-Tree that registers compressed, frequent pattern information

Mining the FP-Tree by Creating Conditional (sub)pattern bases

- Steps:

1. Start from each frequent length-1 pattern (as an initial suffixpattern).
2. Construct its conditional pattern base which consists of theset of prefix paths in the FP-Tree co-occurring with suffix pattern.
3. Then, Construct its conditional FP-Tree & perform miningon such a tree.
4. The pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from aconditional FP-Tree.

- The union of all frequent patterns (generated by step 4)gives the required frequent itemset

FP-Tree Example Continued

- Now, Following the above mentioned steps:
  - Lets start from I5. The I5 is involved in 2 branches namely {I2 I1 I5: 1} and {I2I1 I3 I5: 1}.
  - Therefore considering I5 as suffix, its 2 corresponding prefix paths would be
    - {I2 I1: 1} and {I2 I1 I3: 1}, which forms its conditional pattern base.

Item	Conditional patternbase	Conditional FP-Tree	Frequent pattern generated
I5	{(I2 I1: 1),(I2 I1 I3: 1)}	<I2:2 , I1:2>	I2 I5:2, I1 I5:2, I2 I1 I5: 2
I4	{(I2 I1: 1),(I2: 1)}	<I2: 2>	I2 I4: 2
I3	{(I2 I1: 1),(I2: 2), (I1: 2)}	<I2: 4, I1: 2>,<I 1:2>	I2 I3:4, I1, I3: 2 , I2 I1 I3:2
I2	{(I2: 4)}	<I2: 4>	I2 I1: 4

Mining the FP-Tree by creating conditional (sub) pattern bases

FP-Tree Example Continued

- Out of these, Only I1 & I2 is selected in the conditional FP-Tree because I3 is not satisfying the minimum support count.
  - For I1 , support count in conditional pattern base = 1 + 1 = 2 For I2 , support count in conditional pattern base = 1 + 1 = 2 For I3, support count in conditional pattern base = 1
  - Thus support count for I3 is less than required min_sup which is 2here.
- Now , We have conditional FP-Tree with us.
- All frequent pattern corresponding to suffix I5 are generated by considering all possible combinations of I5 and conditional FP-Tree.
- The same procedure is applied to suffixes I4, I3 and I1.
- Note: I2 is not taken into consideration for suffix because it doesn’thave any prefix at all.

Why Frequent Pattern Growth Fast ?

- Performance study shows
  - – FP-growth is an order of magnitude faster than Apriori,and is also faster than tree-projection
- Reasoning

– No candidate generation, no candidate test

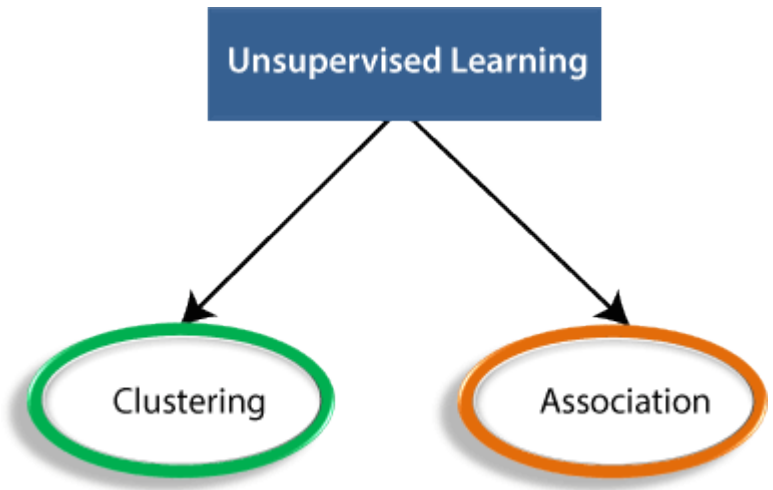
– Use compact data structure

– Eliminate repeated database scan

– Basic operation is counting and FP-tree building

Introduction to unsupervised Algorithms

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to ***analyze and cluster unlabeled datasets***.
- These algorithms *discover hidden patterns or data groupings* without the need for human intervention.
- Its ability to *discover similarities and differences in information* make it the ideal solution for exploratory data analysis, cross- selling strategies, customer segmentation, and image recognition.


Unsupervised Learning
Clustering
Association

K-Means Clustering Algorithm

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning.
- It groups the *unlabeled dataset into different clusters*.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It allows us to *cluster the data into different groups* and a convenient way to discover the categories of groups in the unlabeled dataset on its own *without the need for any training*.

- The k-means clustering algorithm mainly performs two tasks:
  - Determines the best value for K center points or centroids by an iterative process.
  - Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

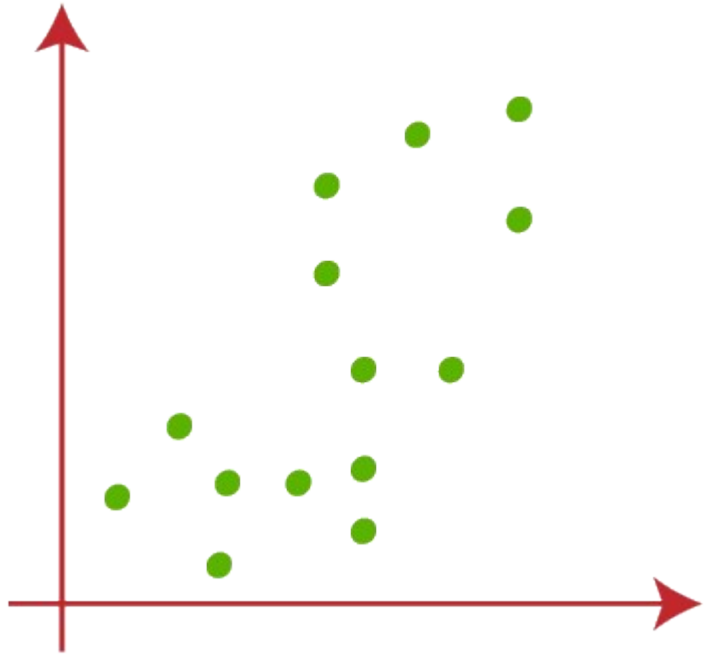
How does the K-Means Algorithm Work?

- The working of the K-Means algorithm is explained in the below steps:
- **Step-01:**
- Choose the number of clusters K.
- **Step-02:**
- Randomly select any K data points as cluster centers.
- Select cluster centers in such a way that they are as farther as possible from each other.
- **Step-03:**
- Calculate the distance between each data point and each cluster center.
- The distance may be calculated either by using given distance function or by using Euclidean distance formula.
-

K-Means Algorithm

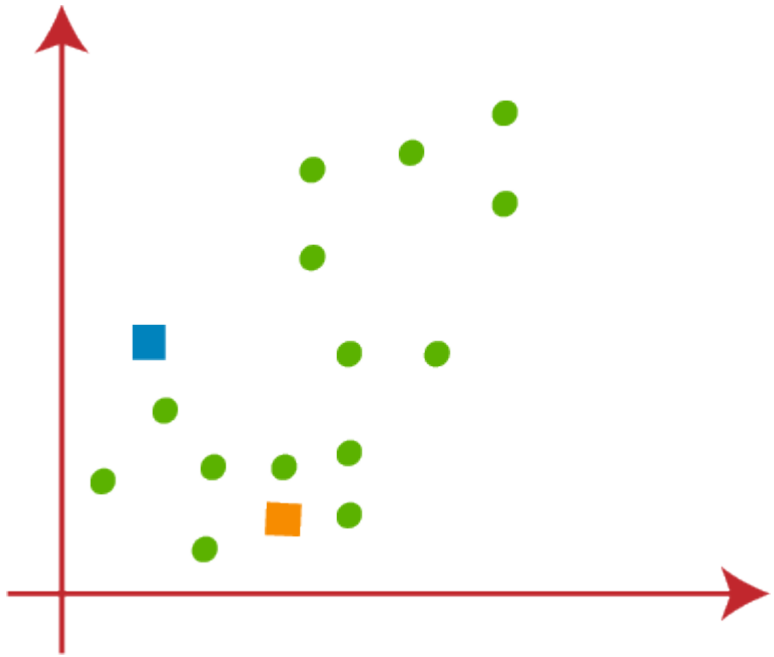
- **Step-04:**
- Assign each data point to some cluster.
- A data point is assigned to that cluster whose center is nearest to that data point.
- **Step-05:**
- Re-compute the center of newly formed clusters.
- The center of a cluster is computed by taking mean of all the data points contained in that cluster.
- **Step-06:**
- Keep repeating the procedure from Step-03 to Step-05 until any of the following stopping criteria is met-
  - Center of newly formed clusters do not change
  - Data points remain present in the same cluster
  - Maximum number of iterations are reached

- Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

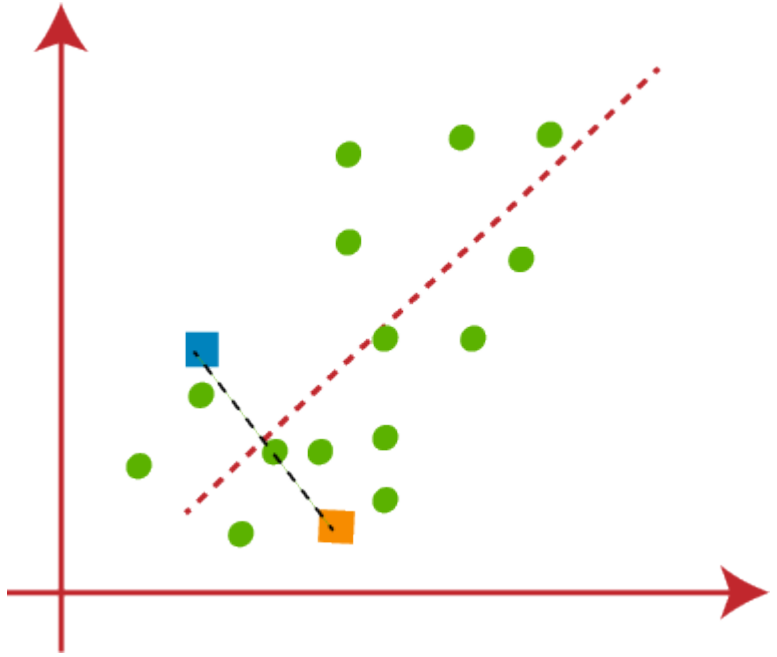


•Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.

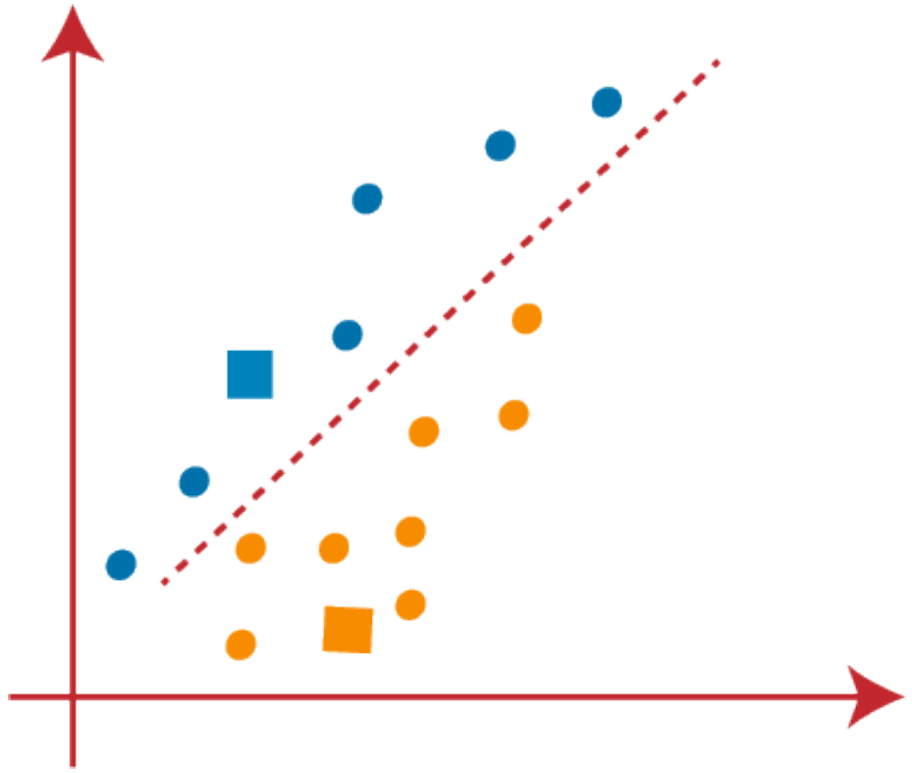
- We need to choose some *random k points or centroid* to form the cluster.
- These points can be either the points from the dataset or any other point.
- So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:



- Now we will assign each data point of the scatter plot to its closest K-point or centroid.
- We will compute it by applying some mathematics that we have studied to calculate the ***distance between two points.***
- So, we will draw a median between both the centroids. Consider the below image:

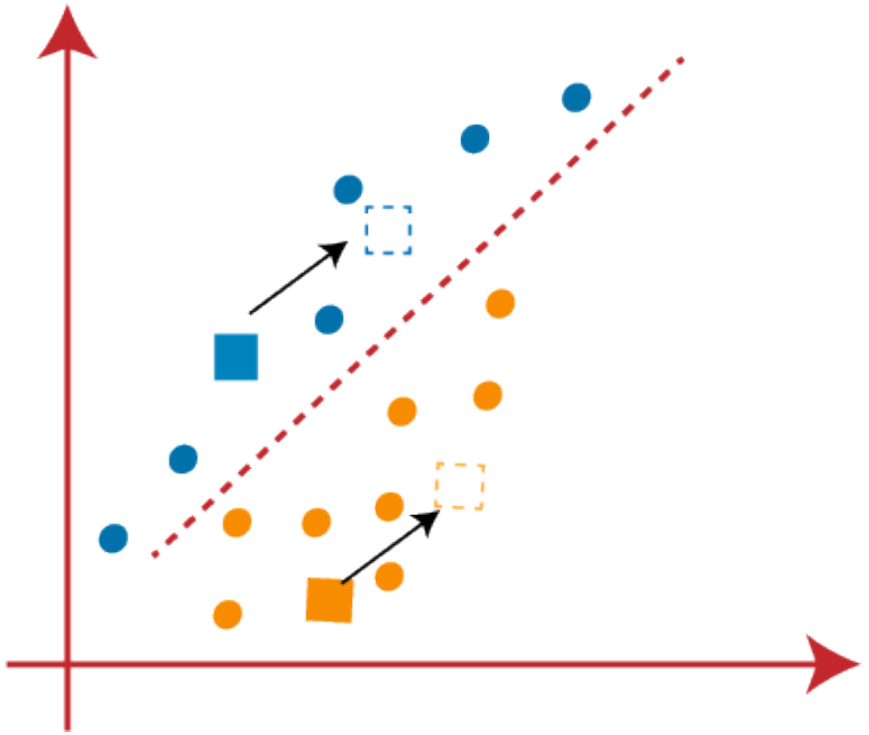


- From the image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid.
- Let's color them as blue and yellow for clear visualization.

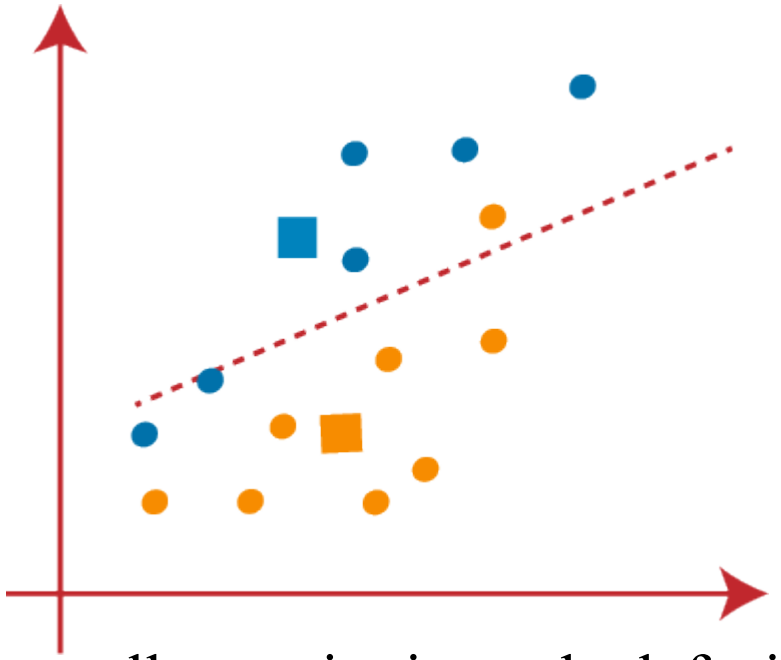


- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**.

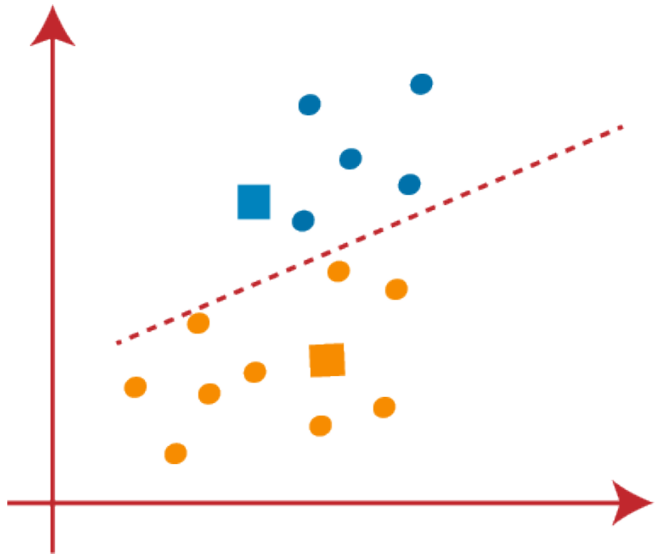
- To choose the new centroids, we will compute the center of gravity of these centroids(*mean* or average point.)



- Next, we will reassign each datapoint to the new centroid.
- For this, we will repeat the same process of finding a median line. The median will be like below image:

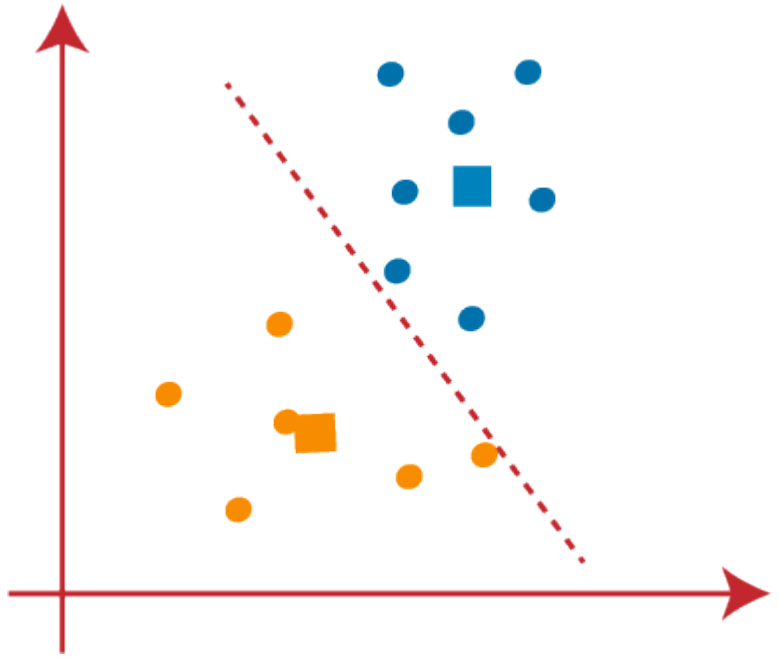


- From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

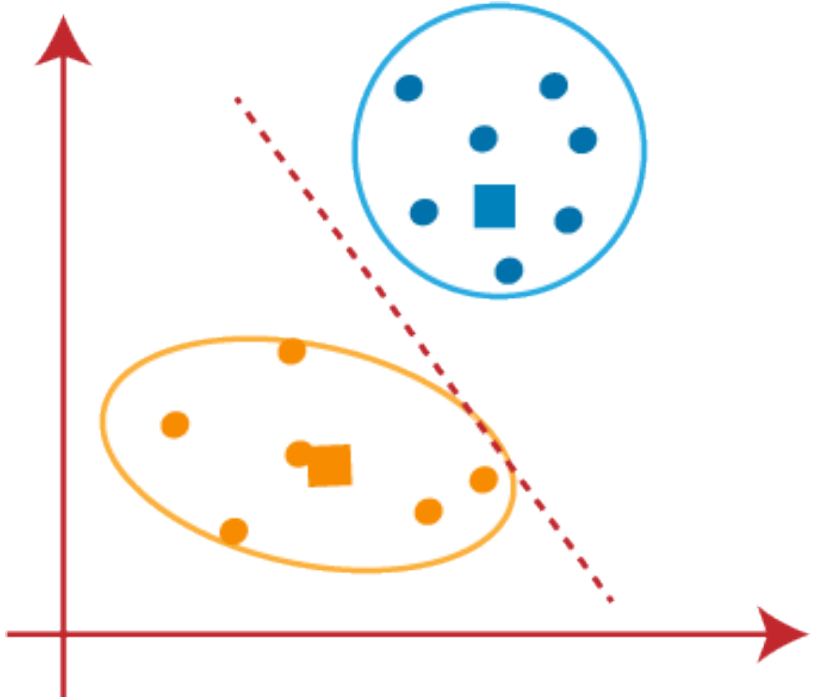


- As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:

- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:

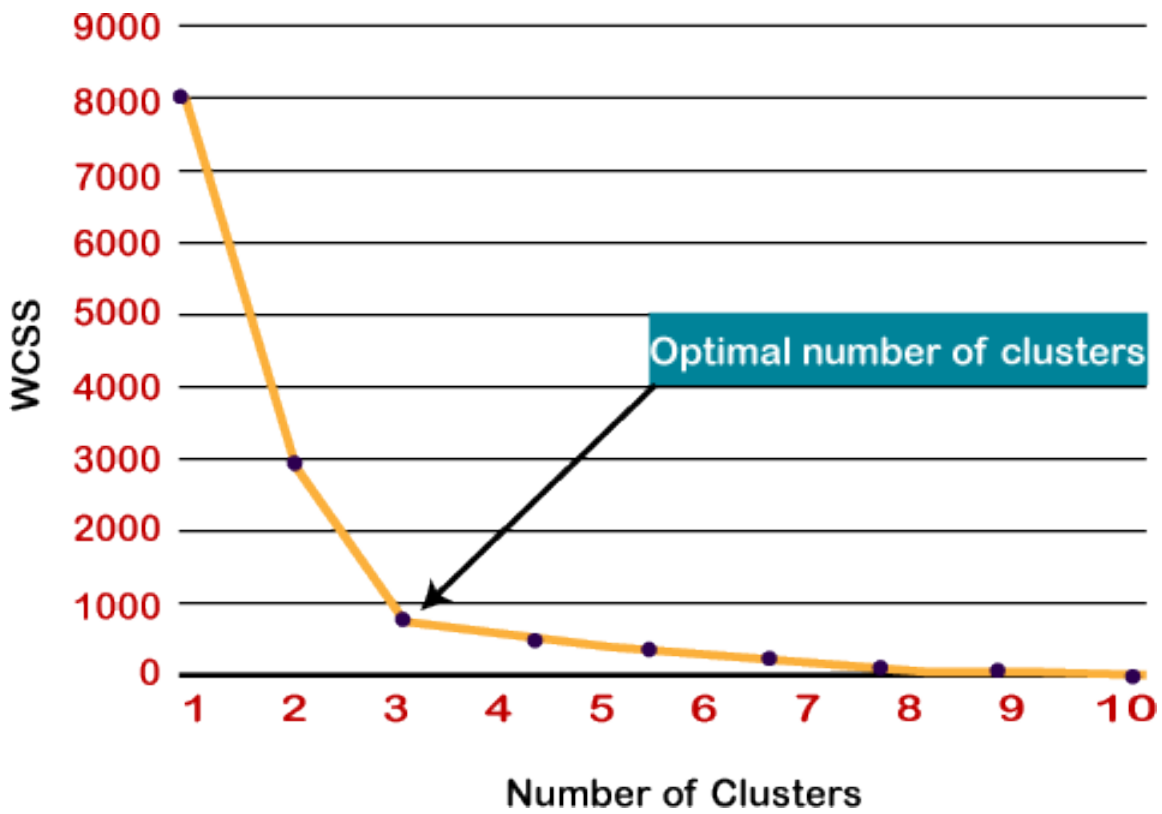


How to choose the value of "K number of clusters" in K-means Clustering?

- Elbow Method
- The Elbow method is one of the most popular ways to find the optimal number of clusters.
- This method uses the concept of WCSS value.
- **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.
- The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS= \sum_{P_i \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{P_i \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{P_i \text{ in CLuster3}} distance(P_i\ C_3)^2$$

- $\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.
- To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.
- To find the optimal value of clusters, the elbow method follows the below steps:
- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
WCSS
Optimal number of clusters
1
2
3
4
5
6
7
8
9
10
Number of Clusters

- The steps to be followed for the implementation are given below:
- **Data Pre-processing**
- **Finding the optimal number of clusters using the elbow method**
- **Training the K-means algorithm on the training dataset**
- **Visualizing the clusters**

Problem on K-Means Clustering

- Cluster the following eight points (with (x, y) representing locations) into three clusters:
- A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)
- Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
- The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as
- $P(a, b) = |x2 – x1| + |y2 – y1|$
- Use K-Means Algorithm to find the three cluster centers after the second iteration.

- **Iteration-01:**
-
- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.
-
- The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-
-
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
-
- P(A1, C1)
- = |x2 – x1| + |y2 – y1|
- = |2 – 2| + |10 – 10|
- = 0

- **Calculating Distance Between A1(2, 10) and C2(5, 8)-**
-
- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |5 – 2| + |8 – 10|
- = 3 + 2
- = 5
-
- **Calculating Distance Between A1(2, 10) and C3(1, 2)-**
-
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1 – 2| + |2 – 10|
- = 1 + 8
- = 9
-
- In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

- From here, New clusters are-
- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)
- **Cluster-03:**
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)

- **For Cluster-01:**
-
- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.
-
- **For Cluster-02:**
-
- Center of Cluster-02
- = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
- = (6, 6)
-
- **For Cluster-03:**
-
- Center of Cluster-03
- = ((2 + 1)/2, (5 + 2)/2)
- = (1.5, 3.5)
-

- **Iteration-02:**
- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.
- The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
- P(A1, C1)
- $= |x2 – x1| + |y2 – y1|$
- $= |2 – 2| + |10 – 10|$
- $= 0$

- **Calculating Distance Between A1(2, 10) and C2(6, 6)-**

- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |6 – 2| + |6 – 10|
- = 4 + 4
- = 8
-
- **Calculating Distance Between A1(2, 10) and C3(1.5, 3.5)-**
-
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1.5 – 2| + |3.5 – 10|
- = 0.5 + 6.5
- = 7
- In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

- From here, New clusters are-
- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- A8(4, 9)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- **Cluster-03:**
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)

- **For Cluster-01:**
-
- Center of Cluster-01
- = ((2 + 4)/2, (10 + 9)/2)
- = (3, 9.5)
-
- **For Cluster-02:**
-
- Center of Cluster-02
- = ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)
- = (6.5, 5.25)
-
- **For Cluster-03:**
-
- Center of Cluster-03
- = ((2 + 1)/2, (5 + 2)/2)
- = (1.5, 3.5)

- After second iteration, the center of the three clusters are-
- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)

- Using K means clustering algorithm form two clusters for given data.- Problem 1

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

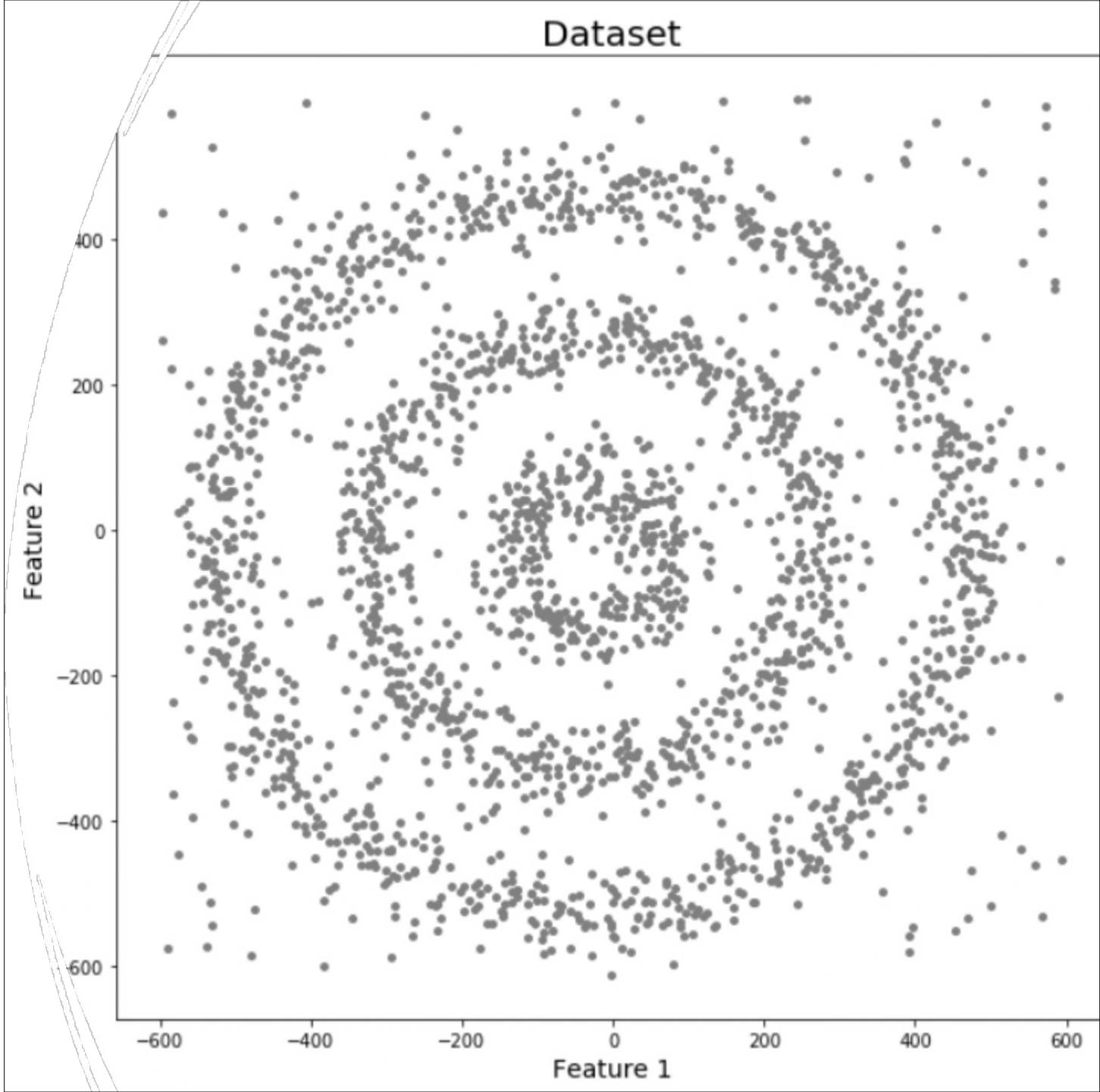
consider first two data points of our data and assign them as a centroid

Problem 2

- Lets consider we have cluster points P1(1,3) , P2(2,2) , P3(5,8) , P4(8,5) , P5(3,9) , P6(10,7) , P7(3,3) , P8(9,4) , P9(3,7).
- First, we take our K value as 3 and we assume that our Initial cluster centers are P7(3,3), P9(3,7), P8(9,4) as C1, C2, C3. We will find out the new centroids after 2 iterations for the above data points.

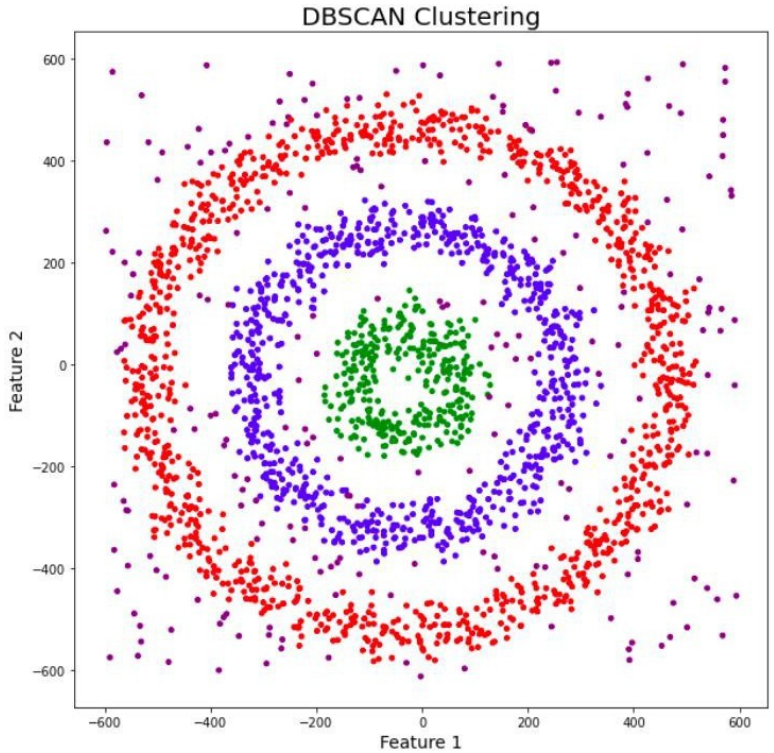
DBSCAN Clustering

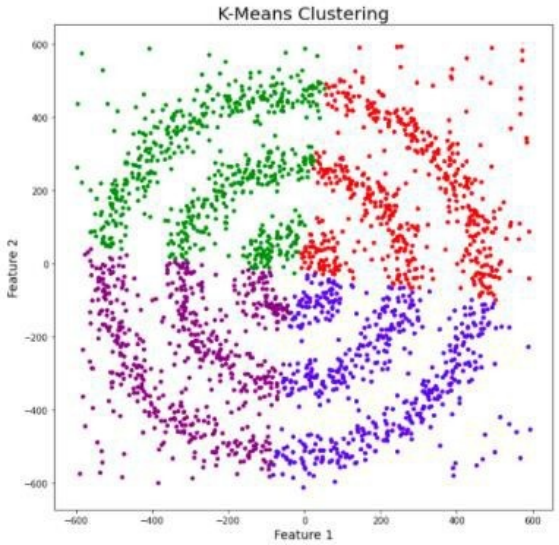
- K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering.
- we have data points densely present in the form of concentric circles:

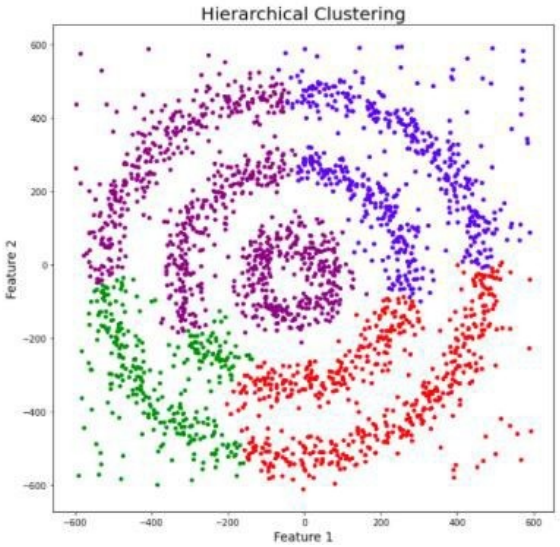


DBSCAN Clustering

- This data contains noise too, therefore, noise is ataken as a different cluster which is represented by the purple color. Both of K means, Hierarchical failed to cluster the data points. Also, they were not able to properly detect the noise present







DBSCAN Clustering

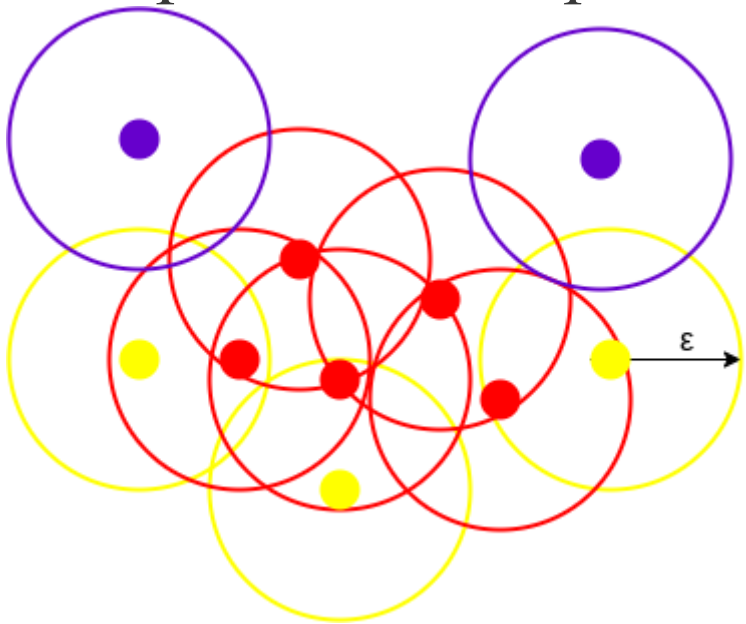
- **DBSCAN** stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.
- It was proposed by Martin Ester et al. in 1996. DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density.
- It groups 'densely grouped' data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. **The most exciting feature of DBSCAN clustering is that it is robust to outliers**. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

parameters required

1. **eps (ε):** This parameter defines the radius of a neighborhood around a data point. Points within this distance are considered neighbors of the central point.
2. **minPts:** This parameter represents the minimum number of points required within the ε-neighborhood of a point to classify it as a core point. A core point is considered to be dense enough to be part of a cluster.

- These parameters work together to identify high-density regions in your data. Points with enough neighbors within the specified distance (core points) are grouped together as clusters, while points with insufficient neighbors are classified as noise

- DBSCAN creates a circle of *epsilon* radius around every data point and classifies them into **Core** point, **Border** point, and **Noise**. A data point is a **Core** point if the circle around it contains at least '*minPoints*' number of points. If the number of points is less than *minPoints*, then it is classified as **Border** Point, and if there are no other data points around any data point within *epsilon* radius, then it treated as **Noise**.



- The above figure shows us a cluster created by DBCAN with *minPoints = 3*. Here, we draw a circle of equal radius *epsilon* around every data point. These two parameters help in creating spatial clusters.
- All the data points with at least 3 points in the circle including itself are considered as **Core** points represented by red color. All the data points with less than 3 but greater than 1 point in the circle including itself are considered as **Border** points. They are represented by yellow color. Finally, data points with no point other than itself present inside the circle are considered as **Noise** represented by the purple color.
- For locating data points in space, DBSCAN uses **Euclidean distance**, although other methods can also be used (like great circle distance for geographical data). It also needs to scan through the entire dataset once, whereas in other algorithms we have to do it multiple times.

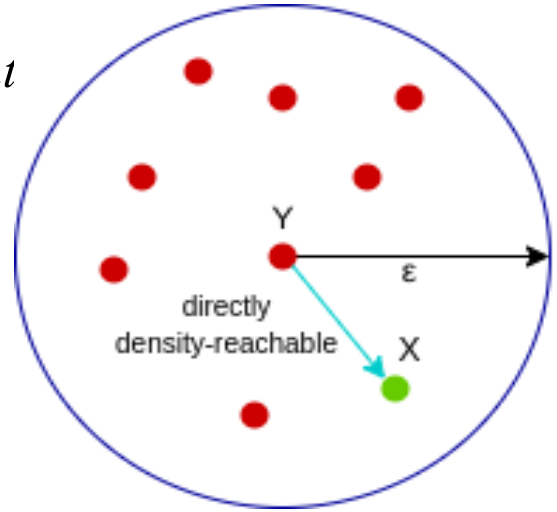
Reachability and Connectivity

- Reachability states if a data point can be accessed from another data point directly or indirectly, whereas Connectivity states whether two data points belong to the same cluster or not. In terms of reachability and connectivity, two points in DBSCAN can be referred to as:
- **Directly Density-Reachable**
- **Density-Reachable**
- **Density-Connected**
- A point **X** is **directly density-reachable** from point **Y** w.r.t *epsilon, minPoint*

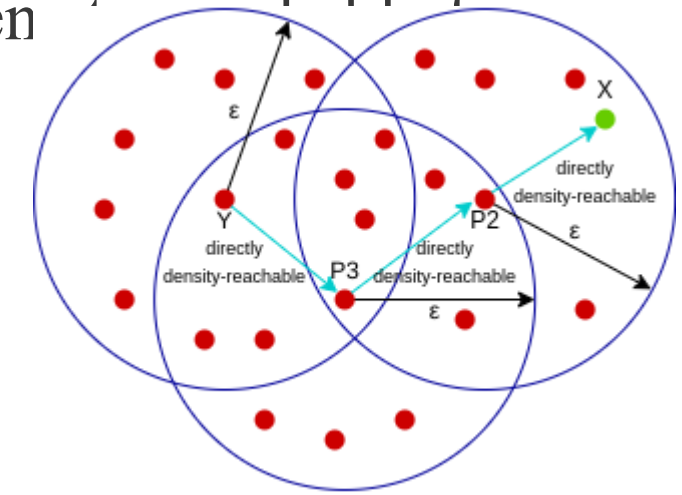
1. **X** belongs to the neighborhood of **Y**, i.e, *dist(X, Y) <= epsilon*

2. **Y** is a core point

Here, **X** is directly density-reachable from Y**,** but vice versa is not valid.

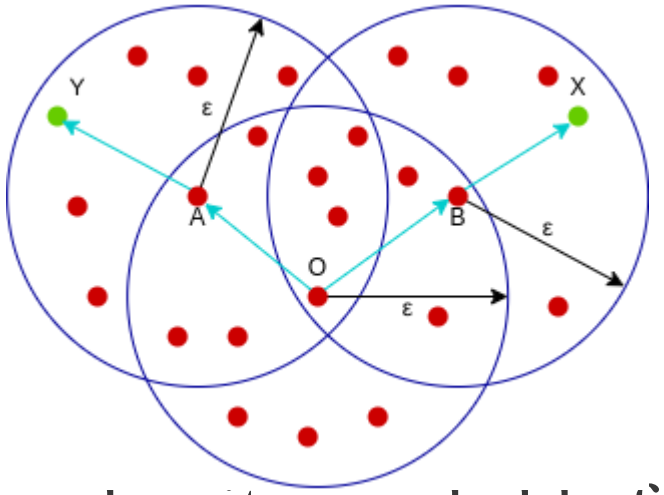


- A point **X** is **density-reachable** from point **Y** w.r.t *epsilon, minPoints* if there is a chain of points p1, p2, p3, …, pn and p1=**X** and pn=**Y** such that pi+1 is directly den



-
-
- Here, **X** is density-reachable from **Y** with **X** being directly density-reachable from **P2**, **P2** from **P3,** and **P3** from **Y.** But, the inverse of this is not valid.

- A point **X** is **density-connected** from point **Y** w.r.t *epsilon and minPoints* if there exists a point **O** such that both **X** and **Y** are density- reachable from **O** w.r.t to *epsilon and minPoints*.



- Here, both **X** and **Y** are density-reachable from **O**, therefore, we can say that **X** is density-connected from **Y**.

Working of Density-Based Clustering

- Suppose a set of objects is denoted by D', we can say that an object I is directly density reachable form the object j only if it is located within the ε neighborhood of j, and j is a core object.
- An object i is density reachable form the object j with respect to ε and MinPts in a given set of objects, D' only if there is a sequence of object chains point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly density reachable from i_i with respect to ε and MinPts.
- An object i is density connected object j with respect to ε and MinPts in a given set of objects, D' only if there is an object o belongs to D such that both point i and j are density reachable from o with respect to ε and MinPts.