PROGRAMME DESIGN COMMITTEE

Prof. (Retd.) S.K. Gupta , IIT, Delhi
Prof. Ela Kumar, IGDTUW, Delhi
Prof. T.V. Vijay Kumar JNU, New Delhi
Prof. Gayatri Dhingra, GVMITM, Sonipat
Mr. Milind Mahajan,. Impressico Business Solutions, New Delhi
Sh. Shashi Bhushan Sharma, Associate Professor, SOCIS, IGNOU
Sh. Akshay Kumar, Associate Professor, SOCIS, IGNOU
Dr. P. Venkata Suresh, Associate Professor, SOCIS, IGNOU
Dr. V.V. Subrahmanyam, Associate Professor, SOCIS, IGNOU
Sh. M.P. Mishra, Assistant Professor, SOCIS, IGNOU
Dr. Sudhansh Sharma, Assistant Professor, SOCIS, IGNOU

COURSE DESIGN COMMITTEE

Prof. T.V. Vijay Kumar, JNU, New Delhi
Prof. S.Balasundaram, JNU, New Delhi
Prof D.P. Vidyarthi, JNU, New Delhi
Prof. Anjana Gosain, USICT, GGSIPU, New Delhi
Dr. Ayesha Choudhary, JNU, New Delhi
Sh. Shashi Bhushan Sharma, Associate Professor, SOCIS, IGNOU
Sh. Akshay Kumar, Associate Professor, SOCIS, IGNOU
Dr. P. Venkata Suresh, Associate Professor, SOCIS, IGNOU
Dr. V.V. Subrahmanyam, Associate Professor, SOCIS, IGNOU
Sh. M.P. Mishra, Assistant Professor, SOCIS, IGNOU
Dr. Sudhansh Sharma, Assistant Professor, SOCIS, IGNOU

SOCIS FACULTY

Prof. P. Venkata Suresh, Director, SOCIS, IGNOU
Prof. V.V. Subrahmanyam, SOCIS, IGNOU
Dr. Akshay Kumar, Associate Professor, SOCIS, IGNOU
Dr. Naveen Kumar, Associate Professor, SOCIS, IGNOU (on EOL)
Dr. M.P. Mishra, Associate Professor, SOCIS, IGNOU
Dr. Sudhansh Sharma, Assistant Professor, SOCIS, IGNOU
Dr. Manish Kumar, Assistant Professor, SOCIS, IGNOU

PREPARATION TEAM

Dr.Sudhansh Sharma, (Writer- Unit 13)
Assistant Professor SOCIS, IGNOU

Ms. Divya Kwatra, (Writer Unit 14)
Assistant Professor, Department of Computer Science.
Hansraj College, University of Delhi

Ms Sofia Goel, (Writer Unit 15)
Research Scholar, SOCIS, IGNOU

Ms.Sanyukta Kesharwani (Writer - Unit 16)
Director(Reserach), Scholastic Seed Inc.,Delhi

Prof Anjana Gosain (Content Editor)
USICT-GGSIPU,Delhi

Dr. Rajesh Kumar(Language Editor)
SOH, IGNOU, New Delhi

Course Coordinator: Dr.Sudhansh Sharma,

Print Production

Sh Sanjay Aggarwal,Assistant Registrar, MPDD

, 2022

ISBN-

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110068.
Printed and published on behalf of the Indira Gandhi National Open University, New Delhi by MPDD, IGNOU.

UNIT 13 FEATURE SELECTION AND EXTRACTION

13.1 INTRODUCTION

Data sets are made up of numerous data columns, which are also referred to as data attributes. These data columns can be interpreted as dimensions on an n-dimensional feature space, and data rows can be interpreted as points inside that space. One can gain a better understanding of a dataset by applying geometry in this manner. In point of fact, these characteristics are measurements of the same entity. It is possible for their existence in the algorithm's logic to get muddled, which will result in a change to how well the model functions.

Input variables are the columns of data that are fed into a model in order to provide a forecast for a target variable. However, if your data is given in the form of rows and columns, such as in a spreadsheet, then features is another term that can be used interchangeably with input variables. It is possible that the presence of a large number of dimensions in the feature space implies that the volume of that space is enormous. As a result, the points (data rows) in that space reflect a small and non-representative sample of the space's contents. It is possible for the performance of machine learning algorithms to degrade when there are an excessive number of input variables. The existence of an excessive number of input variables has a significant impact on the efficiency with which machine learning algorithms function. when it is used to data that contains a large number of input attributes; this phenomenon is referred to as the "curse of dimensionality." As a consequence of this, one of the most common goals is to cut down on the number of input features. The process of decreasing the number of dimensions that characterise a feature space is referred to as "dimensionality reduction," which is a phrase that was made up specifically to describe this phenomenon.

The usefulness of data mining can be hindered by an excessive amount of information on occasion. There are occasions when only a handful of the columns of data characteristics that have been compiled for the purpose of constructing and testing a model do not offer any information that is significant to the model. However, there are some that actually reduce the reliability and precision of the model.

For instance, let's say you want to build a model that can forecast the incomes of people already employed in their respective fields. Therefore, data columns like cellphone number, house

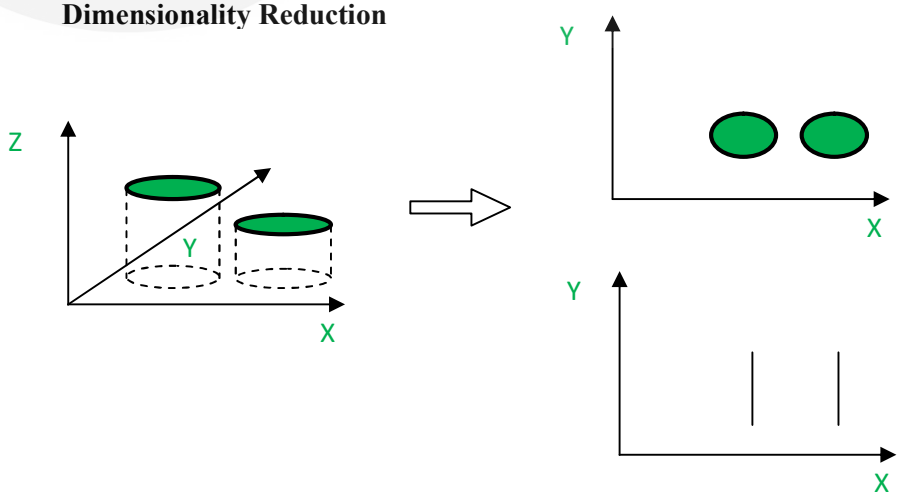
number, and so on will not truly contribute any value to the dataset, and they can therefore be omitted. This is because irrelevant qualities introduce noise to the data and affect the accuracy of the model. Additionally, because of the Noise, the size of the model as well as the amount of time and system resources required for model construction and scoring are both increased.

At this point in time, we are required to put the concept of Dimension Reductionality into practise. This can be done in one of two ways: either by selecting features to be extracted or by extracting features to be selected. Both of these approaches are broken down in greater detail below. The step of dimension reduction is one of the preprocessing phases that occurs during the process of data mining. This step is one of the preprocessing steps that may be beneficial in minimising the impacts of noise, correlation, and excessive dimensionality.

Some more examples are presented below to let you understand What does dimensionality reduction have to do with machine learning and predictive modelling?

- A simple issue concerning the classification of e-mails, in which we are tasked with deciding whether or not a certain email constitutes spam. can be brought up as a practical illustration of the concept of dimensionality reduction. This can include elements like whether or not the email has a standard subject line, the content of the email, whether or not it uses a template, and so on. However, some of these features may overlap with one another.
- A classification problem that involves humidity and rainfall can sometimes be simplified down to just one underlying feature as a result of the strong relationship that exists between the two variables. As a direct consequence of this, the number of characteristics could get cut down in some circumstances.
- A classification problem with three dimensions can be difficult to understand, whereas a problem with two dimensions can be translated to a fundamental space with two dimensions, and a problem with one dimension can be mapped to a line with one dimension. This concept is depicted in the diagram that follows, which shows how a three-dimensional feature space can be broken down into two one-dimensional feature spaces, with the number of features being reduced even further if it is discovered that they are related.

Dimensionality Reduction



In context of dimensionality reduction, various techniques like Principal Component Analysis, Linear Discriminant Analysis, Singular Value Decomposition are frequently used. In this unit we will discuss all the mentioned concepts, related to Dimension reductionality

13.2 Dimensionality Reduction

The Data mining and Machine Learning methodologies both have processing challenges when working with big amounts of data (many attributes). In point of fact, the dimensions of the feature space utilised by the approach, often referred to as the model attributes, play the most important function. Processing algorithms grow more difficult and time-consuming to implement as the dimensionality of the processing space increases.

These elements, also known as the model attributes, are the fundamental qualities, and they can either be variables or features. When there are more features, it is more difficult to see them all, and as a result, the work on the training set becomes more complex as well. This complexity was further increased when a significant number of characteristics were linked; hence, the classification became irrelevant as a result. In circumstances like these, the strategies for decreasing the number of dimensions can prove to be highly beneficial. In a nutshell, "the process of making a set of major variables from a huge number of random variables is what is referred to as dimension reduction." When conducting data mining, the step of dimension reduction can be helpful as a preprocessing step to lessen the negative effects of noise, correlation, and excessive dimensionality.

Dimension reduction can be accomplished in two ways :

- **Feature selection:** During this approach, a subset of the complete set of variables is selected; as a result, the number of conditions that can be utilised to illustrate the issue is narrowed down. It's normally done in one of three ways.:
    - Filter method
    - Wrapper method
    - Embedded method

- **Feature extraction:** It takes data from a space with many dimensions and transforms it into another environment with fewer dimensions.

13.2.1 Feature selection: It is the process of selecting some attributes from a given collection of prospective features, and then discarding the rest of the attributes that were considered. The use of feature selection can be done for one of two reasons: either to get a limited number of characteristics in order to prevent overfitting or to avoid having features that are redundant or irrelevant. For data scientists, the ability to pick features is a vital asset. It is essential to the success of the machine learning algorithm that you have a solid understanding of how to choose

the most relevant features to analyse. Features that are irrelevant, redundant, or noisy can contaminate an algorithm, which can have a detrimental impact on the learning performance, accuracy, and computing cost. The importance of feature selection is only going to increase as the size and complexity of the typical dataset continues to balloon at an exponential rate.

Feature Selection Methods: Feature selection methods can be divided into two categories: supervised, which are appropriate for use with labelled data, and unsupervised, which are appropriate for use with unlabeled data. Filter methods, wrapper methods, embedding methods, and hybrid methods are the four categories that unsupervised approaches fall under.:

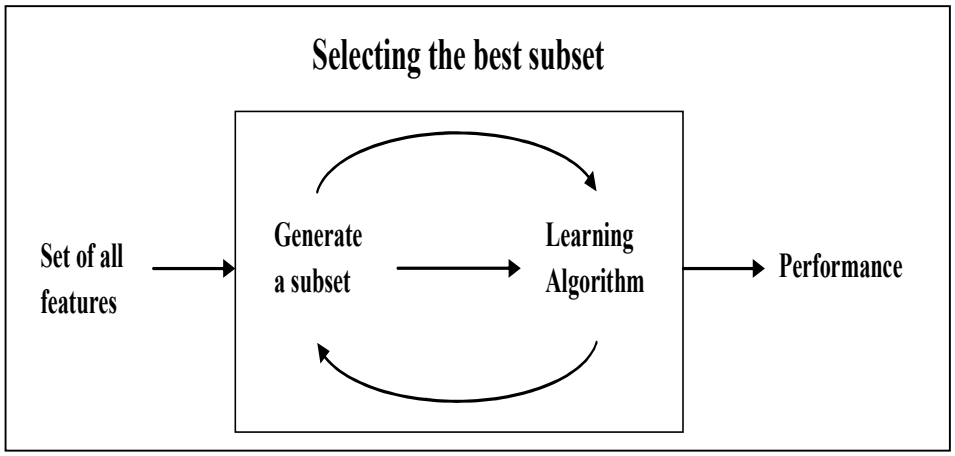
- **Filter methods**: Filter methods choose features based on statistics instead of how well they perform in feature selection cross-validation. Using a chosen metric, irrelevant attributes are found and recursive feature selection is done. Filter methods can be either univariate, in which an ordered ranking list of features is made to help choose the final subset of features, or multivariate, in which the relevance of all the features as a whole is evaluated to find features that are redundant or not important.

- **Wrapper methods**: Wrapper feature selection methods look at the choice of a set of features as a search problem. Their quality is judged by preparing, evaluating, and comparing a set of features to other sets of features. This method makes it easier to find possible interactions between variables. Wrapper methods focus on subsets of features that will help improve the quality of the results from the clustering algorithm used for the selection. Popular examples are Boruta feature selection and Forward feature selection.

- **Embedded methods:** Embedded feature selection approaches incorporate the feature selection machine learning algorithm as an integral component of the learning process. This allows for simultaneous classification and feature selection to take place within the method. Careful consideration is given to the extraction of the characteristics that will make the greatest contribution to each iteration of the process of training the model. A few examples of common embedded approaches are the LASSO feature selection algorithm, the random forest feature selection algorithm, and the decision tree feature selection algorithm.

Among all approaches the most conventional feature selection is feed forward feature selection.

Forward feature selection: The first step in the process of feature selection is to evaluate each individual feature and choose the one that results in the most effective algorithm model. This is referred to as "forward feature selection." After that step, each possible combination of the feature that was selected and a subsequent feature is analysed, and then a second feature is selected, and so on, until the required specified number of features is chosen. The operation of the forward feature selection algorithm is depicted here in the figure.



The procedure to follow in order to carry out forward feature selection

1. Train the model with each feature being treated as a separate entity, and then evaluate its overall performance.
2. Select the variable that results in the highest level of performance.
3. Carry on with the process while gradually introducing each variable.
4. The variable that produced the greatest amount of improvement is the one that gets kept.
5. Perform the entire process once more until the performance of the model does not show any meaningful signs of improvement.

Here, a fitness level prediction based on the three independent variables is used to show how forward feature selection works.

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit
10	190	F	No	Fit

So, the first step in Forward Feature Selection is to train n models and judge how well they work by looking at each feature on its own. So, if you have three independent variables, we'll train three models, one for each of these three traits. Let's say we trained the model using the Calories Burned feature and the Fitness Level goal variable and got an accuracy of 87 percent.

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit
10	190	F	No	Fit

Accuracy = 87%

We'll next use the Gender feature to train the model, and we acquire an accuracy of 80%. –

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit
10	190	F	No	Fit

Accuracy = 80%

And similarly, the **Plays_sport** variable gives us an accuracy of **85%**–

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit
10	190	F	No	Fit

Accuracy = 85%

At this point, we are going to select the variable that produced the most favourable results. If you take a look at this table, you'll notice that the variable titled "Calories Burned" alone has an accuracy rating of 87 percent, while the variable titled "Gender" has an accuracy rating of 80 percent, and the variable titled "Plays Sport" has an accuracy rating of 85 percent. When these two sets of data were compared, the winner was, unsurprisingly, the number of calories burned. As a direct result of this, we will select this variable.

Variable used	**Accuracy**
Calories_burnt	87.00%
Gender	80.00%
Plays_Sport?	85.00%

The next thing we'll do is repeat the previous steps, but this time we'll just add a single variable at a time. Because of this, it makes perfect sense for us to retain the Calories Burned variable as we proceed to add variables one at a time. Consequently, if we use gender as an illustration, we have an accuracy rate of 88 percent. –

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit
10	190	F	No	Fit

Accuracy = 88%

We acquire a 91 percent accuracy when we combine Plays Sport with Calories Burnt. The variable that yields the greatest improvement will be kept. That makes natural sense. As you can see, when we combine Plays Sport with Calories Burnt, we get a better result. As a result, we'll keep it and use it in our model. We'll keep repeating the process till all the features are considered in improving the model performance

ID	Calories_burnt	Gender	Plays_Sport?	Fintess Level
1	121	M	Yes	Fit
2	230	M	No	Fit
3	342	F	No	Unfit
4	70	M	Yes	Fit
5	278	F	Yes	Unfit
6	146	M	Yes	Fit
7	168	F	No	Unfit
8	231	F	Yes	Fit
9	150	M	No	Fit

10	190	F	No	Fit

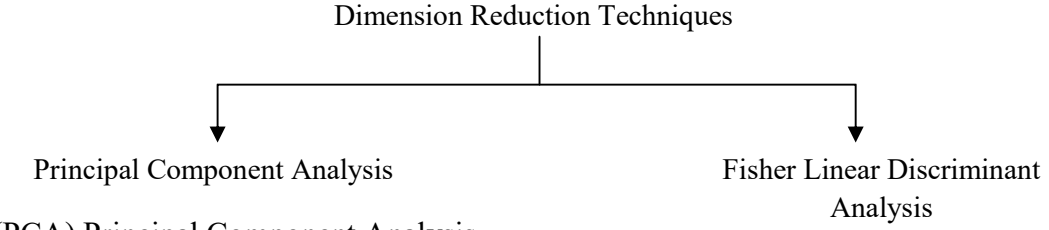
Accuracy = 91%

13.2.2 Feature extraction:

The process of reducing the amount of resources needed to describe a large amount of data is called "feature extraction." One of the main problems with doing complicated data analysis is that there are a lot of variables to keep track of. A large number of variables requires a lot of memory and processing power, and it can also cause a classification algorithm to overfit to training examples and fail to generalise to new samples. Feature extraction is a broad term for different ways to combine variables to get around these problems while still giving a true picture of the data. Many people who work with machine learning think that extracting features in the best way possible is the key to making good models. The data's information must be shown by the features in a way that fits the needs of the algorithm that will be used to solve the problem. Some "inherent" features can be taken straight from the raw data, but most of the time, we need to use these "inherent" features to find "relevant" features that we can use to solve the problem.

In simple terms *"feature extraction."* can be described as a technique for *Defining a set of features, or visual qualities, that best show the information.* Feature Extraction Techniques such as: PCA, ICA, LDA, LLE, t-SNE and AE. are some of the common examples in machine learning.

Feature extraction fills the following requirements:
It takes raw data, called features, and turns them into useful information by reformatting, combining, and changing the primary features into new ones. This process continues until a new set of data is created that the Machine Learning models can use to reach their goals.
Methods of Dimensionality Reduction : The following are two well-known and widely used dimension reduction techniques:



- (PCA) Principal Component Analysis
- (LDA) Fisher Linear Discriminant Analysis

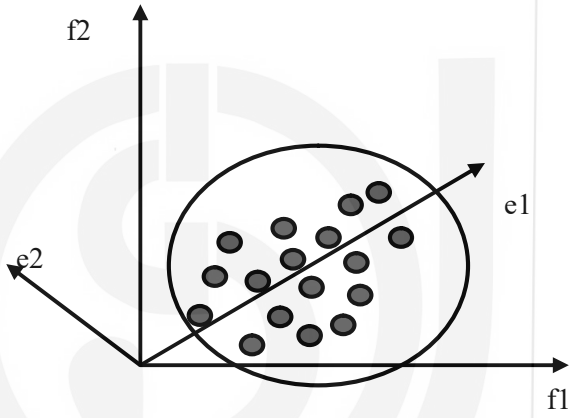
The reduction of dimensionality can be linear or non-linear, depending on the method used. The most common linear method is called Principal Component Analysis, or PCA.

Check Your Progress - 1

Qn1. Define the term feature selection.
Qn2. What is the purpose of feature extraction in machine learning?
Qn3. Expand the following terms : PCA,LDA,GDA
Qn4. Name components of dimensionality reduction.

13.3 Principal Component Analysis

Karl Pearson was the first person to come up with this plan. It is based on the idea that when data from a higher-dimensional space is put into a lower-dimensional space, the lower-dimensional space should have the most variation. In simple terms, principal component analysis (PCA) is a way to get important variables (in the form of components) from a large set of variables in a data set. It tends to find the direction in which the data is most spread out. PCA is more useful when you have data with three or more dimensions.



When applying the PCA method, the following are the primary steps that should be followed:

1. Obtain the dataset you need.
2. Calculate the mean of the vectors ().
3. Deduct the mean of the given data from the total.
4. Complete the computation for the covariance matrix.
5. Determine the eigenvectors and eigenvalues of the matrix that represents the covariance matrix.
6. Creating a feature vector and deciding which components would be the major ones i.e. the principal components.
7. Create a new data set by projecting the weight vector onto the dataset.As a result, we have a smaller number of eigenvectors, and some data may have been lost in the process. However, the remaining eigenvectors should keep the most significant variances.

Merits of Dimensionality Reduction

- It helps to compress data, which reduces the amount of space needed to store it and the amount of time it takes to process it.

- If there are any redundant features, it also helps to get rid of them.

Limitations of Dimensionality Reduction

- You might lose some data.
- You might lose some data.
- PCA fails when the mean and covariance are not enough to describe a dataset.
- We don't know how many major parts we need to keep track of, but in practice, we follow some rules.

Below is the practice question for Principal Component Analysis (PCA) :

Problem-01: 2, 3, 4, 5, 6, 7; 1, 5, 3, 6, 7, 8 are the given data. Using the PCA Algorithm, calculate the primary component.

OR

Consider the two-dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (8, 8) and (9, 10). (7, 8).

Using the PCA Algorithm, calculate the primary component.

OR

Calculate the principal component of following data-

Class1	values
X	2,3,4
Y	1,5,3

Class 2	values
X	5 , 6 , 7
Y	6 , 7 , 8

Answer :

Step-1: Get data.

The given feature vectors are- x1,x2,x3,x4,x5,x6 with the values given below:

$$\begin{bmatrix}2\\1\end{bmatrix}\ \begin{bmatrix}3\\5\end{bmatrix}\ \begin{bmatrix}4\\3\end{bmatrix}\ \begin{bmatrix}5\\6\end{bmatrix}\ \begin{bmatrix}6\\7\end{bmatrix}\ \begin{bmatrix}7\\8\end{bmatrix}$$

Step-2:

Find the mean vector (μ).

Mean vector (μ) = ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)= (4.5, 5)

Thus, Mean vector (μ) = $\begin{bmatrix}4.5\\5\end{bmatrix}$

Step-03:

On subtracting mean vector (μ) from the given feature vectors.

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$

same for others

Feature vectors (x_i) generated after subtraction are $\begin{bmatrix}-2.5\\-4\end{bmatrix}\begin{bmatrix}-1.5\\0\end{bmatrix}\begin{bmatrix}-0.5\\-2\end{bmatrix}\begin{bmatrix}0.5\\1\end{bmatrix}\begin{bmatrix}1.5\\2\end{bmatrix}\begin{bmatrix}2.5\\3\end{bmatrix}$

Step-04:

Now to find covariance matrix : $\text{Covariance Matrix} = \frac{\sum(X_i-\mu)(X_i-\mu)^t}{n}$

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix}-2.5\\-4\end{bmatrix}\begin{bmatrix}-2.5 & -4\end{bmatrix} = \begin{bmatrix}6.25 & 10\\10 & 16\end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix}-1.5\\0\end{bmatrix}\begin{bmatrix}-1.5 & 0\end{bmatrix} = \begin{bmatrix}2.25 & 0\\0 & 0\end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix}-0.5\\-2\end{bmatrix}\begin{bmatrix}-0.5 & -2\end{bmatrix} = \begin{bmatrix}0.25 & 1\\1 & 4\end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix}0.5\\1\end{bmatrix}\begin{bmatrix}0.5 & 1\end{bmatrix} = \begin{bmatrix}0.25 & 0.5\\0.5 & 4\end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix}1.5\\2\end{bmatrix}\begin{bmatrix}1.5 & 2\end{bmatrix} = \begin{bmatrix}2.25 & 3\\3 & 4\end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix}2.5\\3\end{bmatrix}\begin{bmatrix}2.5 & 3\end{bmatrix} = \begin{bmatrix}6.25 & 7.5\\7.5 & 9\end{bmatrix}$$

$$\text{Covariance Marix} = \frac{1}{6}\begin{bmatrix}17.5 & 22\\22 & 34\end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix}2.92 & 3.67\\3.67 & 5.67\end{bmatrix}$$

Step-05:

Eigen values and Eigen vectors of the covariance matrix.

$$\begin{vmatrix}2.92 & 3.67\\3.67 & 5.67\end{vmatrix} - \begin{vmatrix}\lambda & 0\\0 & \lambda\end{vmatrix} = 0$$

$$\begin{vmatrix}2.92 - \lambda & 3.67\\3.67 & 5.67 - \lambda\end{vmatrix} = 0$$

From here,

$$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \text{x } 3.67) = 0$$

$$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$$

$\lambda^2 - 8.56\lambda + 3.09 = 0$

Solving this quadratic equation, we get $\lambda = 8.22, 0.38$

This, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

Clearly, the second eigen value is vary small compared to the first eigen value.

So, the second eigen vactor can be left out.

Eigen vector corresponding to the greatest eigen value is the principle component for the given data set.

So, we find the eigen vector corresponding to eigen value λ_1.

We use the following equation to find the eigen vector-

$$MX = \lambda X$$

Where-

- $M = Covariance\ Matrix\ ;\ X = Eigen\ vector\ , and\ \lambda = Eigen\ value$

Substituting the values in the above equation, we get-

On being substituting the values in the above equation, we get-

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = 8.22 \begin{bmatrix} X1 \\ X2 \end{bmatrix}$$

Solving these, we get-

$2.92X_1 + 3.67X_2 = 8.22X_1$

$3.67X_1 + 5.67X_2 = 8.22X_2$

On simplification, we get-

$5.3X_1 = 3.67X_2$ ... ... (1)

$3.67X_1 = 2.55X_2$ ... ... (2)

From (1) and (2), $X_1 = 0.69X_2$

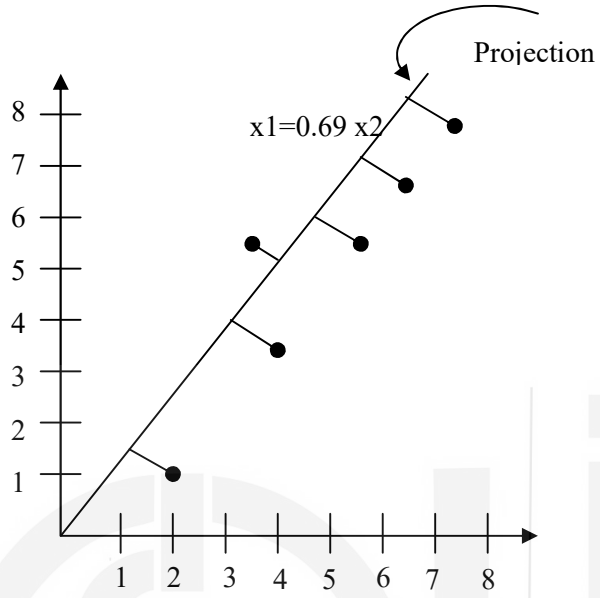
From (2), the eigen vector is-

$$\text{Eigen Vector}: \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Thus, PCA for the given problem is

Principle Component: $\begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Lastly, we project the data points onto the new subspace as-



Problem -02

Use PCA Algorithm to transform the pattern (2, 1) onto the eigen vector in the previous question.

Solution-

The given feature vector is (2, 1) i.e. Given Feature Vector: $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$

The feature vector gets transformed to :

= Transpose of Eigen vector x (Feature Vector – Mean Vector)

$$= \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}^T x \left(\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 4.5 \\ 5 \end{bmatrix} \right) = [2.55 \quad 3.67] \, x \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} = -21.055$$

Check Your Progress -3

Qn1. What are the advantages of dimensionality reduction?

Qn2. What are the disadvantages of dimensionality reduction?

13.4 Linear Discriminant Analysis

In most cases, the application of logistic regression has been restricted to problems involving two classes of subjects. On the other hand, the Linear Discriminant Analysis is the linear classification method that is recommended to use when there are more than two classes.

The algorithm for linear classification known as logistic regression is known for being both straightforward and robust. On the other hand, there are a few restrictions or faults in the system that highlight the requirement for more complex linear classification algorithms. The following is a list of some of the problems:

- **Binary class Problems**. Concerns regarding the binary class is that the Logistic regression is utilised for issues that involve binary classification or two classes. It is possible to enhance it such that it can manage multiple-class categorization, but in practise, this is not very common.
- **Unstable, but with well-defined classes**. When the classes are extremely distinct from one another, logistic regression may become unstable.
- It is prone to instability when there are only a few occurrences. When there are not enough examples from which to draw conclusions about the parameters, the logistic regression model may become unstable.

In view of the limitations of logistic regression that were discussed earlier, the linear discriminant analysis is one of the prospective linear methods that can be used for multi-class classification. This is because it addresses each of the aforementioned concerns in their totality, which is the primary reason for its success (i.e. flaws of logistic regression). When dealing with issues that include binary categorization, two statistical methods that could be effective are logical regression and linear discriminant analysis. Both of these techniques are linear and regression-based.

Understanding LDA Models : In order to simplify the analysis of your data and make it more accessible, LDA will make the following assumptions about it:

1. The distribution of your data is Gaussian, and when plotted, each variable appears to be a bell curve.

2. Each feature has the same variance, which indicates that the values of each feature vary by the same amount on average in relation to the mean.

On the basis of these presumptions, the LDA model generates estimates for both the mean and the variance of each class. In the case where there is only one input variable, which is known as the univariate scenario, it is straightforward to think about this.

When the sum of values is divided by the total number of values, we are able to compute the mean value, or mu, of each input, or x, for each class(k), in the following manner.

$$mu_k = 1/nk * sum(x)$$

Where,
mu_k represents the average value of x for class k and
nk represents the total number of occurrences that belong to class k.
When calculating the variance across all classes, the average squared difference of each individual result's distance from the mean is employed.

$$sigma^2 = 1 / (n-K) * sum((x - mu)^2)$$

Where sigma^2 represents the variance of all inputs (x), n represents the number of instances, K represents the number of classes, and mu is the mean for input x.

Now we will discuss How to use LDA to Make Predictions ?

LDA generates predictions by calculating the likelihood that each class will be given a fresh batch of data and then extrapolating from there. A forecast is created by selecting the output class that contains the events that are the most likely to occur. The Bayes Theorem is incorporated into the model in order to calculate the probabilities involved. Utilizing the likelihood of each class as well as the probability of data belonging to that class, Bayes's Theorem may be utilised to estimate the probability of the output class (k) given the input class (x). This is accomplished by using the following formula:

$$P(Y=x|X=x) = (PIk * fk(x)) / sum(PIl * fl(x))$$

The base probability of each class (k) that can be found in your training data is denoted by the symbol PIk (e.g. 0.5 for a 50-50 split in a two class problem). This concept is referred to as the prior probability within Bayes' Theorem.

$$PIk = nk/n$$

The value of f, which represents the estimated likelihood that x is a member of the class, is presented here as f(x). We make use of a Gaussian distribution function for the variable (x). By simplifying the previous equation and then introducing the Gaussian, we are able to arrive at the equation that is presented below. This type of function is referred to as a discriminate function, and the output classification (y) is determined by selecting the category that contains the greatest value:

$$D_k(x) = x * (mu_k/sigma^2) - (mu_k^2/(2*sigma^2)) + ln(PIk)$$

Where, $D_k(x)$ is the discriminating function for class k given input x, and mu_k, sigma^2, and PIk are all estimated from your data.

Now to perform the above task we need to prepare our data first, so the question arises, How to prepare data suitable for LDA?

This section gives you some ideas to think about when getting your data ready to use with LDA.

Problems with Classification: LDA is used to solve classification problems where the output variable is a categorical one. This may seem obvious, as LDA works with both two and more than two classes.

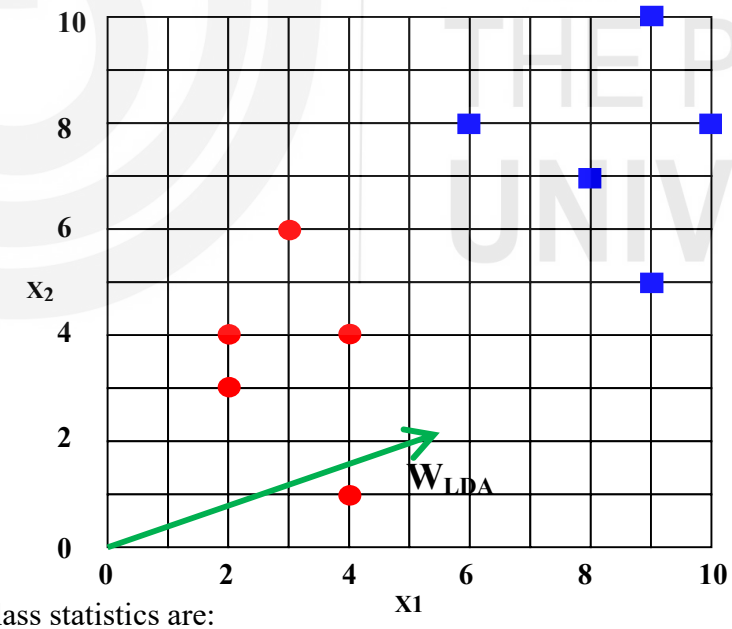
Gaussian Distribution: The standard way to use the model assumes that the input variables have a Gaussian distribution. Think about looking at the univariate distributions of each attribute and using transformations to make them look more like Gaussian distributions (e.g. log and root for exponential distributions and Box-Cox for skewed distributions).

Remove Outliers: Think about removing outliers from your data. These things can mess up the basic statistics like the mean and the standard deviation that LDA uses to divide classes.

Same Variance: LDA assumes that the variance of each input variable is the same. Before using LDA, you should almost always normalise your data so that it has a mean of 0 and a standard deviation of 1.

Below is a practice problems based on Linear Discriminant Analysis (LDA) -

Problem-2 : Compute the Linear Discriminant projection for the following two-dimensional datasetX1=(x1,x2)={(4,1),(2,4),(2,3),(3,6),(4,4)} & X2=(x1,x2)={(9,10),(6,8),(9,5),(8,7),(10,8)}



- The class statistics are:

$$S_1 = \begin{bmatrix} 0.80 & -0.40 \\ -0.40 & 2.60 \end{bmatrix} : S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix}$$

$$\mu_1 = [3.00 \quad 3.60] : \mu_2 = [8.40 \quad 7.60]$$

- The within- and between-class scatter are

$$S_1 = \begin{bmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{bmatrix} : S_2 = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

- The LDA projection is then obtained as the solution of the generalized eigenvalue problem

$$S_w^{.1} S_B v = \lambda v \Rightarrow \left| S_w^{.1} S_B - \lambda I \right| = 0 \Rightarrow \left| \begin{matrix} 11.89 & 8.81 \\ 5.08 & 3.76 - \lambda \end{matrix} \right| = 0 \Rightarrow \lambda = 15.65$$

$$\begin{bmatrix} 11.89 & 8.81 \\ 5.08 & 3.76 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 15.65 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.39 \end{bmatrix}$$

- On directly by

$$w^* = S_w^{-1}(\mu_1 - \mu_2) = [-0.91 \quad -0.39]^T$$

Check Your Progress -3

Q1. Define LDA.

Q2. Write any two limitations of LDA.

13.5 Single Value Decomposition

The Singular Value Decomposition (SVD) method is a well-known technique for decomposing a matrix into a large number of component matrices. This method is valuable since it reveals many of the interesting and helpful characteristics of the initial matrix. We can use SVD to discover the optimal lower-rank approximation to the matrix, determine the rank of the matrix, or test a linear system's sensitivity to numerical error.

Singular value decomposition is a method of decomposing a matrix into three smaller matrices.

$$\mathbf{A = USV^T}$$

Where:

- A : is an $m \times n$ matrix
- U : is an $m \times n$ *orthogonal* matrix
- S : is an $n \times n$ *diagonal matrix*
- V : is an $n \times n$ orthogonal matrix

The rationale for the transposition of the last matrix will be revealed later in the presentation. Also defined (in case your algebra is rusty) will be the word "orthogonal," as well as it describes the reason for having two outer matrices having the same feature.

The diagonal matrix, S, has been flattened into a vector, reducing the formula into a single summation. Singular values, or Si, are variables that are generally organized from largest to smallest.

Below is a practice problems based on single value decomposition

Problem-03: Find the SVD of the matrix $A = \begin{bmatrix} -3 & 1 \\ 6 & -2 \\ 6 & -2 \end{bmatrix}$

Solution : First, we'll work with $A^T A = \begin{bmatrix} 81 & -27 \\ -27 & 9 \end{bmatrix}$. The eigen values are $\lambda = 0,\ 90$.

For $\lambda = 0$, the reduced matrix is $\begin{bmatrix} 1 & -1/3 \\ 0 & 0 \end{bmatrix}$, so v $= \frac{1}{\sqrt{10}} \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

For $\lambda = 90$, the reduced matrix is $\begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}$, so v $= \frac{1}{\sqrt{10}} \begin{bmatrix} -3 \\ 1 \end{bmatrix}$

Now, we can find the reduced SVD right away since $u_1 = \frac{1}{\sigma_1} A V_1 = \frac{1}{3} \begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix}$

We now need a basis for the null space of

$$AA^T = \begin{bmatrix} 10 & -20 & -20 \\ -20 & 40 & 40 \\ -40 & 40 & 40 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & -2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \left\{ \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Now the full SVD is given by:

$$A = \begin{bmatrix} -3 & 1 \\ 6 & -2 \\ 6 & -2 \end{bmatrix} = \begin{bmatrix} 1/3 & 2/\sqrt{5} & 2/\sqrt{5} \\ -2/3 & 1/\sqrt{5} & 0 \\ -2/3 & 0 & 1/\sqrt{5} \end{bmatrix} \begin{bmatrix} 2/\sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & 3/\sqrt{10} \end{bmatrix}^T$$

Check Your Progress - 4

Qn1. Define SVD

13.6 SUMMARY

In this unit we learned about the concept of Dimension Reductionality, wherein we understood basics of Feature Selection and Feature extraction techniques. Thereafter an explicit discussion of Principal Component Analysis(PCA), Linear Discriminant Analysis(LDA) and Singular Value Decomposition(SVD) was given.

13.7 SOLUTIONS TO CHECK YOUR PROGRESS

Check Your Progress – 1

Refer Section 13.2 for detailed Solutions

Ans1. In machine learning and statistics, feature selection is the process of choosing a subset of relevant features to use when making a model. It is also called variable selection, attribute selection, or variable subset selection.

Ans2. The goal of Feature Extraction is to reduce the number of features in a dataset by making new features from the ones that are already there (and then discarding the original features).

Refer Section 13.2 for detailed Solutions

Ans3.Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Generalized Discriminant Analysis (GDA)

Ans4. Feature selection and feature extraction.

Check Your Progress - 2

Refer Section 13.3 for detailed Solutions

Ans1. Advantages of dimensionality reduction

- It reduces the time and space complexity.
- Getting rid of multi-colinearity makes it easier to understand how the machine learning model's parameters work.
- When there are only two or three dimensions, it's easier to see the data.

Ans2. Dimensionality Reduction's Drawbacks

- PCA likes to find linear relationships between variables, which isn't always a good thing.
- PCA doesn't work when the mean and covariance aren't enough to describe a dataset.
- In some problems, dimension reduction could also cause data loss.

Check Your Progress - 3

Refer Section 13.4 for detailed Solutions

Ans1. The LDA technique is a multi-class classification method that may be utilised to automatically perform dimensionality reduction. LDA cuts the number of features down from the initial number of features.

LDA projects the data into a new linear feature space, and obviously, the classifier will have a high level of accuracy if the data can be linearly separated.**Ans2**. Some of the limitations of Logistic Regression are as follows:

- Logistic regression is typically applied when attempting to solve problems involving binary or two-class classifications. Problems involving two classes Even while it is

possible to extrapolate this information and apply it for multi-class classification, very few people actually do this. On the other hand, Linear Discriminant Analysis is regarded as a superior option whenever multi-class classification is necessary, and in the case of binary classifications, both logistic regression and LDA are utilised in the analysis process.

- Instability with Clearly Delineated Social Groups — Logistic Regression is known to be unreliable in situations when the classes are clearly differentiated from one another.

Check Your Progress - 4

Refer Section 13.5 for detailed Solutions

Ans1 The singular value decomposition is a factorization technique that can be used in linear algebra for real or complex matrices. It extends the application of the eigen decomposition of a square normal matrix to any m x n matrix by using an ortho normal eigen basis. It has something to do with the polar decomposition.

13.8 FURTHER READINGS

1. Machine learning an algorithm perspective, Stephen Marshland, 2nd Edition, CRC Press, 2015.
2. Machine Learning, Tom Mitchell, 1st Edition, McGraw- Hill, 1997.
3. Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Peter Flach, 1st Edition, Cambridge University Press, 2012.

UNIT14 ASSOCIATION RULES

Structure

14.1 INTRODUCTION

Imagine a salesperson at a shop. If you buy a product from the shop, the salespersonrecommends you more items related to the product you have purchased. He may also suggest you about the items that are frequently bought with the product you have purchased. The salesperson may also try to figure out your choices about the product you observe at the shop and may recommend you further. One of the common examples of the situation is **market basket analysis** as illustrated in two cases in Figure 1. If you add bread and butter to your basked at the store, the salesperson may recommend you add cookies, eggs, milk to your shopping cart too. Similarly, if customer puts vegetables like onion and potato in their cart, the salesman at shop may suggest adding other vegetables like tomato to the basket. If salesman notices a male customer, then he may suggest adding beer too from his historical analysis of male customer preferring to buy beer as well.

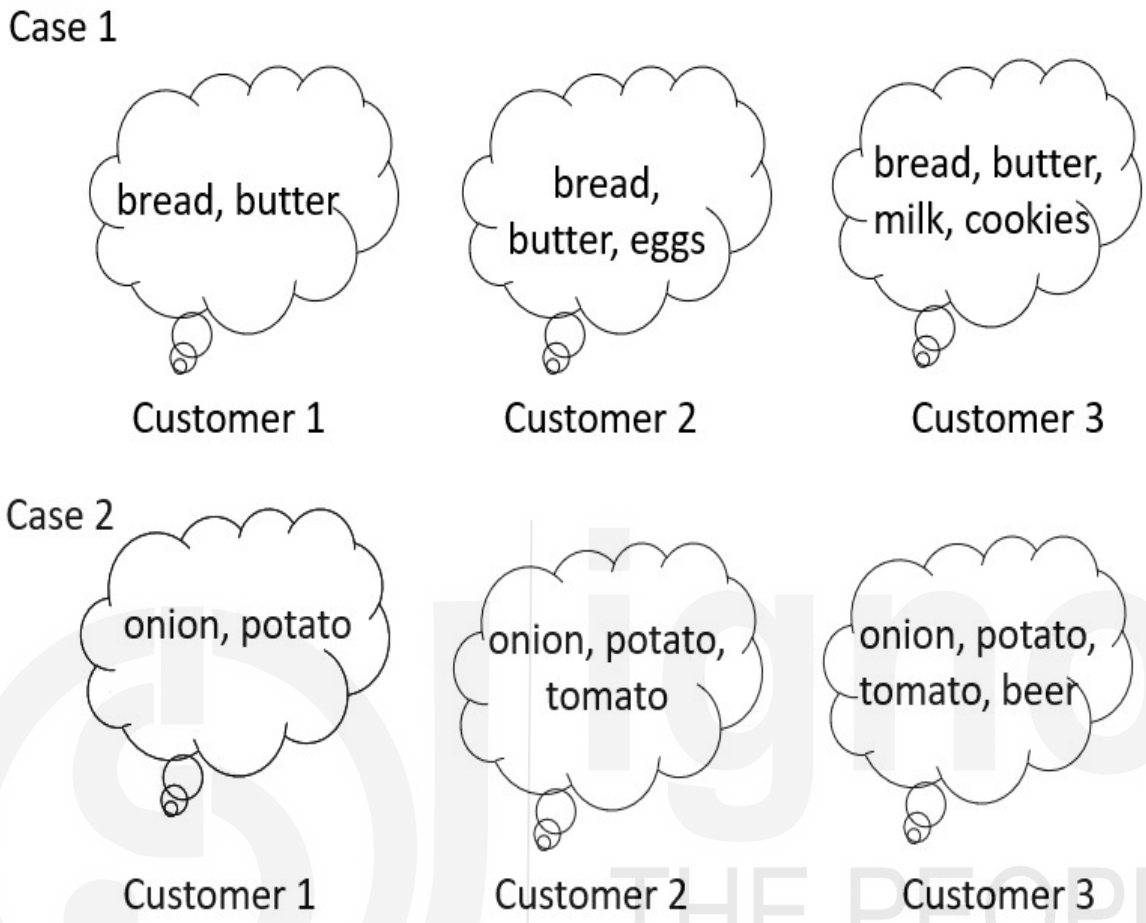


Figure 1: Two different cases of Market Basket Analysis

The salesperson thus analyses the purchasing habits of the customers and tries to analyze the correlations among the items/products that are frequently bought together by them. The analysis of such correlationshelps the retailers to develop marketing strategies to increase sale of products in the shop. Discovering the correlations among all the items sold in a shop help businessman in making decisions regarding designing catalogue, organizing store, and customer shopping analysis.

OBJECTIVES

After going through this unit, you should be able to:

- Understand the purpose of association rules.
- Understand the purpose of pattern search and,
- Describe various algorithms for pattern search

14.3 WHAT ARE ASSOCIATION RULES?

In machine learning, association rules are one of the important concepts that is widely applied in problems like market basket analysis. Consider a supermarket, where all the related items such as grocery items, dairy items, cosmetics, stationary items etc are kept together in same aisle. This helps the customers to find their required items timely. This further helps them to remember the items to purchase they might have forgotten or to they may like to purchase if suggested. Association rules thus enable one to corelate among various products from a huge set of available items. Analysing the items customer buy together also helps the retailers to identify the items they can offer on discount. For example, retailer selling baby lotion and baby shampoo on MRP, but offering a discount on their combination. Customer who wished to buy only shampoo or only lotion, may now think of buying the combination. Other factors too can contribute to the purchase of combination of products. Another strategy can keep related products on the opposite ends of the shelf to prompt the customer to scan through the entire shelf hoping that he might add a few more items to his cart.

It is important to note that the association rules do not extract the customer's preference about the items but find the relations among the items that are generally bought together by them. The rules only identify the frequent associations between the items. The rules work with an antecedent (if) and a consequent (then), both connecting to the set of items. For example, if a person buys pizza, then he may buy a cold drink too. This is because there is a strong relation between pizza and cold drink. Association rules help to find the dependency of one item on other by consider the history of customer's transaction patterns.

14.3.1 Basic Concepts

There are few terms that one should understand before understanding the algorithm.

a. **k-Itemset:** It is a set of *k*items. For example, 2-itemset can be {pencil, eraser} or {bread, butter} etc., 3-itemset can be {bread, butter, milk}.

b. **Support:** Frequency of appearance of an item appears in all the considered transactions is called as the support of an item. Mathematically, support of an item *x* is defined as:

$$support(x) = \frac{Number\ of\ transactions\ containing\ x}{Total\ number\ of\ considered\ transctions}$$

c. **Confidence:** Confidence is defined as the likelihood of obtaining item *y* along with an item *x*. Mathematically, it is defined as the ratio of frequency of transactions containing items x and *y* to the frequency of transactions that contained item *x*.

$$confidence(y) = \frac{Number\ of\ transactions\ containing\ x\ and\ y}{Number\ of\ considered\ containing\ x}$$

Confidence can also be defined as probability of occurrence of y, given probability of occurrence of x.

$$confidence(x => y) = P(y/x)$$

where x is antecedent, and y is a consequent. In terms of support, confidence can be described as:

$$confidence(y) = \frac{support\ (x \cup y)}{support(x)}$$

d. **Frequent Itemset:** An item whose support is at least the minimum support threshold is known as a *frequent itemset*. For example, let minimum support threshold is 10, then an item set with support score 11 is a frequent itemset but an item set with support score 9 is not.

14.3.2 Association Rules: Binary Representation

Let I = {I_1, I_2, I_3……..I_n} be a set of n items and T = {T_1, T_2, T_3……..T_t} be a set of t transactions, where each transaction T_i contains a not null set of items purchased by a customer such that $T_i \subseteq I$. Let each item I_i in the store be represented by a binary variable, B. The variable takes up the value 0 or 1, representing the absence or presence of item at the store.

$$B(i) = 1, \text{if an item } I_i \text{ is available at the store}$$
$$0, \text{otherwise}$$

For example, consider a set of four transactions T1, T2, T3 and T4 with the following items:

T1 = {milk, cookies, bread},
T2 = {milk, bread, egg, butter},
T3 = {milk, cookies, bread, butter}, and
T4 = {cookies, bread, butter}.

The binary representations for the transaction set are shown in Table 1.

Table1: Binary representation of the transactions

Transaction id	Milk	Cookies	Bread	Butter	Egg
T1	1	1	1	0	0
T2	1	0	1	1	1
T3	1	1	1	1	0
T4	0	1	1	1	0

The binary variable can also be used to analyze the purchasing patterns of the customers. One can analyze a basket in terms of binary values of the items that customer has purchased. Such items are said to frequently bought together or associated to each other. These binary patterns are represented in the form of association rules. For example, the customer who buys milk, also tend to buy bread at the same time. This can be represented using support and confidence as:

Milk => Bread with support as 75% (or value as 0.75) and confidence as 100% (or value as 1).

Support(milk) = $\frac{transactions\ containing\ milk}{total\ number\ of\ transactions} = \frac{3}{4} = 0.75$

Confidence (bread) = $\frac{support(milk,bread)}{support(milk)} = \frac{\frac{3}{4}}{0{\cdot}75} = \frac{0.75}{0.75} = 1$

This means that out of all the purchases made at the store, 10% of the times, whenever milk is purchased, bread is also purchased together. Confidence 100% implies that out of all the customers who bought milk, all of them also bought bread. Thus, support and confidence are the two important rules to show the interest in items. Thus, in this case association rules are important to consider if they satisfy the minimum threshold of support and confidence. These thresholds can be set by the experts.

Let A and B be the two set of transactions such that A ⊆ T and B ⊆ T, such that A ≠Ø, B ≠Ø and A∩ B=Ø. For a given transaction T_i, the rule A => B holds with support *s* and confidence *c* as defined in section 1.3.1. Support *s* is defined as the percentage of transactions that contains the items contained in set A as well as in set B i.e, union of set A and set B represented as A∪B. Confidence *c* is defined as the percentage of transactions containing A also containing B. When writing support and confidence in terms of percentage, their value range between 0 and 100, and when expressed as ratios, their values range between 0 and 1.

Association rules can further be defined as *strong* association rules if they satisfy the minimum threshold support called as *min_sup* and minimum threshold confidence called as *min_conf.*

14.3.3 Association Rules Discovery

The problem of discovery of association rules can be stated as: Given a set of transactions *T,* find the rules whose support and confidence are greater than equal to the minimum support and confidence threshold.

Traditional approach to generate association rules is to compute the support and confidence for every possible combination of items. Butt his approach is computationally not possible as the number of combinations of items can be exponentially large. To avoid such large number of computations, basic approach should be to ignore the needless computations without computing their support and confidence scores. For example, we can observe from Table 1 that the combination {milk, egg} can be ignored as the combination if infrequent. Hence, we prune the rule

Milk => Egg without computing the support and confidence for the items.

Therefore, the steps for obtaining association rules can be summarized as:

1. *Find all frequent item sets:* By definition, obtain the frequent itemset as set of items whose support score is at least the min_sup.
2. *Generate strong association rules:* By definition, obtain rules for the item sets obtained in step 1, with support score as min_sup and confidence score as min_conf. These rules are called as *strong rules*.

Challenge in obtaining association rules:

Low threshold value: If minimum threshold support (min_sup) is set quite low, then many items can belong to the frequent itemset.

Solution to the problem is to define a *closed frequent itemset* and *maximal frequent itemset*.

1. *Closed Frequent Itemset:* A frequent itemset A is said to be a closed frequent itemset if it is closed and has support score at least equal to min_sup. An itemset is said to be closed in a data set T if there does not exist any superset B such that support(B) equals support (A).
2. *Maximal Frequent Itemset:* A frequent itemset A is said to be a maximal frequent itemset in T if A is frequent and there exists no super set B such that A⊂B and B is frequent in T.

We use Venn diagram to understand the concept as shown in Figure 2.

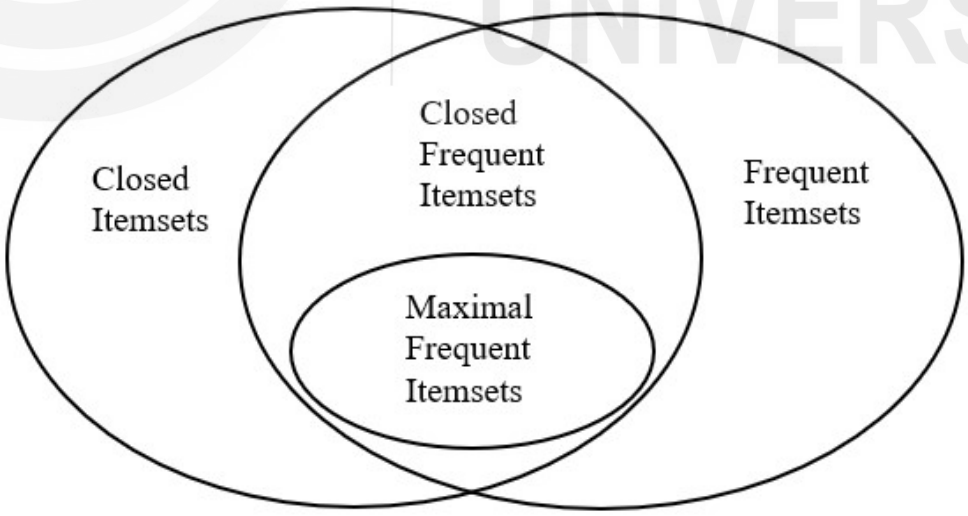


Figure 2: Venn Diagram to demonstrate the relationship between Closed item sets, frequent item sets, closed frequent item sets and maximal frequent item sets.

This illustrates that all the maximal frequent item sets are subsets of closed items sets and frequent item sets.

Check Your Progress 1

Ques 1: Given the following set of transactions for three items X, Y and Z as {{X, Y}, {X,Y, Z}, {X, Z}, {X, Z}, {Z}, {Z}, {Z}, {Z}}.

a) Give the binary representation of the transaction set.

b) Find the support and confidence for the following rules:

i) X =>Y

ii) X =>Z

iii) Y =>Z

Ques 2: Find the maximal frequent itemsets and closed frequent itemsets from the following set of transactions:

T1:A,B,C,E
T2:A,C,D,E
T3: B,C,E
T4:A,C,D,E
T5:C,D,E
T6:A,D,E

14.4 APRIORI ALGORITHM

R Aggarwal and R.Srikant proposed Apriori algorithm in the year 1994. The algorithm is used to obtain the frequent item sets for association rules. The algorithm is names so, as it needs the prior knowledge of the frequent item sets. This section discusses about the generation of frequent patterns as observed in the analysis of market basket problem. Section 1.4.1 presents Apriori algorithm, used to obtain the frequent item sets. Section 1.4.2 talks about generating strong association rules from the frequent item sets generated. Finally, section 1.4.3 presents variations of Apriori algorithm.

14.4.1 Frequent Item sets Generation

For a given set of *n* items, there are 2^n-1 possible combination of items. Consider an itemset I = {A, B, C, D} with four items, there are 15 combinations of items such as {{A}, {B}, {C}, {D}, {A, B}, {A, C}, {A, D}, {B, C}, {B, D}, {C, D}, {A, B, C}, {A, B, D}, {A, C, D}, {B, C, D}, {A, B, C, D}}. This can be represented by a lattice diagram as shown in Figure 3.

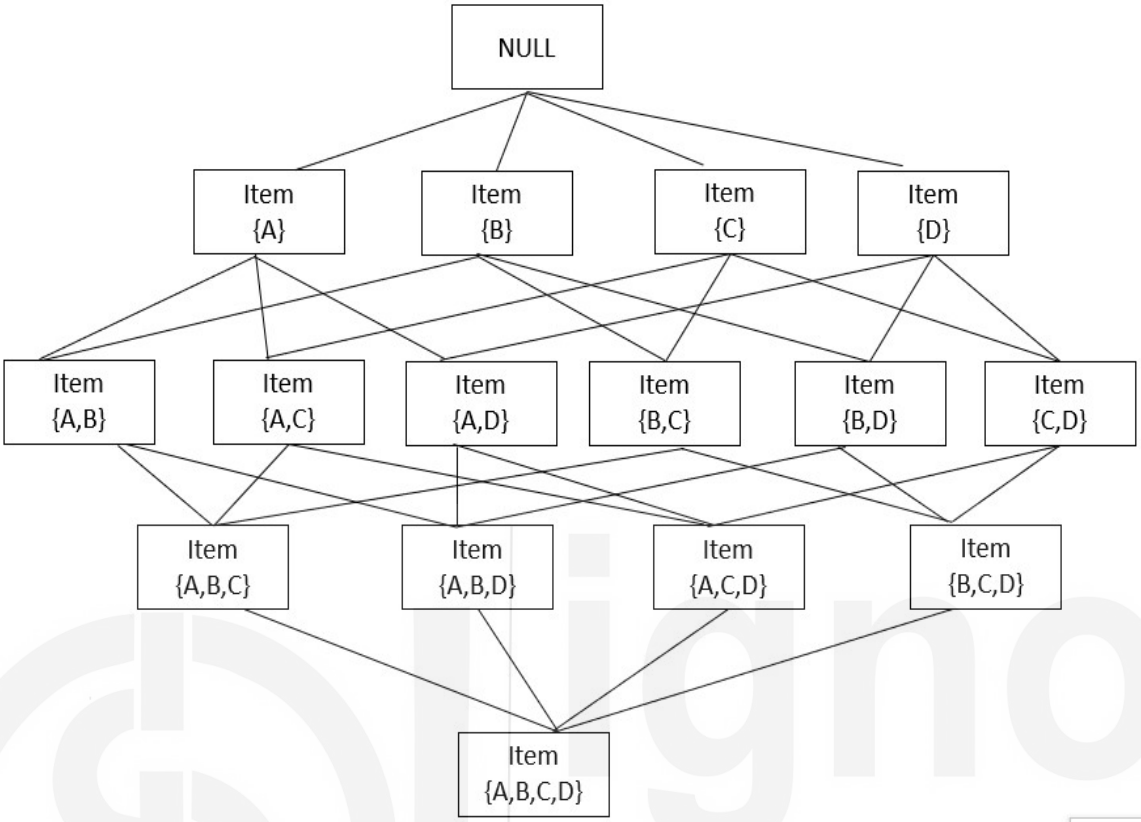


Figure 3: Lattice representing 15 combinations of items

Apriori algorithm searches the items level by level; to find the *(k+1)*item sets, it uses the *k*item sets. To determine the frequent item sets, the algorithm first finds the *candidate* item sets from the lattice representation. But as explained, for a given item sets of size *n*, maximum number of item sets in the lattice can be 2^n -1, one needs to control the search space in exponentially growing item sets and increase the efficiency of the algorithm. For this, two important principles are given below.

Definition 1: Apriori Principle: If an itemset is frequent, then all of its subsets must be frequent.

To understand the principle, let's consider the itemset {A, B, C} is the frequent itemset, then its *subsets* {A, B}, {A, C}, {B, C}, {A}, {B} and {C} are also the frequent item sets (marked in yellow) as shown in Figure 4. This is because if a transaction contains the itemset {A, B, C}, then the transaction will also contain all the subsets of this itemset.

On the other hand, if an itemset say {B, D} is not a frequent itemset, then all the *supersets* containing {B, D} i.e {A, B, D}, {B, C, D} and {A, B, C, D} are also not the frequent item sets (marked in blue) as shown in Figure 4. This is because if an itemset *X* is not a frequent itemset, then any other itemset containing *X* will also not be a frequent itemset. Recalling the definition that an itemset is frequent if and only if its support is greater than or equals to the minimum support threshold. Conversely, an itemset who support is less than the minimum support threshold, then the itemset is infrequent.

This *Apriori property* helps in pruning the item sets, thus reducing the search space and also increases the efficiency of the algorithm. This type of pruning technique is known as *support-based pruning*.

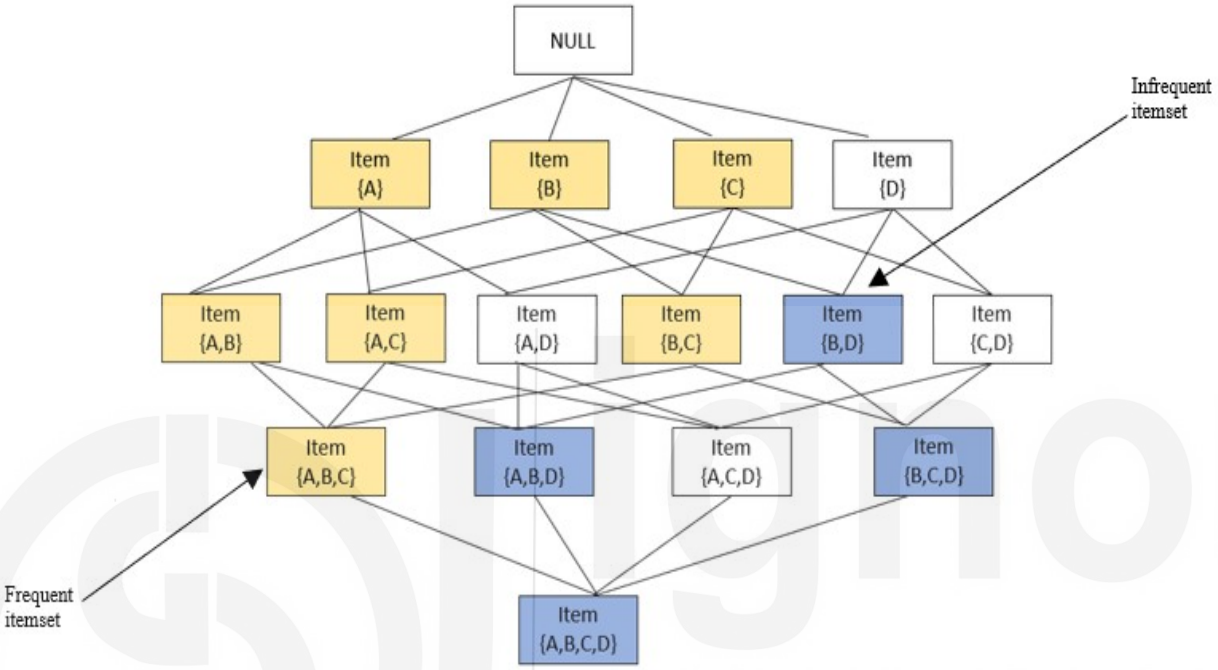


Figure 4: Frequent (yellow nodes) and Infrequent Item sets (blue nodes)

Definition 2: Antimonotone Property: This property states that if an itemset *X* does not satisfy a function, then any proper set containing *X* will also not satisfy the function. This property is called as antimonotone property as it is monotonic in terms of not satisfying a function. This property further helps in reducing the search space.

Given the set of transactions with the minimum support and confidence scores, perform the following steps to generate the frequent item sets using for Apriori algorithm.
Let *C(k)* be the set of *k* candidate item, *F(k)* be the set of *k* frequent items.

Step 1: Find F (1) i.e., frequent 1- item sets by computing the support for each item in the set of transactions.

Step 2: This is a twofold step as described below:

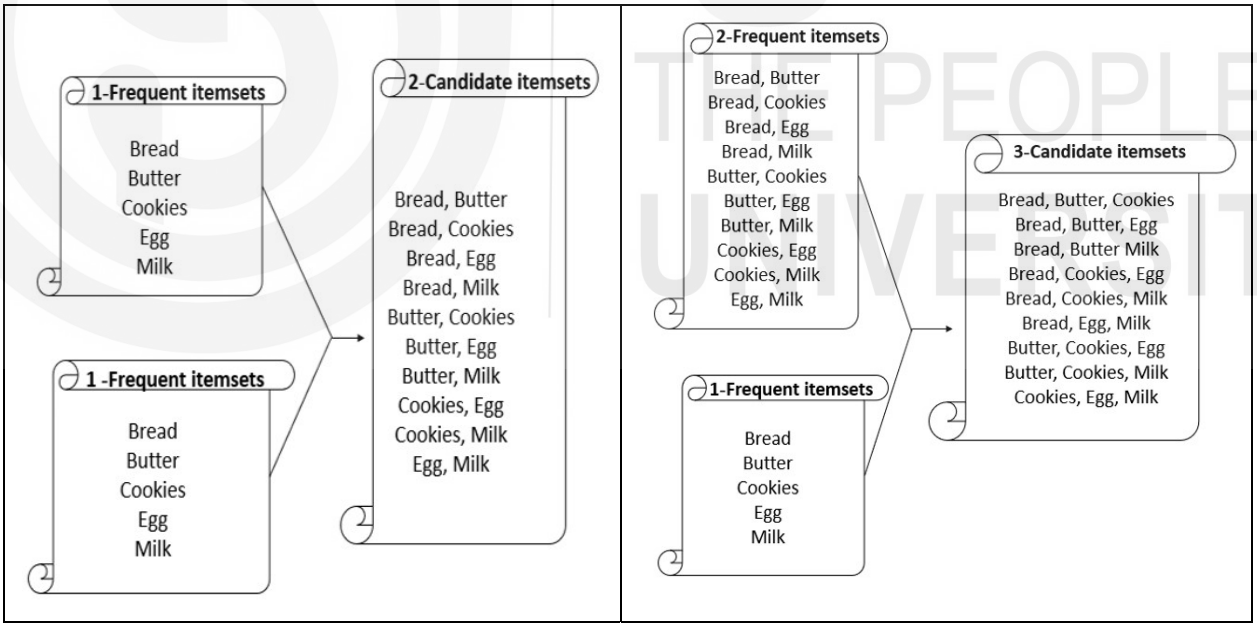
i) **Join Operation:** For $k \geq 2$, it generates C(k), new candidate k- itemset based on frequent item sets obtained in the previous step. This can be done in one of the two

ways:

1. Compute F(k-1) * F(1) to extend each F(k-1) itemset by joining it with each F(1) itemset. This join operation results in C(k) candidate item sets.

   For e.g.: Let k=2, and three 1 item setsF (1) = {A}, F (1) = {B} and F (1) = {C}. The two 1-itemsets can be augmented together to obtain 2-itemset such as F (2) = {A, B}, F (2)= {A, C} and F(2) = {B, C}. Further, to obtain 3-itemsets, we augment 2-itemsets with 1-itemsets such as joining {A, B} with {C} to obtain {A, B, C}.

   However, this method may generate duplicate F(k) item sets. For example, augmenting F(2) = {A, B} and F(1) = {C} generates C(3) = {A, B, C}. Also augmenting F(2) = {A, C} with F(1) = {B} generate C(3) = {A, B, C}. One way to avoid the generation of duplicate item sets is to sort the items in the item sets in lexicographic order. For example, the item sets {A, B}, {B, C}, {A, C}, {A, B, C} are sorted in their lexicographic order but {C, B}, {A, C, B} are not. Thus, items {A, B} , {C, D} can be augmented but the item sets {A, B}, {A, C} cannot be augmented as it will violate the lexicographic order. A working example to generate candidate F(k) itemset using this approach is shown in Figure 5.



a) F(1) joins F(1) to give C(2) *(b) F(2) joins F(1) to give C(3)*

Figure 5: Example of Augmenting F(k−1) and F(1) to generate candidate F(k) itemset

2. F(k-1) * F(k-1): A distinct pair of (k-1) item sets X = $\{A_1, A_2, A_3 \ldots A_{k-1}\}$ and Y = $\{B_1, B_2, \ldots B_{k-1}$, sorted in lexicographic order, are augmented iff:

$X_i = Y_i$, i= 1, 2,….(k-2) but $A_{k-1} \neq B_{k-1}$.

The item sets X and Y would be augmented only if the items A(k-1) and B(k-1) are sorted in lexicographic order. For example, let frequent itemset X(3)= {A, B, C} and Y(3) = {A, B, E}. X is joined with Y to generate the candidate itemset C(4) ={A, B, C, E}. On the other hand, consider another frequent itemset Z = {A, B, D}. Itemset X and Z can be joined to generate candidate set C(4) ={A, B, C, E} but itemset Y and Z cannot be joined as their last items, E and D are not arranged in lexicographic order. As a result, the F(*k*−1) × F(*k*−1) candidate \generation method merges a frequent itemset onlywith ones that contains the items in the sorted list, thus saving some computations. Working example demonstrating candidate generation using F(*k*−1) × F(*k*−1) is demonstrated in Figure 6.

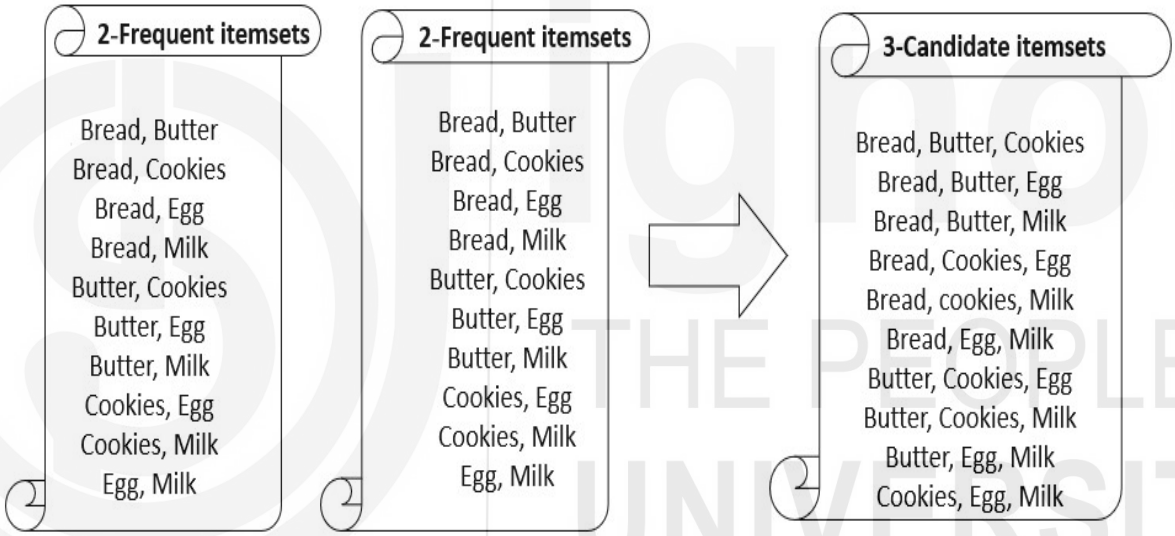


Figure 6: Example of joining F(k−1), F(k−1) to generate candidate F(k) itemset

ii) **Prune Operation:** Scan the generated entire candidate itemsets C(k) to compute the support score of each item in the set. Any itemset with support score less than minimum support threshold is pruned from the candidate itemset. Thus, the itemsets left after pruning would form the Frequent itemset F(K) such that F(k) ⊆ C(k). To prune the itemsets from the dataset, a priori property is used. According to the property, an infrequent (*k*-1)-itemset is not a subset of a frequent *k*-itemset. Thus, if any (*k* -1)-subset of a candidate itemset C(k) is not in *F(k*-1*)*, then the candidate cannot be frequent and so is removed from *C(k)*.

14.4.2 Case Study: *Find the frequent itemset from the given set of items*

Consider the set of transactions represented in binary form in Table 1 as given below. Assume minimum support threshold to be 2.

Transaction Id	List of items
T1	Milk, Cookies, Bread
T2	Milk, Bread, Egg, Butter
T3	Milk, Cookies, Bread, Butter
T4	Cookies, Bread, Butter

Step 1: Arrange the items in lexicographic order. Call this candidate set C(1)

Transaction Id	List of items
T1	Bread, Cookies, Milk
T2	Bread, Butter, Egg, Milk
T3	Bread, Butter, Cookies, Milk
T4	Bread, Butter, Cookies

Step 2: Obtain support score for each item in candidate itemset C(1) as:

S.No	Item	Support
1	Bread	4
2	Butter	3
3	Cookies	3
4	Egg	1
5	Milk	3

Step 3: Prune the items whose support score is less than the minimum support threshold. This results in 1-frequent itemset, F(1).

S.No	Item	Support
1	Bread	4
2	Butter	3
3	Cookies	3
4	Milk	3

Step 4: Generate 2-candidate itemsets from F(1) obtained in previous step and obtain support score of each itemset i.e frequency of each itemset in the original transaction set.

As support score of each itemset is at least 2, hence, none of the itemset is pruned. So, the table above represents 2-candidate itemsets, F(2).

S.No	Itemset	Support
1	Bread, Butter	3
2	Bread, Cookies	3
3	Bread, Milk	3
4	Butter, Cookies	2
5	Butter, Milk	2
6	Cookies, Milk	2

Step 5: Generate 3-candidate itemsets by joining F(2) and F(1) to obtain C(3). Compute the support score of each itemset.

S.No	Itemsets	Count
1	Bread, Butter, Cookies	2
2	Bread, Butter, Milk	2
3	Bread, Cookies, Milk	2
4	Butter, Cookies, Milk	1

Step 6: Prune the itemsets in C(3) if their support is less than the minimum support threshold. We then obtain F(3).

S.No	Itemsets	Count
1	Bread, Butter, Cookies	2
2	Bread, Butter, Milk	2
3	Bread, Cookies, Milk	2

Step 7: Similarly we obtain C(4) using F(3) and prune the candidate itemset to obtain frequent itemset F(4).

S.No	Itemset	Support
1	Bread, Butter, Cookies, Milk	1

As the only itemset in C(4) has support count 1, so it is pruned and F(4) =Ø.

Now, the iteration stops.

Detail of the algorithm is given in Figure 7.

Input: *T:* a set of transactions; *minsup:* minimum support threshold
Output: *F(k):* set of k-frequent itemsets (result)

```
1.  F1 ← {large 1 - itemsets}
2.     k ← 2
3.  while F(k−1) is not empty
4.        C(k)← Apriori_gen(F(k−1), k)
5.  for transactions t in T
6.           Dt ← {c in C(k) : c ⊆ t}
7.  for candidates c in Dt
8.                count[c] ← count[c] + 1
9.
10.       F(k) ← {c in C(k): count[c] ≥ minsup}
11.       k ← k + 1
12.
13. return Union(F(k))
14.
15. Apriori_gen(F, k)
16. for all X ⊆ F, Y ⊆ F where X1 = Y1, X2 = Y2, ..., Xk-2 = Yk-2 and Xk-1,
              Yk-1 are in lexicographic order
17.        c = X ∪ {Yk-1}
18. if u ⊆ c for all u in C
19.           result ←append (result, c)
20. return result
```

Figure 7: Apriori Algorithm

14.4.3 Generating Association Rules using Frequent Itemset

Once the frequent itemsets F(k) are generated from set of transactions T, next step is to generate strong association rules from them. Recall that *strong* association rules satisfy both minimum support as well as minimum confidence. Theoretically, confidence is defined as the ratio of frequency of transactions containing items x and *y* to the frequency of transactions that contained item *x*.

Based on the definition. Follow the given rules to generate strong association rules:

1. For each itemset f ϵ F(k), generate subsets of f.

2. For every non-empty subset $x \epsilon f$, generate the rule $x => f - x$

$$\frac{support(f)}{support(x)} \geq min\_conf$$

Consider the set of transactions and the generated frequent itemsets described in section 1.3.2. One of the frequent itemset *f* belonging to set F(3) is:

f = {Bread, Butter, Cookies}.

Following the steps given above, the non-empty subsets *x ⊆F(3)*,

x= {{Bread}, {Butter}, {Cookies}, {Bread, Butter}, {Bread, Cookies}, {Butter, Cookies}}

where all itemsets are sorted in lexicographic order. The generated association rules with their confidence scores are stated:

Rule	Confidence
{Bread, Butter} => {Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread,\ Butter)} = \frac{2}{3} = 66.67\%$
{Bread, Cookies} => {Butter}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread,\ Cookies)} = \frac{2}{3} = 66.67\%$
{Butter, Cookies} => {Bread}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Butter,\ Cookies)} = \frac{2}{2} = 100\%$
{Butter} => {Bread, Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Butter)} = \frac{2}{3} = 66.67\%$
{Bread} => {Butter, Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread)} = \frac{2}{4} = 50\%$
{Cookies} => {Bread, Butter}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Cookies)} = \frac{2}{3} = 66.67\%$

Depending on the minimum confidence scores, the obtained association rules are preserved, and rest are pruned.

Check Your Progress 2

Ques 1: For the following given Transaction Data-set, Generate Rules using Apriori Algorithm. Consider the values as Support=50% and Confidence=75%

Transaction Id	Set of Items
T1	Pen, Notebook, Pencil, Colours
T2	Pen, Notebook, Colours
T3	Pen, Eraser, Scale
T4	Pen, Colours, Eraser
T5	Notebook, Colours, Eraser

Ques 2: For the following given Transaction Data-set, Generate Rules using Apriori Algorithm. Consider the values as Support=15% and Confidence=45%

Transaction Id	Set of Items
T1	A, B, C
T2	B, D
T3	B, E
T4	A, B, D
T5	A, E
T6	B, E
T7	A, E
T8	A, B, C, E
T9	A, B, E

14.5 FP TREE GROWTH

Although there had been significant reduction in the numb er of candidate itemsets, yet Apriori algorithm can be slow due to scanning of entire transaction set iteratively. So, *Frequent Pattern growth,* also called as *FP growth* method employs divide and conquer strategy to generate frequent itemsets.

Working of FP growth algorithm:

1. Create **Frequent Pattern Tree,** or **FP-tree** by compressing the transaction database. Along with preserving the information about the itemsets, the tree structure also retains the association among the itemsets.

2. Divide the transaction database into a set of *conditional databases*. where each associated with one frequent item or "pattern fragment," and examines each database separately.

3. For each "pattern fragment," examine its associated itemsets only. Therefore, this approach may substantially reduce the size of the itemsets to be searched, along with examining the "growth" of patterns.

Advantages of FP growth over Apriori algorithm:

1. Efficient than Apriori algorithm
2. No candidate itemset generation
3. Only two passes over the transaction set

14.5.1 FP Tree Construction

1. Obtain the 1−itemsets and their support score as computed in the first step of Apriori algorithm.
2. Sort the itemsets in descending order of their support score.
3. Create the root of tree as NULL.
4. Examine the first transaction in the set T, create the nodes of tree as the itemset with the maximum support score at the top, the next itemset with lower count and so on. At the end, the leaves of the tree will have the itemsets with the least support score.
5. Examine every transaction in the set, whose itemsets are also sorted in descending order of their support score. If any itemset of this transaction is already existing in the tree, then they share the same node, but count is incremented by 1. Else, a new node and branch will be created for a new itemset.
6. Divide the FP tree obtained at the end of step 5 into conditional FP trees. For this, examine the leaf (lowest) node of FP tree. The lowest node represents the frequency pattern of length 1. For each item, traverse the path in the FP Tree to create *conditional pattern base*. It is the path labels of al paths to any node of the given item in FP tree.
7. Conditional pattern base generates the frequent patterns. Thus, one conditional tree is generated for one frequent pattern. Perform this step recursively to generate all the conditional FP trees.

Consider the following transaction set T given in section 1.3.2

Transaction `Id`	**List of items**

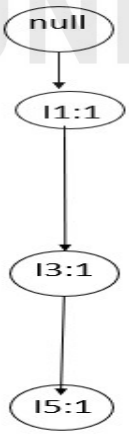
T1	Bread, Cookies, Milk
T2	Bread, Butter, Egg, Milk
T3	Bread, Butter, Cookies, Milk
T4	Bread, Butter, Cookies

We shall be representing each item with its S.No. in the description below. The items are sorted in descending order of their support score as given below.

S.No	Item	Support
I1	Bread	4
I2	Butter	3
I3	Cookies	3
I4	Milk	3
I5	Egg	1

i) Create the null node.

ii) For transaction T_1, item in descending order of their support score is I1 (4), I3 (3) and I4 (3). Create a branch connecting these three nodes in the tree as shown in Figure 8.



Examine T_1

Figure 8: Creating FP Tree from transaction T_1

iii) Examine the transaction T_2 andT_3, and further expand the FP tree as shown in Figure 9

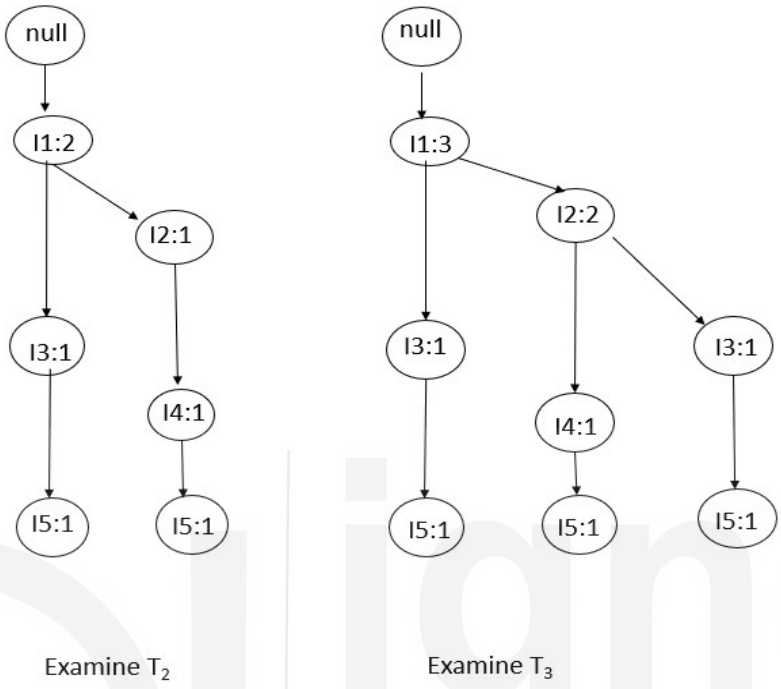


Figure 9: FP Tree creation from transactions T_2 and T_3

iv) Examining the last transaction T_4 generates the final FP tree as shown in Figure 10.

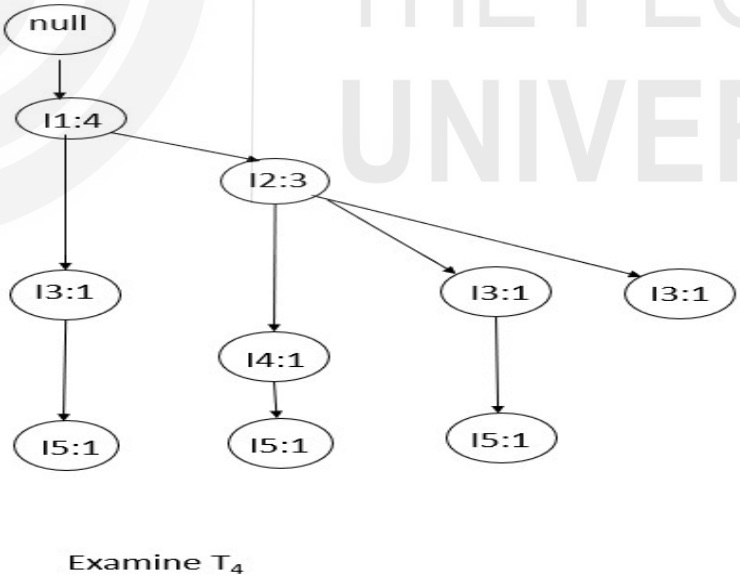


Figure 10: Final FP tree obtained after scanning transaction T_4

v) Now, obtain the conditional FP tree for each item I_1.....$I_5$by scanning the path from the lowest leaf node as shown in the table below:

Item	Conditional Pattern Base
I5	{I1, I3}:1, {I1, I2, I4}:1, {I1, I2, I3}:1
I4	{I1, I2}:1
I3	{I1):1, {I1, I2}:2
I2	{I1}:3
I1	

vi) For each item, build the conditional Pattern Tree by taking the set of elements common in all paths in the Conditional Pattern Base of that item. Further calculate its support score by summing the support scores of all the paths in the Conditional Pattern Base.

Item	Conditional Pattern Base	Conditional FP Tree
I5	{I1, I3}:1, {I1, I2, I4}:1, {I1, I2, I3}:1	{I1: 3, I2: 2}
I4	{I1, I2}:1	
I3	{I1):1, {I1, I2}:2	{I1}:3
I2	{I1}:3	{I1}:3
I1		

vii) Frequent patterns generated are: {I1, I2, I5} i.e {Bread, Butter, Milk}, {I1, I3} i.e. {Bread, Cookies} and {I1, I2}: {Bread Butter}.

Check Your Progress 3

Ques 1: Generate FP Tree for the following Transaction Dataset, with minimum support 40%.

Transaction Id	Set of Items
T1	I5, I1, I4, I2
T2	I4, I1, I3, I5, I2
T3	I3, I1, I2, I5

T4	I2, I1, I4
T5	I4
T6	I4, I2
T7	I1, I4, I5
T8	I2, I3

Ques 2: Find the frequent itemset using FP Tree, from the given set of Transaction Dataset with minimum support score as 2.

Transaction Id	Set of Items
T1	Banana, Apple, Tea
T2	Apple, Carrot
T3	Apple, Peas
T4	Banana, Apple, Carrot
T5	Banana, Peas
T6	Apple, Peas
T7	Banana, Peas
T8	Banana, Apple, Peas, Tea
T9	Banana, Apple, Peas

14.6 Pincer Search

In Apriori algorithm, the computation begins from the smallest set of frequent itemsets and proceeds till it reaches the maximum frequent itemset. But the algorithm's efficiency decreases as the algorithm passes through many iterations. Solution to the problem is to generate the frequent itemsets by using bottom-up and top-down approach together. Thus, the Pincer algorithm works on bidirectional approach. It attempts to find the frequent item sets in a bottom – up manner but, at the same time, it maintains a list of maximal frequent item sets. While iterating the transactions set, it computes the support score of the candidate maximal frequent item sets to record the frequent itemsets. This way, the algorithm outputs the subsets of the

frequent itemsets and, hence, they need not maintain support score in the next iteration. Thus, Pincer–search is advantageous over Apriori algorithm when the frequent item set is long.

In each iteration, while computing the support count of each iteration in bottom-up direction, the algorithm also computes the support score of some itemsets in top-down direction. These itemsets are called as *Maximal Frequent Candidate Set (MFCS)*. Consider an itemset x ϵ MFCS, such that cardinality of x is greater than k, in the k^{th} iteration. Then all the subsets of x must also be frequent. Thus, all these subset itemsets can be pruned from the candidate sets in bottom-up direction. Thus, increasing the algorithm efficiency. Similarly, when the algorithm finds an infrequent itemset in bottom–up direction, then MFCS is updated I order to remove these infrequent itemsets.

The MFCS is initialized to a singleton, which is the item set of cardinality *n* that contains all the elements of the transaction set. If some 'm' infrequent 1– itemsets are observed after the first iteration, then these infrequent itemsets will be pruned from the set, thereby reducing the cardinality of singleton to n –m. This is an efficient algorithm as one the algorithm may result a maximal frequent set in few iterations only.

Check Your Progress 4

Ques1: What is the advantage of using Princer algorithm over Apriori algorithm?

Ques 2: What is Maximal Frequent Candidate set?

Ques 3: What happens if an item is a maximal frequent item?

Ques 4: What is the bi–directional approach of Princer algorithm?

14.7 Summary

This unit presented the concepts of association rules. In this unit, we briefly described about what are these association rules and how various models help to analyse data for patterns, or co-occurrences, in a transaction database. These rules are created by finding the frequent itemsets in the database using the support and the confidence. Support indicates the frequency of occurrence of an item given the entire transaction database. High support score indicates a popular item among the users. Confidence of a rule indicates the rule's reliability. Higher confidence indicates more occurrence of item(s) in a set.
A typical example that has been used to study association rules is market basket analysis.

The unit presented different concepts like frequent itemsets, closed items and maximal itemsets.

Various algorithms viz, Apriori algorithm, FP tree and Pincer algorithm are discussed in detail in this unit, to generate the association rules by identifying the relationship between the items that are often purchased or occurred together.

14.8 Solutions to Check your progress

Check Your Progress 1

Ques 1

a) Binary Representation of the Transaction set:

Transaction	X	Y	Z
{X, Y}	1	1	0
{X, Y, Z}	1	1	1
{X, Z}	1	0	1
{Z}	0	0	1

b) Finding support and confidence of each rule:

Support = frequency of item/ number of transactions

Confidence(A->B) = support(A∪B)/support(B)

Support(X) = 4/8 = 50%
Support(Y) = 2/8 = 25%
Support(Z) = 7/8 = 87.5%
Support (X, Y) = 2/8 = 25%
Support (X, Z) = 3/8 = 37.5%
Support (Y, Z) = 1/8 = 12.5%

i) Confidence(X-> Y) = support (X, Y)/support(X) = (2/8)/(4/8) = 2/4 = 50%
ii) Confidence(X-> Z) = support (X, Z)/support(X) = (3/8)/(4/8) = 3/4 = 75%
iii) Confidence(Y-> Z) = support (Y, Z)/support(Y) = (1/8)/(2/8) = 1/2 = 50%

Ques 2 For finding the maximal frequent itemsets and the closed frequent itemsets:

a) Frequent itemset X ϵ F is maximal if it does not have any frequent supersets.
b) Frequent itemset X ϵ F is closed if it has no superset with the same frequency.

Step i) Count the occurrence of each item set:

Itemset	Frequency	Itemset	Frequency
{A}	4	{D, E}	3
{B}	2	{A, B, C}	1
{C}	5	{A, B, D}	0
{D}	4	{A, B, E}	1

{E}	6	{A, C, D}	2
{A, B}	1	{A, C, E}	3
{A, C}	3	{A, D, E}	3
{A, D}	3	{B, C, D}	0
{A, E}	4	{B, C, E}	2
{B, C}	2	{C, D, E}	3
{B, D}	0	{A, B, C, D}	0
{B, E}	2	{A, B, C, E}	1
{C, D}	3	{B, C, D, E}	0
{C, E}	5		

Let minimum support = 3.

a) {B}, {A, B}, {B, C}, {B, D}, {B, E}, {A, B, C}, {A, B, D}, {A, B, E}, {A, C, D}, {B, C, D}, {B, C, E}, {A, B, C, D}, {A, B, C, E} and {B, C, D, E} are not frequent itemsets as their support count is less than the minimum support score. Hence, we prune these itemsets.

b) {A} is not closed as frequency of {A, E} (superset of {A}) is same as frequency of {A} = 4.

c) {C} is not closed as frequency of {C, E} (superset of {C}) is same as frequency of {C} = 5.

d) {D} and {E} are closed as they do not have any superset with same frequency, but {D} and {E} are not maximal as they both have frequent supersets as {C, D} and {C, E} respectively.

e) {A, C} is not closed as frequency of its superset {A, C, E} is same as that of {A, C}.

f) {A, D} is not closed as frequency of its superset {A, D, E} is same as that of {A, D}.

g) {A, E} is closed as it has no superset with same frequency. But {A, E} is not maximal as it has a frequent superset i.e {A, D, E}.

h) {C, D} is not closed due to its superset {C, D, E}.

i) {C, E} is closed but not maximal due to its frequent superset {C, D, E}.

j) {D, E} is not closed due to its superset {C, D, E}.

k) {A, C, E} is a maximal itemset as it has no frequent super itemset. It is also closed as it has no superset with same frequency.

l) {A, D, E} is also a maximal and closed itemset.

Check Your Progress 2

Ques 1:

Step i) Find Frequent itemset and its support, where

support = frequency of item/number of transactions

Item	Frequency	Support %
Pen	4	4/5 = 80%
Notebook	3	3/5 = 60%
Pencil	1	1/5 = 20%
Colours	4	4/5 = 80%
Eraser	3	3/5 = 60%
Scale	1	1/5 = 20%

Step ii) Remove the items whose support is less than the minimum support (=50%)

Item	Frequency	Support %
Pen	4	4/5 = 80%
Notebook	3	3/5 = 60%
Colours	4	4/5 = 80%
Eraser	3	3/5 = 60%

Step iii) Form the two-item candidate set and find their frequency and support.

Item	Frequency	Support %
Pen, Notebook	2	2/5 = 40%
Pen, Colours	3	3/5 = 60%
Pen, Eraser	2	2/5 = 40%
Notebook, Colours	3	3/5 = 60%
Notebook, Eraser	1	1/5 = 20%
Colours, Eraser	2	2/5 = 40%

Step (iv) Remove the pair of items whose support is less than the minimum support

Item	Frequency	Support %
Pen, Colours	3	3/5 = 60%
Notebook, Colours	3	3/5 = 60%

Step (v) Generate the rule:

For rules, consider the following item pairs:

1. (Pen, Colours): Pen -> Colours and Colours -> Pen
2. (Notebook, colours): Notebook -> Colours and Colours -> Notebook

Confidence(A->B) = support (A∪ B)/support(A)

Hence,

Rule 1: Confidence (Pen -> Colours) = support (Pen, Colours)/support (Pen)
= (3/5) / (4/5) = 3/4 = 75%

Rule 2: Confidence (Colours -> Pen) = support (Colours, Pen)/support (Colours)
= (3/5) / (4/5) = 3/4 = 75%

Rule 3: Confidence (Notebook->Colours)
= support (Notebook, Colours) / support (Notebook)
= (3/5) / (3/5) = 1 = 100%

Rule 4: Confidence (Colours -> Notebook)
= support (Notebook, Colours) / support (Colours)
= (3/5) / (4/5) = 3/4 = 75%

All the 4 generated rules are accepted as the confidence score of each rule is greater than or equal to the minimum confidence given in the question.

Ques 2

Given set of transactions as:

Transaction Id	Set of Items
T1	A, B, C
T2	B, D
T3	B, E
T4	A, B, D
T5	A, E
T6	B, E
T7	A, E
T8	A, B, C, E
T9	A, B, E

Step i) Find the frequency and support of each item in the transaction set, where
support = frequency of item / number of transactions

Item	Frequency	Support
A	6	6/9 = 66.67%
B	7	7/9 = 77.78%
C	2	2/9 = 22.22%

D	2	2/9 = 22.22%
E	6	6/9 = 66.67%

Step ii) Prune the items whose support is less than the minimum support: As support of each item is greater than the minimum support (15%), hence, no item is pruned from the set. Now, form the two-item candidate set and find their support score.

Item	Frequency	Support
A, B	4	4/9 = 44.44%
A, C	2	2/9 = 22.22%
A, D	1	1/9 = 11.11%
A, E	3	3/9 = 33.33%
B, C	6	6/9 = 66.67%
B, D	2	2/9 = 22.22%
B, E	4	4/9 = 44.44%
C, D	0	0
C, E	1	1/9 = 11.11%
D, E	0	0

Step iii) Prune the two items whose support is less than the minimum support(15%)

Item	Frequency	Support
A, B	4	4/9 = 44.44%
A, C	2	2/9 = 22.22%
A, E	3	3/9 = 33.33%
B, C	6	6/9 = 66.67%
B, D	2	2/9 = 22.22%
B, E	4	4/9 = 44.44%

Step iv) Now, find three item frequent set and their support score

Item	Frequency	Support
A, B, C	2	2/9 = 22.22%
A, B, D	1	1/9 = 11.11%
A, B, E	1	1/9 = 11.11%
A, C, E	1	1/9 = 11.11%
B, C, D	0	0
B, C, E	1	1/9 = 11.11%
B, D, E	0	0

Pruning the three item sets whose support is less than the minimum support, we get:

Item	Frequency	Support
A, B, C	2	2/9 = 22.22%
A, B, E	2	2/9 = 22.22%

Step v) Generate the rules from each three itemset and compute the confidence of each rule as:

Confidence(x->y) = support(x∪y)/support(x)

For three itemset (A, B, C):

Rule	Support
(A, B)->C	(2/9) / (4/9) = 1/2 = 50%
(A, C) -> B	(2/9) / (2/9) = 1 = 100%
(B, C) -> A	(2/9) / (2/9) = 1 = 100%
A -> (B, C)	(2/9) / (6/9) = 2/6 = 33.33%
B -> (A, C)	(2/9) / (7/9) = 2/7 = 28.57%
C -> (A, B)	(2/9) / (2/9) = 1 = 100%

From the generated set of rules, the rules with confidence greater than or equal to the minimum confidence (45%) are the valid rules.

Thus, the valid rules are:

i) (A, B) -> C
ii) (B, C) -> A
iii) (A, C) -> B
iv) C -> (A, B)

Check Your Progress 3

Ques 1

Step i) From the given transaction set, find the frequency of each item and arrange them in decreasing order of their frequencies.

Item	Frequency
I1	5
I2	6
I3	3
I4	6
I5	4

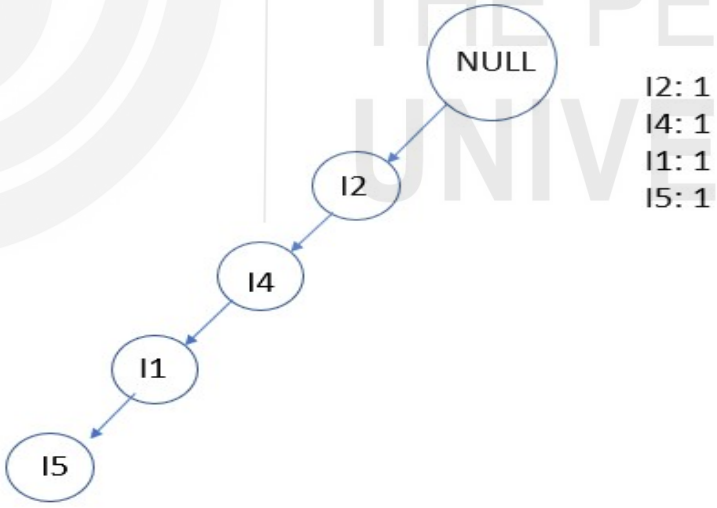
Item	Frequency
I2	6
I4	6
I1	5
I5	4
I3	3

Step ii) For every transaction set, arrange the item in the decreasing order of their frequency

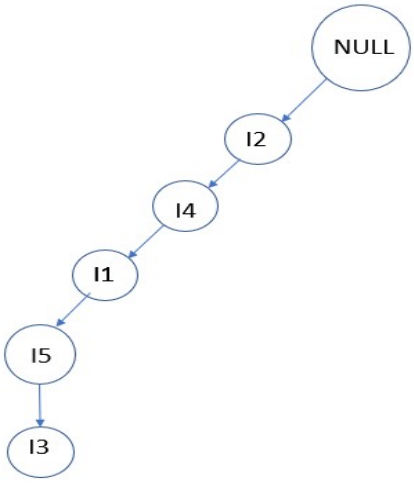
Transaction ID	Items	Ordered Items
T1	I5, I1, I4, I2	I2, I4, I1, I3
T2	I4, I1, I3, I5, I2	I2, I4, I1, I5, I3
T3	I3, I1, I2, I5	I2, I1, I5, I3
T4	I2, I1, I4	I2, I4, I1
T5	I4	I4
T6	I4, I2	I2, I4
T7	I1, I4, I5	I4, I1, I5
T8	I2, I3	I2, I3

Step iii) Create the FP tree now.

- Create a NULL node.
- From ordered items in T1, we get FP tree as:

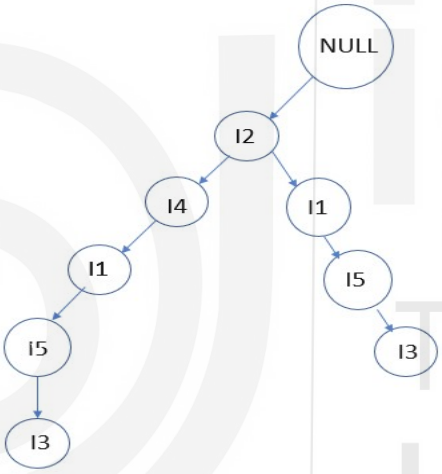


- Follow transaction T2 and increase the count of existing items and add the new items to the tree



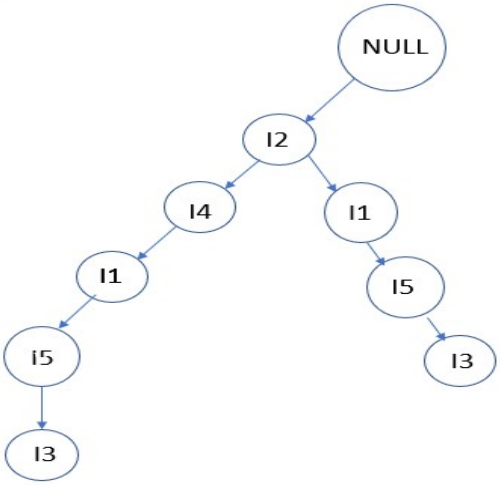
I2: 3
I4: 2
I1: 2
I5: 2
I3: 1

- Follow transaction T3 and repeat the above process



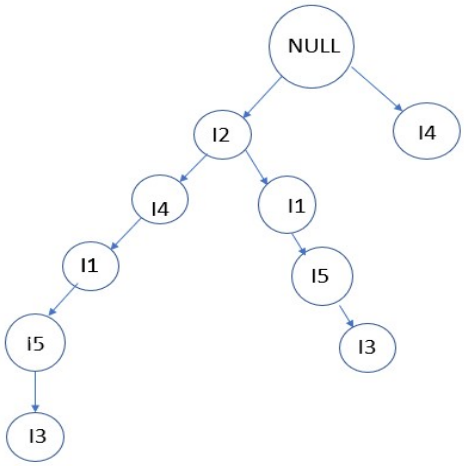
I2: 3
I4: 2
I1: 2
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1

- Follow Transaction 4, this leads to increase in count of I2, I4, I1.



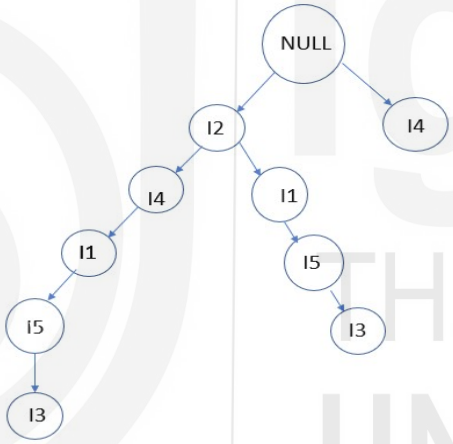
I2: 4
I4: 4
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1

- Follow transaction T5



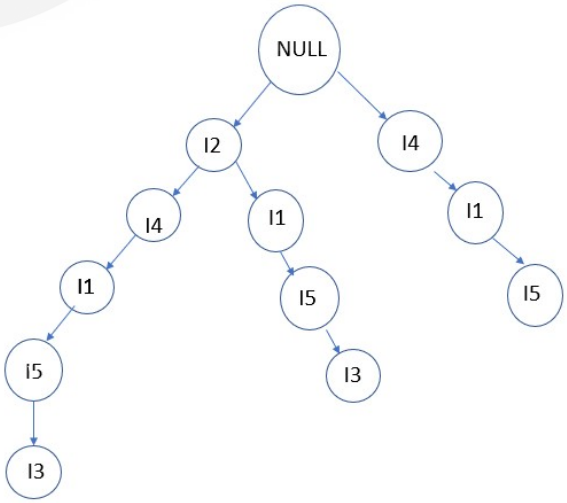
I2: 4
I4: 4
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 1

- Follow transaction T6. This leads to increase in count of I2 and I4 as the path in tree already exists.



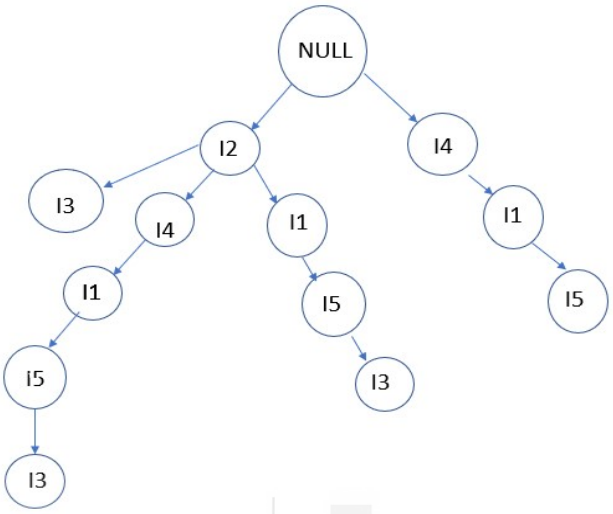
I2: 5
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 1

- Follow Transaction T7



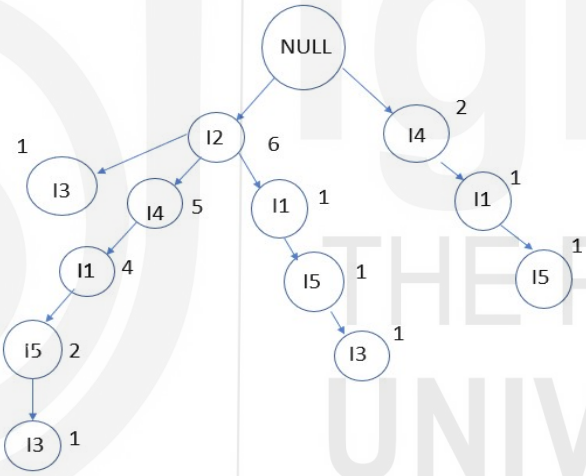
I2: 5
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1

- Finally follow transaction T8



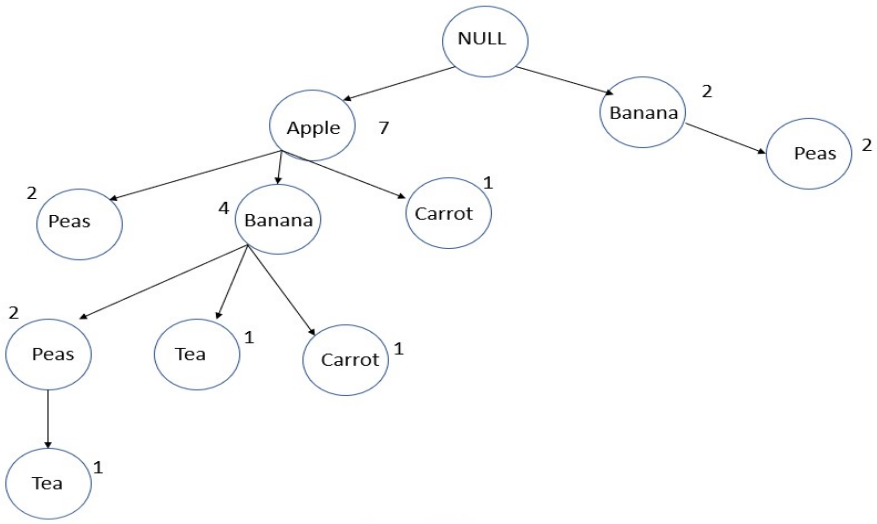
I2: 6
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1
I3: 1

The final FP tree is represented as:



I2: 6
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1
I3: 1

Ques 2 Step i) FP Tree for the given Transaction set is:



Step ii) Create the conditional Pattern base as:

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Generation
Tea	{{Apple, Banana:1}, {Apple, Banana, Peas: 1}}	{Apple: 2, Banana: 2}	{Apple,Tea:2},{Banana,Tea:2}, {Apple,Banana,Tea:2}
Carrot	{{Apple,Banana:1},{Apple:1}	{Apple:2}	{Apple,Carrot:2}
Peas	{{Apple,Banana:2}, {Apple:2},{Banana:2}}	{Apple:4,Banana:2}, {Banana:2}	{Apple, Peas:4},{Banana, Peas:4},{Apple,Banana,Peas:2}
Banana	{{Apple:4}}	{Apple:4}	{Apple,Banana:4}

As the minimum support given 3. Hence all the itemsets with support score greater than equal to 3 form the frequent itemset as:
{Apple, Banana}, {Apple, Peas}, and {Banana, Peas}

14.9 FURTHER READINGS

1. Machine learning an algorithm perspective, Stephen Marshland, 2nd Edition, CRC Press, 2015.
2. Machine Learning, Tom Mitchell, 1st Edition, McGraw- Hill, 1997.
3. Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Peter Flach, 1st Edition, Cambridge University Press, 2012.
4. Data Mining – Concepts and Techniques(3rd Edition) , Kamber and Han, Morgan Kaufman

UNIT 15 CLUSTERING

Structure

15.1 INTRODUCTION

Clustering or cluster analysis is a method for dividing a group of data objects into subgroups based on a single observation. Here, each cluster is a subset of the data, where objects with resemblance or similar properties are grouped. Also, they are similar to each other in one cluster but differ from those objects which are in different clusters. In simple terms, the effort of dividing a population into multiple groups so that data points of one group can easily be compared with data points of another group. Thus, to separate the groups with identical features and assign them into clusters. This process is performed by machines, not by humans and is known as unsupervised learning because clustering is a form of learning by observation. Clustering is often confused with classification in data analysis, where separation of data happens based on class labels, while in clustering partitioning of large data sets occurs in groups occurs based on similarity.

Let's understand this with an example. Suppose, you are the head of a rental store and wish to understand the preferences of your customers to scale up your business. Is it possible for you to look at the details of each customer and devise a unique business strategy for each one of them? Not. But, what you can do is to cluster all of your customers into say 10 groups based on their purchasing habits and use a separate strategy for customers in each of these 10 groups. And this is what we call **clustering.**

"Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data."

Data clustering is a prominent technique in data analysis and is applied in various research areas including mining of data, data statistics, area of machine learning for any kind of analysis. It is also applied in the world of financial services, health information systems web mining, financial sectors and many more. Cluster analysis is the most recent area of research in data analysis due

to the massive volumes of data produced in databases. An example of clustering is outlier detection where credit card fraud and criminal activities are monitored. Clustering can be used in image recognition to find clusters or patterns in image or text recognition systems. Clustering has a lot of uses in web search as well. Due to the enormous quantity of online pages, a keyword search may frequently produce a huge number of hits (i.e., pages relevant to the search). Thus, clustering is a very promising machine learning process and is proved to be one of the most pragmatic data mining tools. In this unit, you will learn about the various types of clustering techniques and algorithms.

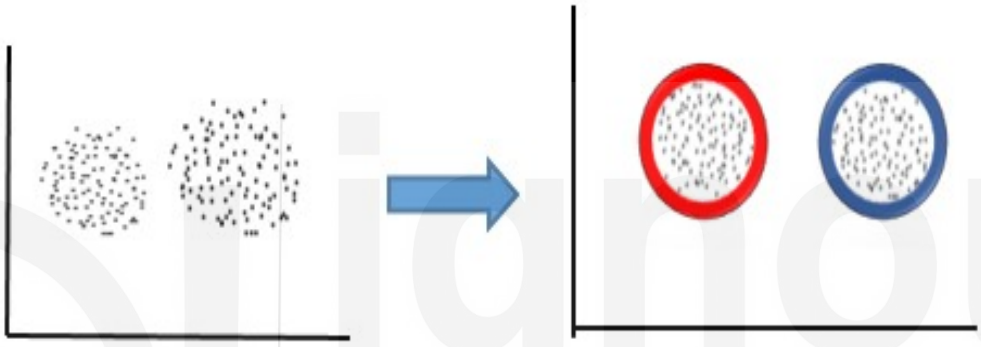


Fig 1. Clustering

Real World Example:

A bank can potentially have millions of customers. Would it make sense to look at the details of each customer separately and then decide? Certainly not! It is a manual process and will take a huge amount of time.

So, what can the bank do? Clustering comes to the rescue in these situations where the banks can group the customers based on their income, as shown:

Applications of Clustering in Real-World Scenarios

Clustering is a widely used technique in the industry. It is being used in almost every domain, ranging from banking to recommendation engines, document clustering to image segmentation.

- Customer Segmentation

Customer segmentation is one of the most common applications of clustering. And it isn't just limited to banking. This strategy is across functions, including telecom, e-commerce, sports, advertising, sales, etc.

- Document Clustering

Document Clustering is another common application of clustering. Let's assume that you have multiple documents, and you need to cluster similar documents together. Clustering helps us group these documents such that similar documents are in the same clusters.

- Image Segmentation

We can also use clustering to perform image segmentation. Here, we try to club similar pixels in the image together. We can apply clustering to create clusters having similar pixels in the same group.

15.2 Types of Clustering

There are many clustering algorithms in the literature. Traditionally, documents are grouped based on how similar they are to other documents. Similarity-based algorithms define a function for computing document similarity and use it as the basis for assigning documents to clusters. Each cluster should have data that are comparable to one another but different from those in other clusters. Clustering algorithms fall into different categories based on the underlying methodology of the algorithm (agglomerative or partition), the structure of the final solution (flat or hierarchical), or the density based All the above-mentioned clustering types are discussed in detail in the rest of this chapter.

Check Your Progress - 1

Qn1. What do you understand by the term "Clustering"?

Qn2. Where is Clustering used in present day scenario?

15.3 Partition Based Clustering

Partition algorithm is one of the most applied clustering algorithms. It has been widely used in many applications due to its simple structure and easy implementation as compared to other clustering algorithms. This clustering method classifies the information into multiple groups based on the characteristics and similarities of the data. In the partitioning method when database(D) contains multiple(N) objects then the partitioning approach divides the data into user-specified(K) partitions, each of which represents a cluster and a specific region. That is, the data is divided into K groups. That is, it divides the data into K groups or partitions in such a manner that each group should have at least one object from the existing data. To put it another way, it splits the data items into non-overlapping subsets (clusters) so that each data object fits perfectly into one of them.

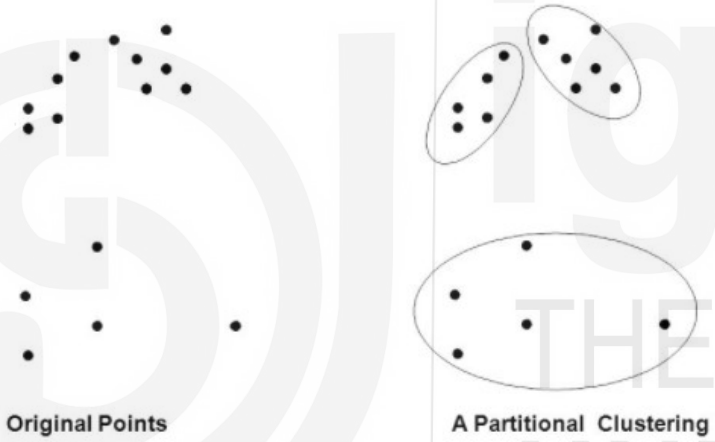


Fig 2. Partition based clustering

Partitioning approaches require a set of starting seeds (or clusters), which are then enhanced iteratively by transferring objects from one group to another to improve partitioning.Objects in the same cluster are "near" or related to one other, whereas objects in other clusters are "far apart" or significantly distinct, according to the most prevalent approach to good partitioning.

Many algorithmsthat come under partitioning method some of the popular ones are K-Mean, PAM (K-Mediods), and CLARA algorithm (Clustering Large Applications) etc.

CHECK YOUR PROGRESS

Qn1. Briefly explain how Partition Based Clustering works.

Qn2. Name three Partition Based Clustering methods.

15.4 Hierarchical BasedClustering

Hierarchical Clustering analysis is an algorithm that groups the data points with similar properties and these groups are termed "clusters". As a result of hierarchical clustering, we get a set of clusters, and these clusters are always different from each other. Clustering of this data into clusters is classified as:

- Agglomerative Clustering (involving decomposition of cluster using bottom-up strategy)
- Divisive Clustering (involving decomposition of cluster using top-down strategy)

Hierarchical clustering helps in creating clusters in the proper order (or hierarchy).

Example:

The most common everyday example we see is how we order our files and folders in our computer by proper hierarchy.

As mentioned, Hierarchical clustering is classified into two types i.e., Agglomerative clustering (AGNES) and Divisive Clustering (DIANA)

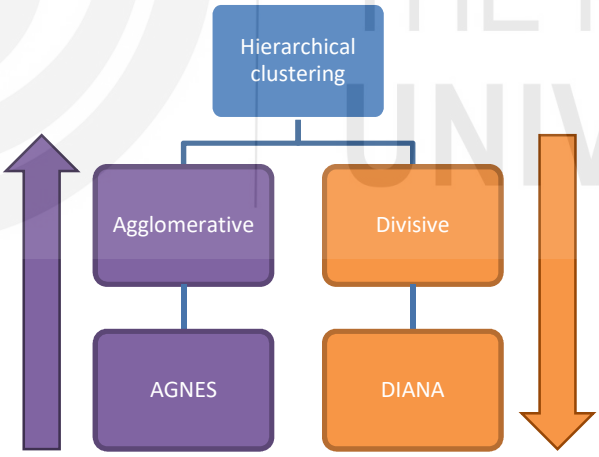


Fig3. Showing hierarchical clustering AGNES vs DIANA

Hierarchical clustering Technique in terms of space and time complexity:

- **Space complexity:** When the number of data points is large, the space required for the Hierarchical Clustering Technique is large since the similarity matrix must be stored in RAM. The space complexity is measured by the order of the square of n.

  Space complexity = $O(n^2)$ where n is the number of data points.

- **Time complexity:** The time complexity is also very high because we have to execute n iterations and update and restore the similarity matrix in each iteration. The order of the cube of n is the time complexity.

  Time complexity = $O(n^3)$ where n is the number of data points.

Limitations of HierarchicalClustering Technique:

1. Hierarchical clustering does not have a mathematical goal.
2. All the methodologies applied for calculating the similarity index between clusters does not apply fully in every situation, each technique has its own merits and demerits.
3. Due to high space and time complexity. This clustering algorithm is not applicable for huge data.

Check Your Progress - 2

Qn1. Differentiate between Hierarchical Clustering and Partition Based Clustering.

Qn2. What are sub-types of Hierarchical Clustering and what is the difference between them?

Qn3. What is the space and time complexity of Hierarchical clustering?

Qn4. List some of the limitations of Hierarchical clustering

15.5 Density Based Clustering Technique

Clusters are formed in the Density Depending Clustering Technique based on the density of the data points represented in the data space.Those locations that become dense as a result of the large amount of data points that reside there are termed as clusters.

How it works:

1. It starts with a random unvisited starting data point. All points within a distance 'Epsilon' – Ɛ classify as neighborhood points.

2. We need a minimum number of points within the neighborhood to start the clustering process. In this scenario, the current point of data turns into the first point in the cluster. Else, the point is regardedas 'Noise.' In either case, the current point becomes a visited point.

3. All points within the distance(Ɛ) become part of the same cluster. This process is repeated for all the newly added data points in the cluster group.

4. Continue with the process until you visit and label each point within the 'Ɛ' neighborhood of the cluster.

5. On completion of the process, start again with a new unvisited point thereby leading to the discovery of more clusters or noise. At the end of the process, you ensure that you mark each point as either cluster or noise.

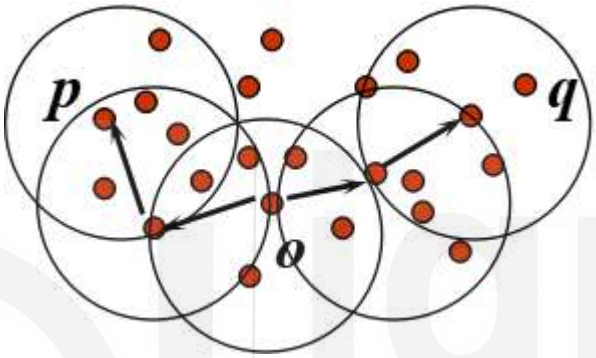


Fig 4. Showing Density connected points

Check Your Progress - 3

Qn1. What do you understand by the term "Density Based Clustering"?

Qn2. How is "Density Based Clustering" performed?

15.6 Clustering Algorithms

a) K-Means

Among clustering algorithms, is an algorithm that tries to minimize the distance of the points in a cluster with their *centroid* – the k-means clustering technique.

K-means is a centroid-based algorithm. It can also be called asa distance-based technique where distances between points are calculated to allocate a point to a cluster. Each cluster in K-Means is paired with a centroid.

The K-Means algorithm's main goal is to reduce the sum of distances between points and their corresponding cluster centroid.

Real World Example:

Let's have a look at an example. Assume you went to a bookstore to purchase some books. There are several types of books that can be found there. One thing you'll notice is that the books are sorted into groups based on their category. All of the literature books will be kept in one location, while science books will be organized by type. The K-Means algorithm's main goal is to reduce the sum of distances between points and their corresponding cluster centroid.

Now we will understand this with the help of these figures.

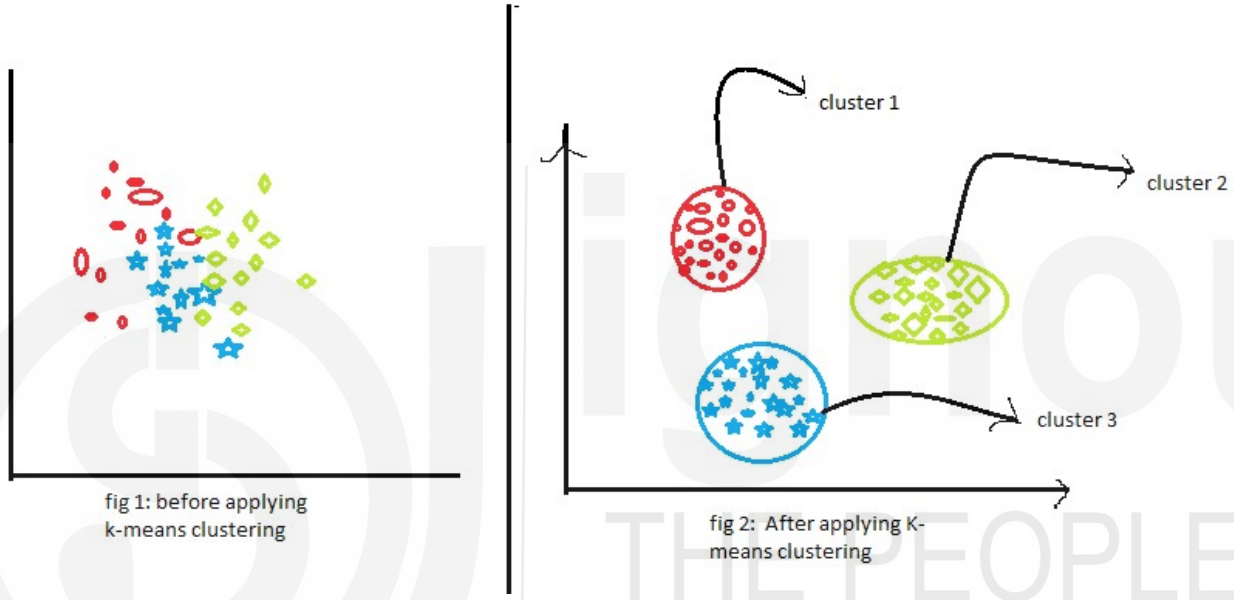


Fig 5. Before and after K-means clustering

In the figure 5 data is presented into two stages. The first figure shows the data in raw stage which is not clustered by k-means. Here all types of data are clubbed together thus becomes impossible to differentiate them into their original category.

The second figure shows three clusters of three different colors red, green, and blue. These clusters are formed after applying k-means clustering. The second figure shows data into three different categories which are called clusters.

Working of K-means clustering algorithm .

k-means clustering technique is an immense clustering algorithm to group similar types of data in groups which are so called clusters. It is the simplest and commonly used technique in machine learning extensively used for data analysis. It can easily locate the similarity points between the different data items and can group them into the clusters. The working of K-means clustering algorithm is shown in the following three steps. Let’s see what these three steps are.

1. Selection of k values.
2. Initialization of the centroids.
3. Selection of the cluster and finding the average.

Let us understand the above steps with the help of the Fig6:

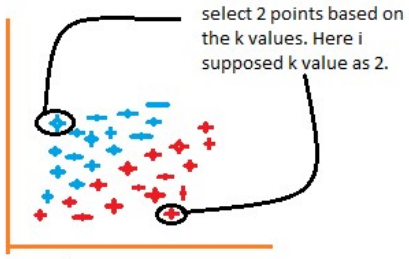


Figure 1

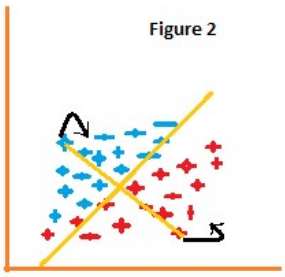
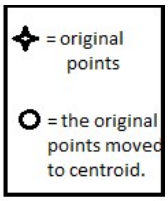
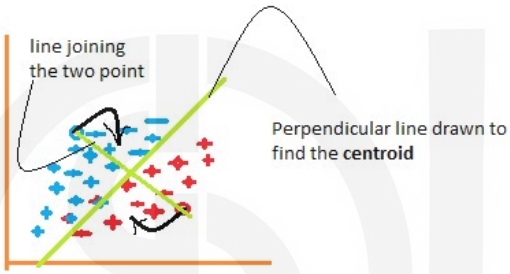


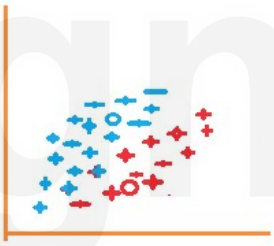
Figure 2

F2: Find the average of all the blue points and red points and move the selected points to **centroid.**





F3: Some of the red points changed to blue points, that means they belong to the group blue now. Again the repeat the same process.



F4: The same process has been applied here. This process will be continued until we get the **two complete different cluster.**

Below is a Practice Problem Based on K-Means Clustering Algorithm:

Problem-01: The following eight points (with (x, y) denoting places) should be grouped into three clusters:

A1(2, 11), A2(2, 15), A3(8, 5), A4(6, 8), A5(7, 9), A6(6, 3), A7(1, 4), A8(4, 8)

The first cluster centers will be: A1(2, 11), A4(6, 8) and A7(1,4).

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.

Solution: We follow the above discussed K-Means Clustering Algorithm-

Iteration-01:

- We measure the distance between each location and the center of each of the three clusters.
- The specified distance function is used to determine the distance.

The calculation of distance between point A1(2, 11) and each of the center of the three clusters-

Calculating Distance Between A1(2, 11) and C1(2, 11)-

P(A1, C1)

= |x2 – x1| + |y2 – y1|

= |2 – 2| + |11 – 11|

= 0

Calculating Distance Between A1(2, 10) and C2(6, 8)-

P (A1, C2)

= |x2 – x1| + |y2 – y1|

= |6 – 2| + |8 – 10|

= 4 + 2

= 6

Calculating Distance Between A1(2, 10) and C3(1, 4)-

P(A1, C3)

= |x2 – x1| + |y2 – y1|

= |1 – 2| + |4 – 10|

= 1 + 6

= 7

Problem-02 (Self-Test)

Use the k-means algorithm and Euclidean distance to cluster the following 8 examples into 3 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). The distance matrix based on the Euclidean distance is given below:

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

Assume that the first seeds (cluster centers) are A1, A4, and A7. Only use the k-means method for one epoch. Show: a) The new clusters (i.e., the cases belonging to each cluster) b) The new clusters' centers at the end of this epoch c) Draw a 10 by 10 space with all 8 points and illustrate the clusters and new centroids after the first epoch. d) How many more iterations will it take to reach convergence? For each period, draw the result.

b) Agglomerative clustering

In this case of clustering, the hierarchical decomposition is done with the help of bottom-up strategy where it starts by creating atomic (small) clusters by adding one data object at a time and then merges them together to form a big cluster at the end, where this cluster meets all the termination conditions. This procedure is iterative until all the data points are brought under one single big cluster.

Basic algorithm of agglomerative clustering

1. Determine the proximity matrix.

2. Assume that each data point belongs to a cluster.

3. Do it again.

4. Combine the two groups that are the closest together.

5. Make changes to the proximity matrix.

6. Continue until just one cluster remains.

AGNES (Agglomerative Nesting) is a type of agglomerative clustering that combines the data objects into a cluster based on similarity. The result of this algorithm is a tree-based structure called Dendrogram. Here it uses the distance metrics to decide which data points should be combined with which cluster. Basically, it constructs a distance matrix and checks for the pair of clusters with the smallest distance and combines then. The figure 7. given below shows dendrogram.

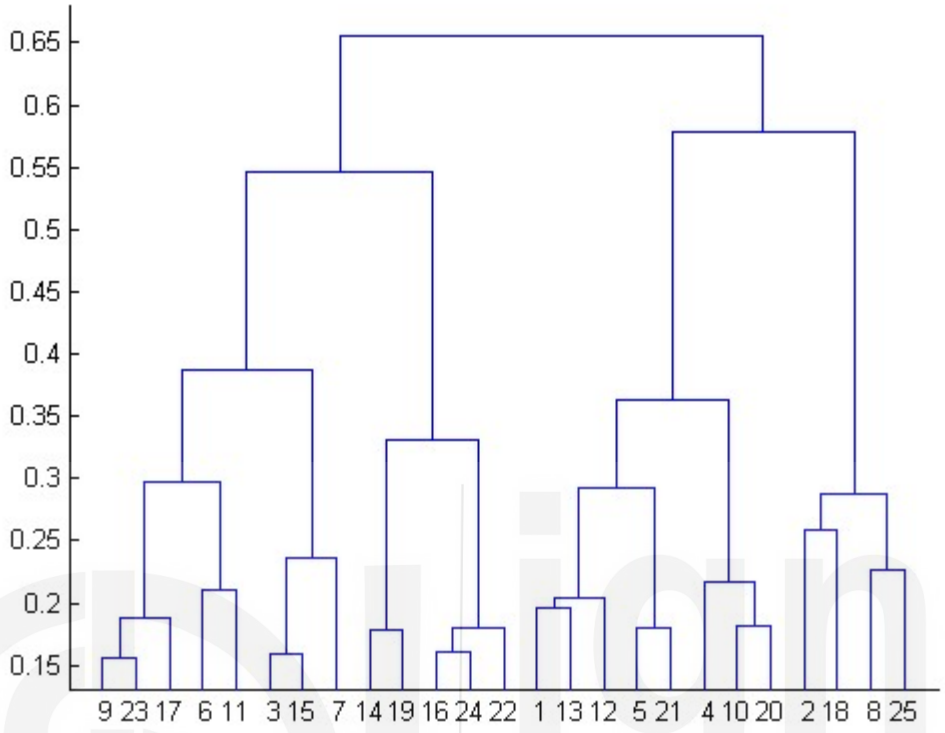


Fig 7. Agglomerative clustering

We begin at the bottom with 25 data points, each of which is assigned to a different cluster. Then two nearest clusters are selected to merge till we get only one cluster at the topmostposition. The distance between two clusters in the data space is represented by the height in the dendrogram at which two clusters are merged. Based on how the distance between each cluster is measured, we can have 3 different methods

- **Single linkage:** Where the shortest distance between the two points in each cluster is defined as the distance between the clusters.
- **Complete linkage:** In this case, we will consider the longest distance between each cluster's points as the distance between the clusters.

- **Average linkage:** In this situation, we'll take the average of each point in one cluster compared to every other point in the other.

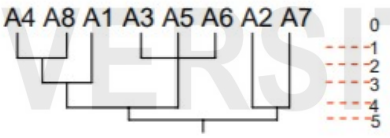
AGNES has few limitations one of this is it has a time complexity of at least **O(n2)**; hence it doesn't do well in scaling, and one other major drawback is that whatever has been done can never be undone, i.e. If we incorrectly group any cluster in an earlier stage of the algorithm, then we will not be able to change the outcome/modify it. But this algorithm has a bright side since there are many smaller clusters are formed; it can be helpful in the process of discovery. It produces an ordering of objects that is very helpful in visualization.

Problem-03: Hierarchical clustering (to be done at your own time, not in class) Use single-link, complete-link, average-link agglomerative clustering as well as medoid and centroid to cluster the following 8 examples: A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9). The distance matrix is the same as the one in Exercise 1. Show the dendrograms.

Solution:

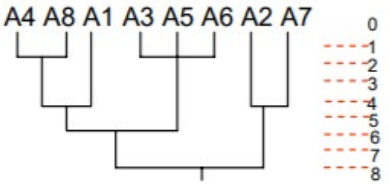
Single Link:

d	k	K
0	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
1	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
2	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
3	4	{A4, A8, A1}, {A3, A5, A6}, {A2}, {A7}
4	2	{A1, A3, A4, A5, A6, A8}, {A2, A7}
5	1	{A1, A3, A4, A5, A6, A8, A2, A7}



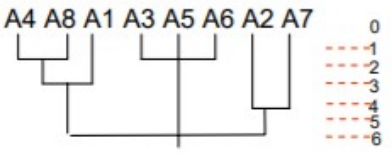
Complete Link

d	k	K
0	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
1	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
2	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
3	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
4	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
5	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
6	2	{A4, A8, A1, A3, A5, A6}, {A2, A7}
7	2	{A4, A8, A1, A3, A5, A6}, {A2, A7}
8	1	{A4, A8, A1, A3, A5, A6, A2, A7}



Average Link

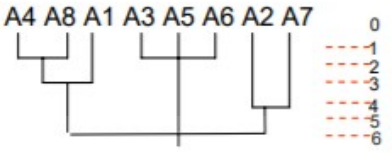
d	k	K
0	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
1	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
2	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
3	4	{A4, A8, A1}, {A3, A5, A6}, {A2}, {A7}
4	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
5	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
6	1	{A4, A8, A1, A3, A5, A6, A2, A7}



Average distance from {A3, A5, A6} to {A1, A4, A8} is 5.53 and is 5.75 to {A2, A7}

Centroid

D	k	K
0	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
1	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
2	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
3	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
4	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
5	3	{A4, A8, A1}, {A3, A5, A6}, {A2, A7}
6	1	{A4, A8, A1, A3, A5, A6, A2, A7}

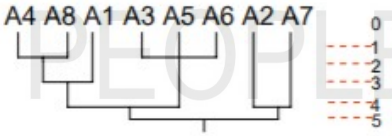


Centroid of {A4, A8} is B=(4.5, 8.5) and centroid of {A3, A5, A6} is C=(7, 4.33)
distance(A1, B) = 2.91 Centroid of {A1, A4, A8} is D=(3.66, 9) and of {A2, A7} is E=(1.5, 3.5)
distance(D,C)= 5.74 distance(D,E)= 5.90

Medoid

This is not deterministic. It can be different depending upon which medoid in a cluster we chose.

d	k	K
0	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
1	8	{A1}, {A2}, {A3}, {A4}, {A5}, {A6}, {A7}, {A8}
2	5	{A4, A8}, {A1}, {A3, A5, A6}, {A2}, {A7}
3	4	{A4, A8, A1}, {A3, A5, A6}, {A2}, {A7}
4	2	{A1, A3, A4, A5, A6, A8}, {A2, A7}
5	1	{A1, A3, A4, A5, A6, A8, A2, A7}



c) Divisive Clustering (DIANA)

Diana basically stands for Divisive Analysis; this is another type of hierarchical clustering where basically it works on the principle of top-down approach (inverse of **AGNES**) where the algorithm begins by forming a big cluster, and it recursively divides the most dissimilar cluster into two, and it goes on until we're all the similar data points belong in their respective clusters. These divisive algorithms result in highly accurate hierarchies than the agglomerative approach, but they are computationally expensive.

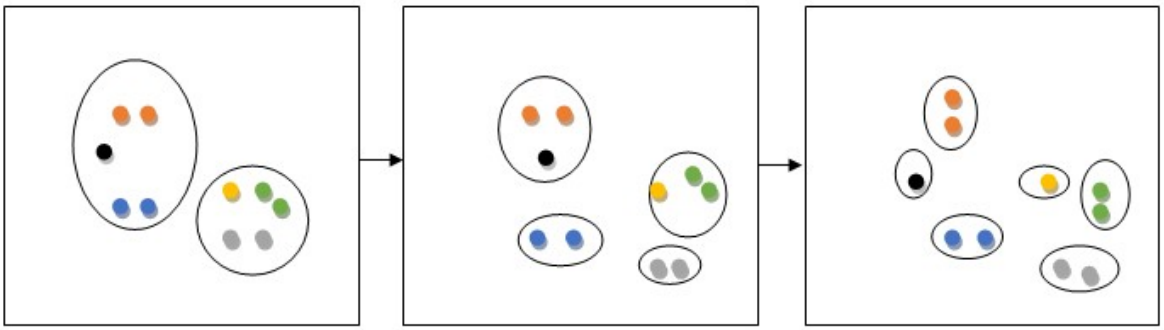


Fig 8. Divisive clustering step by step process

d) Density-Based Spatial Scan (DBSCAN)

The distance between objects is used to cluster things in most partitioning methods. Such algorithms can only locate spherical-shaped clusters and have trouble finding clusters of other forms. DBSCAN can form clusters in different shapes; this type of algorithm is most suitable when the dataset contains noise or outliers. Also, it depends on a density-based concept of cluster: A cluster is defined as the most densely connected set of points.

Other clustering algorithms based on the concept of density have been developed. Their basic concept is to keep creating a cluster till the density which consists of data points in the "neighbourhood" surpasses a certain threshold.

For instance, every data point present in a given cluster should meet the requirement of having minimum number of points in the neighborhood of a given radius. This type of method is extremely useful for detecting outliers or to filter out noise. This is also useful for finding clusters of arbitrary shape.

The best part in density-based methods is that they can divide a set of objects into multiple exclusive clusters, or a hierarchy of clusters. Density-based techniques often only evaluate exclusive clusters and ignore fuzzy clusters. Furthermore, density-based clustering algorithms can be extended from entire space to subspace.

DBSCAN uses noise to find clusters of any shape in spatial databases.

In density-based clustering we partition points into dense regions separated by not-so-dense regions. Characterization of points is done in following manner:

If a point has more than a given number of points (MinPts) within Eps, it is considered a core point.

▪ These points belong in a dense region and are at the interior of a cluster.

Within Eps, a border point has less points than MinPts, but it is close to a core point.

Any point that isn't a core point or a boundary point is referred to as a noise point.

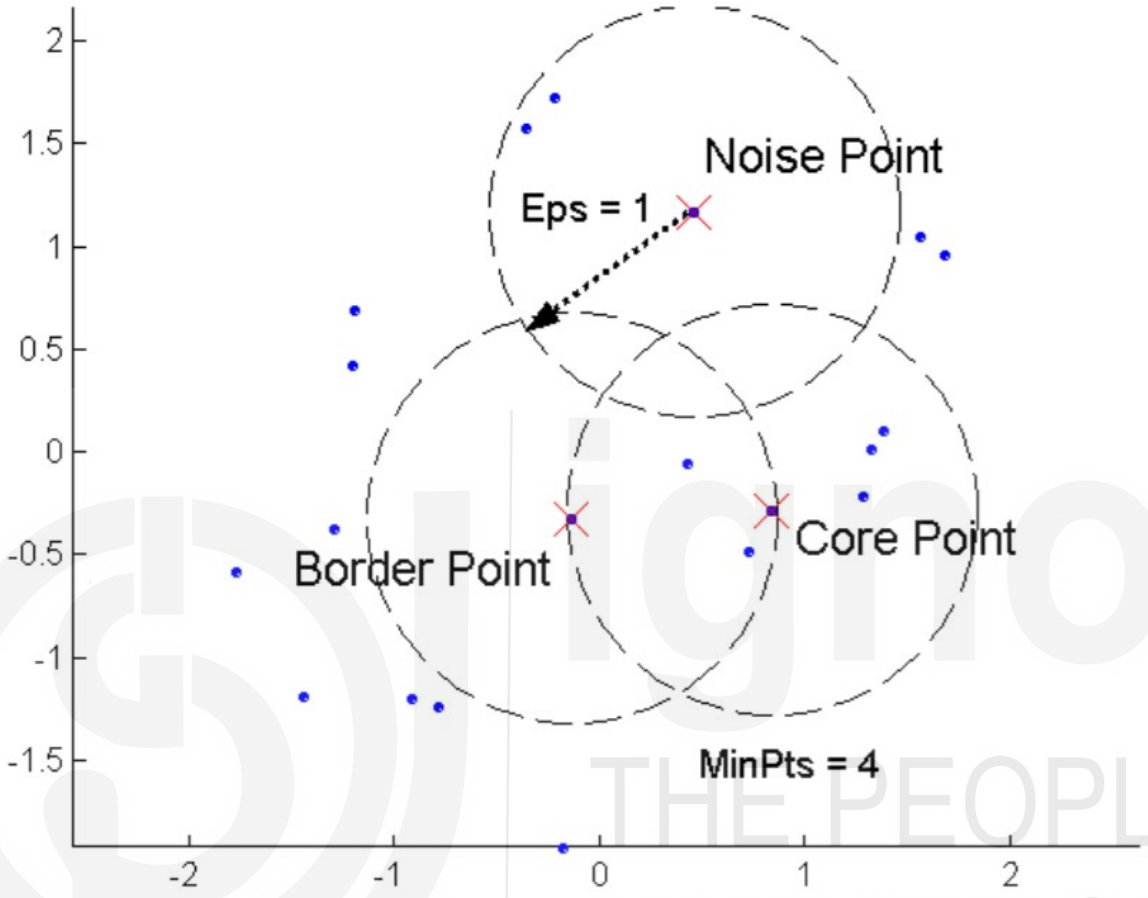
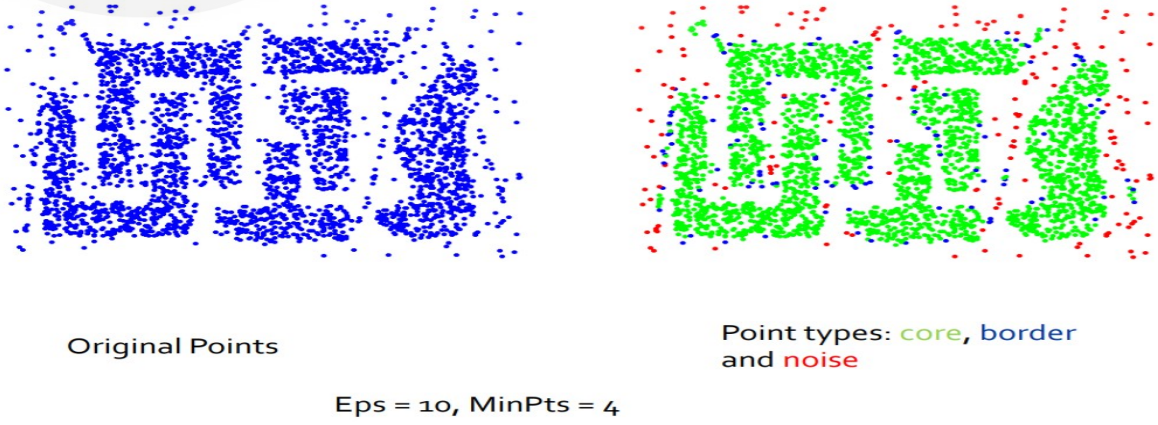


Fig 9. Core point, Border point and Noise point in DBScan



DBSCAN Algorithm

1. Label points as core, border and noise
2. Eliminate noise points for every core point p that has not been assigned to a cluster
3. Create a new cluster with the point p and all the points that are density-connected to p.
4. Assign border points to the cluster of the closest core point.

e) Fuzzy Clustering

Fuzzy clustering is the most used clustering algorithm present in real world and is a sort of clustering in where each data point is assigned to multiple clusters. The algorithm suggests that the data points can belongs to more than one cluster unlike hard clustering where data points can actually belong to only one cluster.

Illustrative Example
A Guava can either be Yellow and Green (hard clustering), but a Guava can also be Yellow and Green (this is fuzzy clustering). Here, the Guava can be Yellow to a certain degree as well as Green to a certain degree. Instead of the Guava belonging to Green [green = 1] and not Yellow [Yellow = 0], the Guava can belong to Green [green = 0.5] and Yellow [Yellow = 0.5]. These values are normalized between 0 and 1; however, they do not represent probabilities, so the two values do not need to add up to 1.

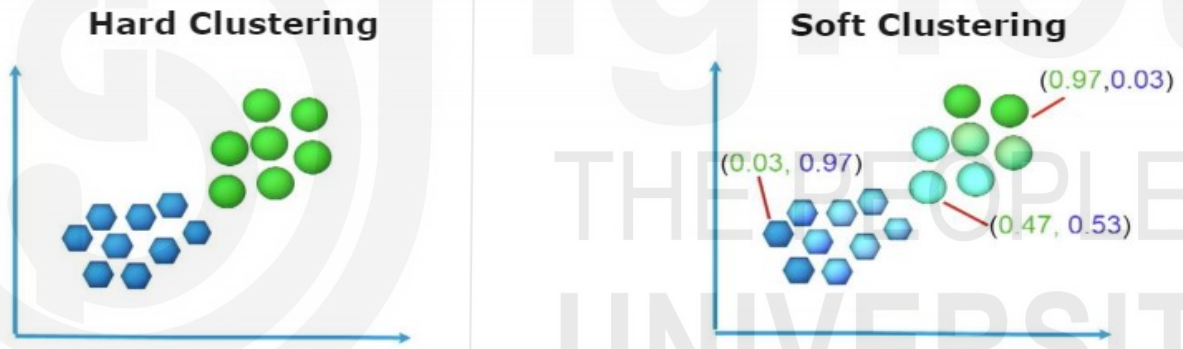


Fig 10. Understanding the algorithm

The Fuzzy Clustering algorithm attempts to group a finite set of n items.
$R = R1, \ldots, rnR = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n\}$ into a set of c fuzzy clusters with respect to some given criterion.

when a finite set of data is given, the algorithm returns a list of clusters centers
$F = f1, \ldots, ffF = \{\boldsymbol{f}_1, \ldots, f_f\}$And a partition matrix

$W = w_{i,j} \in [0,1], i = 1, \ldots, n, j = 1, \ldots, f$,where each element, w_{ij} tells the degree to which element,$\boldsymbol{r}_i$ belongs to cluster $\boldsymbol{f}_j$

The FCM aims to minimize an objective function:

$$\underset{C}{argmin} \sum_{i=1}^{n} \sum_{j=1}^{f} w_{ij}^{m} \parallel \boldsymbol{r}_i - \boldsymbol{f}_j \parallel^2,$$

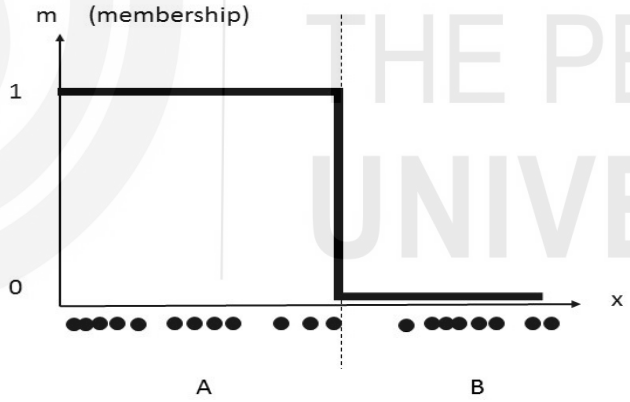
where:

$$w_{ij} = \frac{1}{\sum_{k=1}^{f} \left(\frac{\| \boldsymbol{r}_i - \boldsymbol{f}_j \|}{\| \boldsymbol{r}_i - \boldsymbol{f}_k \|} \right)^{\frac{2}{m-1}}}.$$

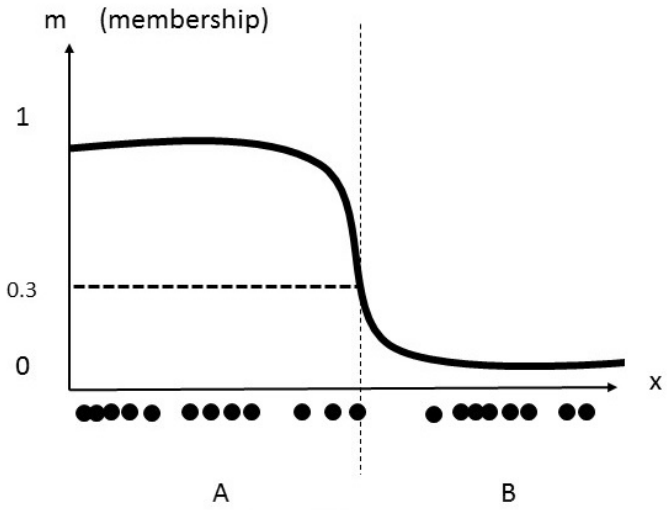
Illustrative Example

To better understand this principle, a classic example of mono-dimensional data is given below on an x axis.

This data set can be conventionally grouped into two clusters, and this is done by selecting a threshold on the x-axis.
The resulting two clusters are labelled 'A' and 'B', as seen in the image below. Each point belonging to the data set would therefore have a membership coefficient of either 0 or 1. This membership coefficient of each corresponding data point is represented by the inclusion of the y-axis.



In fuzzy clustering, each data point can have membership to multiple clusters. By relaxing the definition of membership coefficients from strictly 0 or 1, these values can range from inclusive value from 0 to 1. The following image shows the data set from the previous clustering, but now fuzzy c-means clustering has been applied. First, a new threshold value defining two clusters is generated. Next, new membership coefficients for each data point are generated based on clusters centroids, as well as distance from each cluster centroid.



As we can see, the middle data point belongs to cluster A and cluster B. The value (point) of 0.3 is this data point's membership coefficient for cluster A

Check Your Progress - 4

Qn1. Briefly explain 5 types of clustering.

Qn2. What is Agglomerative Clustering? What are 3 methods that can be used for Agglomerative Clustering?

Qn3. Briefly explain fuzzy clustering and provide a Real-World Example for the same.

Qn4. Differentiate Between AGNES and DIANA

Qn5. What is DBSCAN?

Qn6. Explain how K-Means Algorithm works.

15.7 SUMMARY

The technique of separating a set of data objects into subgroups based on some observation is known as clustering or cluster analysis. Each subset is referred to as a 'cluster,' with objects that are related to one another but different from those in other clusters. In simple terms, the process of separating a population or set of data points into several groups so that data points from the same group can be compared to data points from different groups. Clustering is often confused with classification in data analysis, where separation of data happens on the basis of class labels, while in clustering partitioning of large data sets occurs in groups occurs on the basis of similarity.

Clustering algorithms fall into different categories based on the underlying methodology of the algorithm, the structure of the final solution, or the density based.

In Density Based Clustering Technique, the clusters are created based upon the density of the data points which are represented in the data space.

K-Means: Among clustering algorithms, is an algorithm that tries to minimize the distance of the points in a cluster with their centroid the k-means clustering technique. The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.

AGNES is a type of agglomerative clustering that combines the data objects into a cluster based on similarity.

DBSCAN uses noise to find clusters of any shape in spatial databases.

Fuzzy Clustering is the most widely used clustering technique in practice, and it is a sort of clustering in which each data point belongs to many clusters.

15.8 SOLUTIONS TO CHECK YOUR PROGRESS

Check Your Progress - 1

For detailed answers refer to Section 15.2.

Answer 1.

Clustering is the process of grouping data into groups (also known as clusters) based on patterns found in the data. To put it another way, the work of dividing a population or Data points into groups so that data points in the same group are more comparable to data points in other groups.

Answer 2.

Clustering is a widely used technique in the industry. It is being used in almost every domain, ranging from banking to recommendation engines, document clustering to image segmentation.

- Customer Segmentation
- Document Clustering
- Image Segmentation

Check Your Progress - 2

For detailed answers refer to Section 15.3

Answer 1.

Partitioning approaches start with a set of beginning seeds (or clusters), which are then improved iteratively by shifting objects from one group to another. According to the general criterion of good partitioning, objects in the same cluster are "close" or connected to one another, whereas objects in other clusters are "far away" or notably distinct.

Answer 2.

Popular partitions based on clustering are:

- K-Mean,
- PAM (K-Medoids),
- CLARA algorithm (Clustering Large Applications)

Check Your Progress - 3

For detailed answers refer to Section 15.4.

Answer 1:

Differences in assumptions, runtime, input, and output are all factors. Partitional clustering is, on average, faster than hierarchical clustering. Stronger assumptions are required for partitional clustering. Hierarchical clustering, on the other hand, simply requires a similarity metric. There are no input parameters required for hierarchical clustering, but partitional clustering techniques require a number of clusters to begin. Clusters are divided more meaningfully and subjectively using hierarchical clustering. Partitional clustering, on the other hand, produces k clusters.

Answer 2:

The major distinction between Hierarchical and Partitional Clustering is that each cluster begins as a singleton or individual cluster. The nearest clusters are joined with each iteration. This technique is repeated until only one cluster for Hierarchical clustering remains.

Partitional clustering, on the other hand, needs the analyst to designate K clusters before executing the algorithm, and objects that are closest to the clusters are clustered together. The spacing between the cluster's changes with each repetition. This process continues until the centroid of each cluster no longer moves, or until the halting requirement is reached.

Answer 3:

Hierarchical clustering Technique in terms of space and time complexity:

- **Space complexity:** When the number of data points is large, the space required for the Hierarchical Clustering Technique is large since the similarity matrix must be stored in RAM. The space complexity is measured by the order of the square of n.

  Space complexity = $O(n^2)$ where n is the number of data points.

- **Time complexity:** The time complexity is also very high because we have to execute n iterations and update and restore the similarity matrix in each iteration. The order of the cube of n is the time complexity.

  Time complexity = $O(n^3)$ where n is the number of data points.

Answer 4:

Limitations ofHierarchical Clustering Technique:

1. Hierarchical clustering does not have a mathematical goal.

2. Every method for calculating cluster similarity has its own set of drawbacks.
3. Hierarchical clustering has a high spatial and temporal complexity. As a result, when we have a lot of data, we can't apply this clustering approach.

Check Your Progress - 4

For detailed answers refer to Section 15.5.

Answer 1.

The term "Clustering" is a technique in which the clusters are created based upon the density of the data points which are represented in the data space. The regions that become dense due to the huge number of data points residing in that region are considered as clusters.

Answer 2.

Refer to Page 8

Check Your Progress - 5

For detailed answers refer to Section 15.6.

Answer 1.

a) K-Means

Among clustering algorithms, is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.
K-means is a centroid-based or distance-based technique in which the distances between points are calculated to allocate a point to a cluster. Each cluster in K-Means is paired with a centroid.

b) AGNES (Agglomerative Nesting)

AGNES is a type of agglomerative clustering that combines the data objects into a cluster based on similarity. The result of this algorithm is a tree-based structure called Dendrogram. Here it uses the distance metrics to decide which data points should be combined with which cluster. Basically, it constructs a distance matrix and checks for the pair of clusters with the smallest distance and combines then.

c) Divisive Clustering (DIANA)

Diana basically stands for Divisive Analysis; this is another type of hierarchical clustering where basically it works on the principle of top-down approach (inverse of AGNES) where the algorithm begins by forming a big cluster, and it recursively divides the most dissimilar cluster into two, and it goes on

d) Density-Based Spatial Scan (DBSCAN)

The distance between objects is used to cluster things in most partitioning methods. Only spherical-shaped clusters can be found using these approaches, and clusters of any shape are difficult to find. DBSCAN may create clusters of various shapes; this technique is best suited to datasets with noise or outliers.

Fuzzy Clustering

The most widely used clustering technique in the real world is fuzzy clustering, which is a sort of clustering in which each data point belongs to many clusters. The algorithm suggests that the data points can belongs to more than one cluster unlike hard clustering where data points can actually belong to only one cluster.

Answer 2.

a) **AGNES** (AGglomerativeNESting)

AGNES is a type of agglomerative clustering that combines the data objects into a cluster based on similarity. The result of this algorithm is a tree-based structure called Dendrogram. Here it uses the distance metrics to decide which data points should be combined with which cluster. Basically, it constructs a distance matrix and checks for the pair of clusters with the smallest distance and combines then.

Based on how the distance between each cluster is measured, we can have 3 different methods

- **Single linkage:** Where the shortest distance between the two points in each cluster is defined as the distance between the clusters.
- **Complete linkage:** In this case, we will consider the longest distance between each cluster's points as the distance between the clusters.
- **Average linkage:** In this situation, we'll take the average of each point in one cluster compared to every other point in the other.

Answer 3.

The most widely used clustering technique in the real world is fuzzy clustering, which is a type of clustering in which each object belongs to many clusters**.** The algorithm suggests that the data points can actually belongs to more than one cluster unlike hard clustering where data points can actually belong to only one cluster.

Real-World Example:

A Guava can either be Yellow and Green (hard clustering), but a Guava can also be Yellow and Green (this is fuzzy clustering). Here, the Guava can be Yellow to a certain degree as well as Green to a certain degree. Instead of the Guava belonging to Green [green = 1] and not Yellow [Yellow = 0], the Guava can belong to Green [green = 0.5] and Yellow [Yellow = 0.5]. These values are normalized between 0 and 1; however, they do not represent probabilities, so the two values do not need to add up to 1.

Answer 4:

DIANA is like the reverse of **AGNES**. It begins with the root, in which all observations are included in a single cluster. At each step of the algorithm, the current cluster is split into two clusters that are considered most heterogeneous. The process is iterated until all observations are in their own cluster.

Answer 5:

DBSCAN can form clusters in different shapes; this type of algorithm is most suitable when the dataset contains noise or outliers.

Also, it depends on a density-based concept of cluster: A cluster is defined as the most densely connected set of points.

Answer 6:

Refer Page 10

MULTIPLE CHOICE QUESTIONS

Q1. The objective of clustering is to-

A. Sort the data points into categories.
B. To classify the objects into different classes
C. To predict the values of input data points and generate output.
D. All of the above

Solution: (A)

Q2. Clustering is a-

A. Supervised learning
B. Unsupervised learning
C. Reinforcement learning
D. None

Solution:(B)

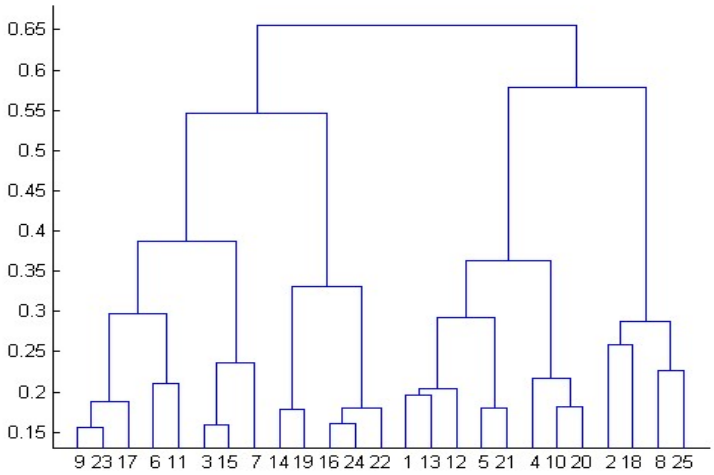
Q3. Which of the following clustering algorithm is most sensitive to outliers?

A. K-means
B. K-modes
C. K-medians
D. K-medoids

Solution: (A)

Explanation: The K-Means clustering approach, which employs the mean of cluster data points to locate the cluster center, is the most sensitive to outliers of all the options.

Q4. You saw the dendrogram below after doing K-Means Clustering analysis on a dataset. From the dendrogram, which of the following conclusions can be drawn?



A. There were 32 data points in clustering analysis

B. The data points analyzed has best number of clusters is 4

C. The proximity function used here is Average-link clustering

D. The above interpretation of dendrogram is not possible for K-Means clustering analysis

Solution: (D)

Explanation: Dendrogram is not possible for K-Means clustering analysis. However, one can create a cluster gram based on K-Means clustering analysis.

.

Q5. What are the two types of Hierarchical Clustering?

A. Top-Down Clustering (Divisive)

B. Bottom-Top Clustering (Agglomerative)

C. Dendrogram

D. K-means

Solution: Both A & B

Q6. Is it reasonable to assume the same clustering results from two K-Mean clustering runs?

A. Yes

B. No

Solution: (B)

Explanation: Instead, the K-Means clustering technique talks about local minima, which may or may not equate to global minima in some situations. As a result, it's a good idea to run the K-Means method several times before making any conclusions about the clusters.

It's worth noting, though, that by using the same seed value for each run, you may get the same clustering results via K-means. However, this is accomplished by simply instructing the algorithm to select the same set of random numbers for each iteration.

Q7. Which of the following clustering techniques has a difficulty with local optima convergence?

A. Agglomerative clustering algorithm

B. K- Means clustering algorithm

C. Expectation-Maximization clustering algorithm

D. Diverse clustering algorithm

Options:

A. A only
B. B and C
C. B and D
D. A and D

Solution: (B)

Out of the options given, only K-Means clustering algorithm and EM clustering algorithm has the drawback of converging at local minima.

Q8. Which of the following is a bad characteristic of a dataset for clustering analysis-?

A. Objects with outliers
B. Objects with different densities
C. Objects with non-convex shapes
D. All of the above

Solution: (D)

Q9. Which is the following statement being incorrect?

A. k-means clustering is a method of vector quantization.
B. k-means clustering groups number of observations into k clusters.
C. k-means is same ask-nearest neighbor.
D. None

Solution: (C)

Q10. What do you understand by dendrogram?

A. A hierarchical structure
B. A diagrammatical view
C. A graph
D. None

Solution: (A)

UNIT 16 MACHINE LEARNING – PROGRAMMING USING PYTHON

16.0 INTRODUCTION

In this unit we will see the implementation of various machine learning algorithms, learned in this course. To understand the codes you need to have understanding of the respective Machine learning algorithms along with that understanding of Python programming is must. The codes are readily using various libraries of Python programming language viz. Scikit Learn, Matplotlib, numpy etc., you can execute these codes through any of the Python programming tools. Most of the machines learning algorithms, you learned in this course, are implemented here, just try to execute them and analyse the results.

16.1 OBJECTIVES

After going through this unit, you should be able to:

- Understand the implementation aspect of various machine learning algorithms

16.2 CLASSIFICATION ALGORITHMS

The starting units of this course primarily focused on the various classification algorithms viz. Naïve Bayes classifiers, K-Nearest Neighbour (K-NN), Decision Trees, Logistic Regression and Support Vector Machines.The theoretical aspects of

the same is already discussed in the respective units, now we will see the implementation part of the mentioned classifiers, in Python programming language.

16.2.1 NAIVE BAYES

It is a method of classification that is founded on Bayes' Theorem and makes the assumption that predictors are free to act independently of one another. A Naive Bayes classifier, to put it in layman's words, makes the assumption that the existence of one particular characteristic in a class is unrelated to the presence of any other feature.

We have already discussed this classifier in detail in Block 3 Unit 10 of this course, you may refer to Block 3 Unit 10 to understand the concept.

The following procedures need to be carried out in order to classify data using the Naive Bayes method.

- In the first step, we will begin by importing the dataset as well as any necessary dependencies...
- The second step is to get the prior probability of each class using the formula P(y).
- The Third Step is to Determine the likelihood of each characteristic using the table you just created...
- Final and the Fourth Step is to Calculate the Posterior Probability for each class by applying the Naive Bayesian equation.

The screenshot of the executed code is given below

Implementation code in Python

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.4, random_state=1)
# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(X_test)
# comparing actual response values (y_test) with predicted response  values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```

OUTPUT :

Gaussian Naive Bayes model accuracy(in %): 95.0

16.2.2 K-Nearest Neighbour (K-NN)

You have already discussed this classifier in detail in Block 3 Unit 10 of this course, you may refer to Block 3 Unit 10 to understand the concept.

We learned that Suppose the value of K is 3. The KNN algorithm starts by calculating the distance of point X from all the points. It then finds the 3 nearest points with least distance to point X

In the example shown below following steps are performed:

• In Step 1, the scikit-learn package is used to import the k-nearest neighbour algorithm.
• Step 2. is to create the feature variables and the target variables.
• Step 3. Separate the data into the test data and the training data.
• Step 4.Generate a k-NN model using neighbours value.
• Step 5. Train the model using the data or adjust the model based on the data.
• Proceed to Step 6, which is to make a forecast.

Now, in this section, we will see how Python's Scikit-Learn library can be used to implement the KNN algorithm

Implementation code in Python

The screenshot of the executed code is given below

```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
# Loading data
irisData = load_iris()
# Create feature and target arrays
X = irisData.data
y = irisData.target
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size = 0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
# Predict on dataset which model has not seen before
print(knn.predict(X_test))
```

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

16.2.3 Desicion Tree Implementation

A decision tree is a type of supervised machine learning algorithm that may be used for both regression and classification tasks. It is one of the most popular and widely used machine learning techniques.

In this case, the decision tree method creates a node for each attribute present in the dataset, with the attribute that is considered to be the most significant being placed at the top of the tree. When we first get started, we will think of the entire training set as the root. There must be a categorical breakdown of the feature values.

Before beginning to develop the model, the values are discretized in order to determine whether or not they are continuous. A recursive process distributes records according to the attribute values of each record. A statistical method is utilised in order to determine which qualities should be placed at the tree's root and which should be placed at internal nodes.

You have already discussed this classifier in detail in Block 3 Unit 10 of this course, you may refer to Block 3 Unit 10 to understand the concept.

Implementation code in Python

The screenshot of the executed code is given below

```
# Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
# Function importing Dataset
def importdata():
 balance_data = pd.read_csv( 'https://archive.ics.uci.edu/ml/machine-learning-'+
'databases/balance-scale/balance-scale.data', sep= ',', header = None)
# Printing the dataset shape
 print ("Dataset Length: ", len(balance_data))
 print ("Dataset Shape: ", balance_data.shape)
# Printing the dataset obseravtions
 print ("Dataset: ",balance_data.head())
 return balance_data
# Function to split the dataset
def splitdataset(balance_data):
```

```
# Function to split the dataset
def splitdataset(balance_data):
# Separating the target variable
 X = balance_data.values[:, 1:5]
 Y = balance_data.values[:, 0]
# Splitting the dataset into train and test
 X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.3, random_state = 100)
 return X, Y, X_train, X_test, y_train, y_test
# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):
# Creating the classifier object
 clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,max_depth=3, min_samples_leaf=5)
# Performing training
 clf_gini.fit(X_train, y_train)
 return clf_gini
# Function to perform training with entropy.
def train_using_entropy(X_train, X_test, y_train):
# Decision tree with entropy
 clf_entropy = DecisionTreeClassifier(
 criterion = "entropy", random_state = 100,
 max_depth = 3, min_samples_leaf = 5)
# Performing training
```

```
 # Performing training
  clf_entropy.fit(X_train, y_train)
  return clf_entropy
 # Function to make predictions
 def prediction(X_test, clf_object):
 # Predicton on test with giniIndex
  y_pred = clf_object.predict(X_test)
  print("Predicted values:")
  print(y_pred)
  return y_pred
 # Function to calculate accuracy
 def cal_accuracy(y_test, y_pred):
  print("Confusion Matrix: ",
  confusion_matrix(y_test, y_pred))
  print ("Accuracy : ",
  accuracy_score(y_test,y_pred)*100)
  print("Report : ",
  classification_report(y_test, y_pred))
 # Driver code
 def main():
 # Building Phase
```

```
 classification_report(y_test, y_pred))
# Driver code
def main():
# Building Phase
 data = importdata()
 X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
 clf_gini = train_using_gini(X_train, X_test, y_train)
 clf_entropy = train_using_entropy(X_train, X_test, y_train)
# Operational Phase
 print("Results Using Gini Index:")
# Prediction using gini
 y_pred_gini = prediction(X_test, clf_gini)
 cal_accuracy(y_test, y_pred_gini)
 print("Results Using Entropy:")
# Prediction using entropy
 y_pred_entropy = prediction(X_test, clf_entropy)
 cal_accuracy(y_test, y_pred_entropy)
# Calling main function
if __name__=="__main__":
 main()
```

```
Dataset Length:  625
Dataset Shape:  (625, 5)
Dataset:     0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
Results Using Gini Index:
Predicted values:
['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R'
 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 67 18]
 [ 0 19 71]]
```

```
Accuracy :  73.40425531914893
Report :               precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.73      0.79      0.76        85
           R       0.74      0.79      0.76        90

    accuracy                           0.73       188
   macro avg       0.49      0.53      0.51       188
weighted avg       0.68      0.73      0.71       188

Results Using Entropy:
Predicted values:
['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy :  70.74468085106383
```

```
Accuracy :  70.74468085106383
Report :               precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.71      0.74      0.72        85
           R       0.71      0.78      0.74        90

    accuracy                           0.71       188
   macro avg       0.47      0.51      0.49       188
weighted avg       0.66      0.71      0.68       188

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
```

16.2.4 Logistic Regression

Logistic Regression (LR) is a classification algorithm that is used in Machine Learning to predict the likelihood of a categorical dependent variable. It is also known as "logistic regression." The dependent variable in logistic regression is a binary variable, which means that it comprises data that is either recorded as 1 (yes, success, etc.) or 0. (no, failure, etc.).

It should be brought to your attention that the Naive Bayes model is a generative model, whereas the LR model is a discriminative model. LR performs better than naive bayes when it comes to colinearity. This is because naive bayes expects all of the characteristics to be independent, while LR does not. Naive bayes works well with small datasets.

You have already discussed this classifier in detail in Block 3 Unit 10 of this course, you may refer to Block 3 Unit 10 to understand the concept.

Implementation code in Python

The screenshot of the executed code is given below

```
# Logistic Regression

# Importing the libraries

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

# Importing the datasets

datasets = pd.read_csv('Social_Network_Ads.csv')
X = datasets.iloc[:, [2,3]].values
Y = datasets.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

```
# Feature Scaling

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)

# Fitting the Logistic Regression into the Training set

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_Train, Y_Train)

# Predicting the test set results

Y_Pred = classifier.predict(X_Test)

# Making the Confusion Matrix
```

```
# Making the Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_Test, Y_Pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap
X_Set, Y_Set = X_Train, Y_Train
X1, X2 = np.meshgrid(np.arange(start = X_Set[:,0].min() -1, stop = X_Set[:, 0].max() +1, step = 0.01),
                     np.arange(start = X_Set[:,1].min() -1, stop = X_Set[:, 1].max() +1, step = 0.01))

plt.contourf(X1,X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X2.max())
plt.ylim(X2.min(), X2.max())
```

```
for i, j in enumerate(np.unique(Y_Set)):
    plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j,1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression ( Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which
*c* argument looks like a single numeric RGB or RGBA sequence, which
```

16.2.5 Support Vector Machine

Support Vector Machine, more usually referred to as SVM, is a technique for supervised and linear machine learning that is most frequently utilised for the purpose of addressing classification issues. Support Vector Classification is another name for SVM. In addition, there is a subset of SVM known as SVR, which stands for Support Vector Regression. SVR applies the similar concepts to the problem-solving process when addressing regression issues. SVM also offers a method known as the kernel method, which is also known as the kernel SVM. This method enables us to deal with non-linearity.

The following are the steps involved in implementation:

- Import the Libraries
- Make sure the Dataset is loaded.
- Dataset will be divided into X and Y.
- Create a Training set and a Test set from the X and Y Datasets.
- Scaling the features should be done.
- Ensure that the SVM is adjusted to the Training set.
- Make a guess about the results of the test set.
- Construct the Matrix of Confusion.

You have already discussed this classifier in detail in Block 3 Unit 10 of this course, you may refer to Block 3 Unit 10 to understand the concept.

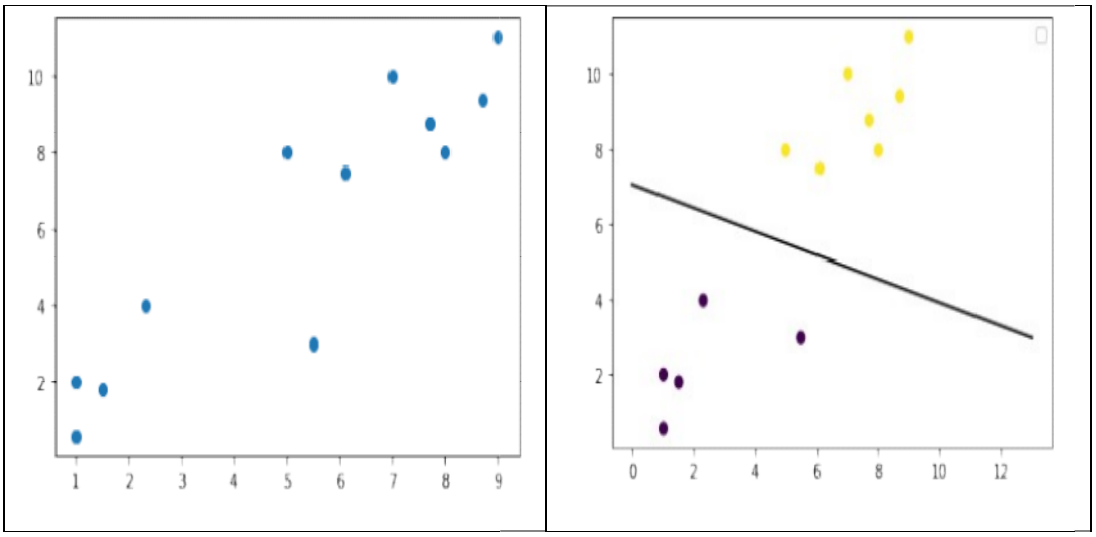
Implementation code in Python

The screenshot of the executed code is given below

```
# SUPPORT VECTOR MACHINES
# Importing Libraries
import matplotlib.pyplot as plt
import numpy as np
from sklearn import svm
# linear data
X = np.array([1, 5, 1.5, 8, 1, 9, 7, 8.7, 2.3, 5.5, 7.7, 6.1])
y = np.array([2, 8, 1.8, 8, 0.6, 11, 10, 9.4, 4, 3, 8.8, 7.5])
# show unclassified data
plt.scatter(X, y)
plt.show()
# shaping data for training the model
training_X = np.vstack((X, y)).T
training_y = [0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1]
# define the model
clf = svm.SVC(kernel='linear', C=1.0)
# train the model
clf.fit(training_X, training_y)
# get the weight values for the linear equation from the trained SVM  model
w = clf.coef_[0]
# get the y-offset for the linear equation
a = -w[0] / w[1]
# make the x-axis space for the data points
XX = np.linspace(0, 13)
# get the y-values to plot the decision boundary
yy = a * XX - clf.intercept_[0] / w[1]
```

```
# get the y-values to plot the decision boundary
yy = a * XX - clf.intercept_[0] / w[1]
# plot the decision boundary
plt.plot(XX, yy, 'k-')
# show the plot visually
plt.scatter(training_X[:, 0], training_X[:, 1], c=training_y)
plt.legend()
plt.show()
```

OUTPUT:



Check Your Progress - 1

1. Make Suitable assumptions and modify the python code of following Classification algorithms:
    a. K-NN
    b. Decision Tree
    c. Logistic Regression
    d. Support Vector Machines

16.3 REGRESSION ALGORITHMS

We learned about the basic concept of regression in the respective unit of this course, in this unit we will implement the Linear regression and Polynomial regression in Python language. Lets start with the Linear regression.

16.3.1 Linear Regression

The purpose of a linear regression model is to determine whether or not there is a connection between one or more characteristics (also known as independent variables) and a target variable that is continuous (dependent variable). Linear Regression is referred to as Uni-variate Linear Regression when there is only one feature, and it is referred to as Several Linear Regression when there are multiple features.

Following are the stages involved in the implementation of a linear regression model:

- Firstly, initialise the parameters.
- Given the value of an independent variable, predict what the value of a dependent variable will be.
- Determine the amount of error that each forecast has for each data point.
- Using a0 and a1, perform the calculation for the partial derivative.

- Add up the individual costs that you have determined for each of the numbers.

You have already discussed this classifier in detail in Block 3 Unit 11 of this course, you may refer to Block 3 Unit 11 to understand the concept.

Implementation code in Python

The screenshot of the executed code is given below

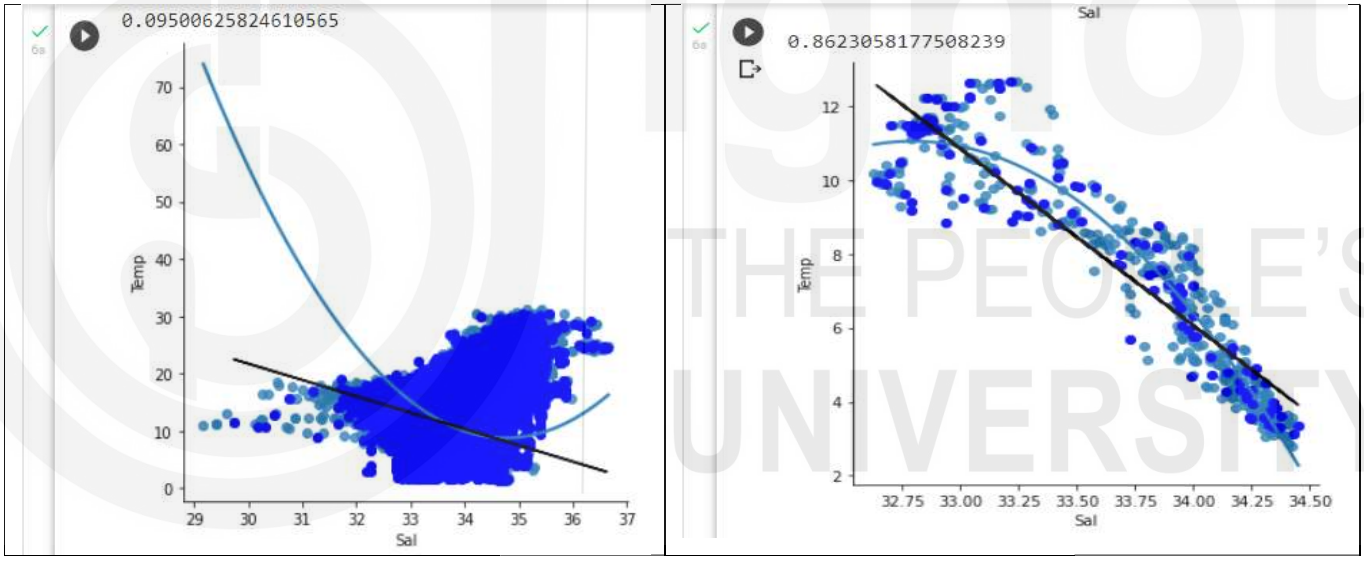
```
# LINEAR REGRESSION

# Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
#Reading Dataset, and Changing the file read location to the location of the dataset
df = pd.read_csv('bottle.csv')
df_binary = df[['Salnty', 'T_degC']]
# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
# Renaming the columns for easier writing of the code
df_binary.head()
# Displaying only the 1st rows along with the column names
##Exploring data scatter
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)
# Plotting the data scatter
##Data Cleaning
# Eliminating NaN or missing input numbers
df_binary.fillna(method ='ffill', inplace = True)
##Training Model
X = np.array(df_binary['Sal']).reshape(-1, 1)
```

```
##Training Model
X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)
# Separating the data into independent and dependent variables # Converting each dataframe into a numpy array
# since each dataframe contains only one column
df_binary.dropna(inplace = True)
# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
##Exploring our results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
# Data scatter of predicted values
##Working with a smaller dataset.
df_binary500 = df_binary[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="Sal", y ="Temp", data = df_binary500,order = 2, ci = None)
df_binary500.fillna(method ='ffill', inplace = True)
X = np.array(df_binary500['Sal']).reshape(-1, 1)
y = np.array(df_binary500['Temp']).reshape(-1, 1)
df_binary500.dropna(inplace = True)
```

```
df_binary500.fillna(method ='ffill', inplace = True)
X = np.array(df_binary500['Sal']).reshape(-1, 1)
y = np.array(df_binary500['Temp']).reshape(-1, 1)
df_binary500.dropna(inplace = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
```

OUTPUT:



16.3.2 Polynomial Regression

Polynomial Regression is a type of linear regression in which the relationship between the independent variable x and the dependent variable y is described as an nth degree polynomial. This type of regression is also known as "extended" linear regression. Polynomial regression is used to model a nonlinear relationship between the value of an independent variable x and the conditional mean of a dependent variable y. This relationship is represented by the notation E(y |x). solely as a result of the non-linear relationship that exists between the dependent and the independent variables When we want to transform linear regression into polynomial regression, we just add some polynomial terms.

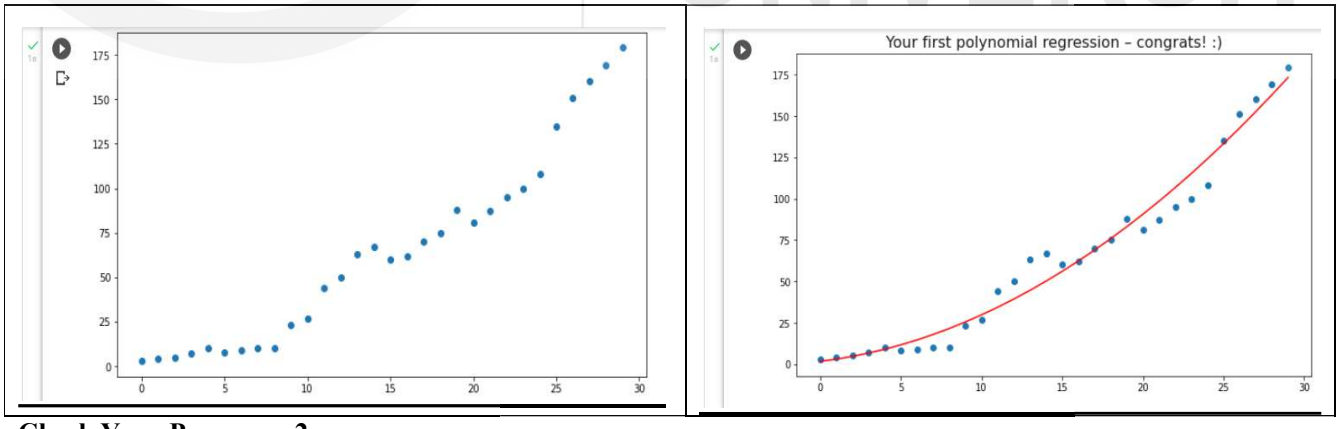
You have already discussed this classifier in detail in Block 3 Unit 11 of this course, you may refer to Block 3 Unit 11 to understand the concept.

Implementation code in Python

The screenshot of the executed code is given below

```
#POLYNOMIAL REGRESSION
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x = np.arange(0, 30)
y = [3, 4, 5, 7, 10, 8, 9, 10, 10, 23, 27, 44, 50, 63, 67, 60, 62, 70, 75, 88,
     81, 87, 95, 100, 108, 135, 151, 160, 169, 179]
plt.figure(figsize=(10,6))
plt.scatter(x, y)
plt.show()
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(x.reshape(-1, 1))
poly_features = poly.fit_transform(x.reshape(-1, 1))
from sklearn.linear_model import LinearRegression
poly_reg_model = LinearRegression()
poly_reg_model.fit(poly_features, y)
y_predicted = poly_reg_model.predict(poly_features)
plt.figure(figsize=(10, 6))
plt.title("Your first polynomial regression - congrats! :)", size=16)
plt.scatter(x, y)
plt.plot(x, y_predicted, c="red")
plt.show()
```

OUTPUT:



Check Your Progress - 2

2. Make Suitable assumptions and modify the python code of following Regression algorithms:
   a. Linear regression
   b. Polynomial egression

16.4 FEATURE SELECTION AND EXTRACTION

Feature selection and extraction are one of the most important steps that must be performed in order for machine learning to be successful. While we covered the theoretical aspects of this process in the earlier units of this course, it is now time to understand the implementation part of the mechanisms that we have learned for Feature selection and extraction. Let's begin with dimensionality reduction, which is the process of lowering the number of random variables that are being considered by generating a set of primary variables. Dimensionality reduction may be seen of as a way to streamline the analysis process.

16.4.1 Principal Component Analysis (PCA)

You have already discussed this classifier in detail in Block 4 Unit 13 of this course, you may refer to Block 4 Unit 13 to understand the concept.

Among the various techniques the Principal Component Analysis (PCA) is most frequently used, and the implementation of PCA is given below:

Implementation code in Python

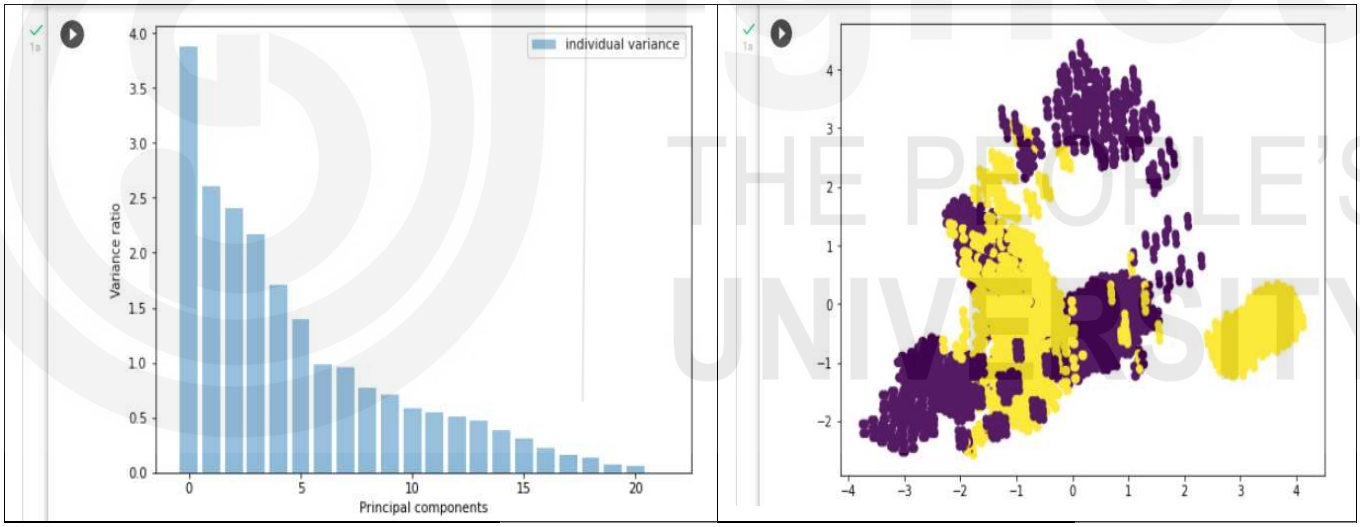
The screenshot of the executed code is given below

```
# PRINCIPAL COMPONENT ANALYSISI(PCA)
# Importing the libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
m_data = pd.read_csv('mushrooms.csv')

# Machine learning systems work with integers, we need to encode these
# string characters into ints
encoder = LabelEncoder()
# Now apply the transformation to all the columns:
for col in m_data.columns:
 m_data[col] = encoder.fit_transform(m_data[col])
X_features = m_data.iloc[:,1:23]
y_label = m_data.iloc[:, 0]
# Scale the features
scaler = StandardScaler()
X_features = scaler.fit_transform(X_features)
# Visualize
```

```
# Visualize
pca = PCA()
pca.fit_transform(X_features)
pca_variance = pca.explained_variance_
plt.figure(figsize=(8, 6))
plt.bar(range(22), pca_variance, alpha=0.5, align='center',  label='individual variance')
plt.legend()
plt.ylabel('Variance ratio')
plt.xlabel('Principal components')
plt.show()
pca2 = PCA(n_components=17)
pca2.fit(X_features)
x_3d = pca2.transform(X_features)
plt.figure(figsize=(8,6))
plt.scatter(x_3d[:,0], x_3d[:,5], c=m_data['class'])
plt.show()
```

OUTPUT :



Check Your Progress - 3

3. Make Suitable assumptions and modify the python code of Principal Component Analysis, for dimensionality reduction.

16.5 ASSOCIATION RULES

We discussed Apriori algorithm and FP Growth algorithm, while studying the topic of Association Rules. These algorithms are frequently used in pattern matching. Since FP Growth is a step ahead to Apriori Algorithm, we are discussing the implementation of Apriori algorithm only

16.5.1 Apriori Algorithm

The Apriori algorithm is a data mining technique that is used for mining frequent item sets and appropriate association rules. It does this by using a mathematical formula. We focused on the definitions of association rule mining and Apriori algorithms, as well as the application of an Apriori algorithm, in the area of this class that was most pertinent to the topic. In this section, we will construct one Apriori model utilising the Python programming language and a hypothetical situation involving a small firm. However, it does have some limits, the effects of which can be mitigated using a variety of different approaches. Data mining and pattern recognition are two of the many applications that see widespread use of the method.

The candidate set is produced by the model that is described further down below by merging the set of frequent items from the step before it.

Conduct testing on subsets, and if the candidate set contains infrequent item sets, remove them. And then calculate the final frequent itemset by obtaining the items that meet the minimal support requirement.

You have already discussed this classifier in detail in Block 4 Unit 14 of this course, you may refer to Block 4 Unit 14 to understand the concept.

Implementation code in Python

The screenshot of the executed code is given below

```
# APRIORI ALGORITHM
##Importing the required libraries
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
## Loading and exploring the data
# Changing the working location to the location of the file
# Loading the Data
data = pd.read_excel('Online Retail.xlsx')
data.head()
# Exploring the columns of the data
data.columns
# Exploring the different regions of transactions
data.Country.unique()
# Stripping extra spaces in the description
data['Description'] = data['Description'].str.strip()
# Dropping the rows without any invoice number
data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)
data['InvoiceNo'] = data['InvoiceNo'].astype('str')
# Dropping all transactions which were done on credit
data = data[~data['InvoiceNo'].str.contains('C')]

##Splitting the data according to the region of transaction
```

```
##Splitting the data according to the region of transaction
# Transactions done in France
basket_France = (data[data['Country'] =="France"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
# Transactions done in the United Kingdom
basket_UK = (data[data['Country'] =="United Kingdom"] .groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
# Transactions done in Portugal
basket_Por = (data[data['Country'] =="Portugal"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
# Transactions done in Sweden
basket_Sweden = (data[data['Country'] =="Sweden"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
## Hot encoding the Data
```

```
# Defining the hot encoding function to make the data suitable
 # for the concerned libraries
def hot_encode(x):
 if(x<= 0):
  return 0
 if(x>= 1):
  return 1
# Encoding the datasets
basket_encoded = basket_France.applymap(hot_encode)
basket_France = basket_encoded
basket_encoded = basket_UK.applymap(hot_encode)
basket_UK = basket_encoded
basket_encoded = basket_Por.applymap(hot_encode)
basket_Por = basket_encoded
basket_encoded = basket_Sweden.applymap(hot_encode)
basket_Sweden = basket_encoded
## Building the models and analyzing the results
# Building the model
###France
frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)
# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False,  False])
```

```
# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False,  False])
print(rules.head())
###United Kingdom
frq_items = apriori(basket_UK, min_support = 0.01, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False,  False])
print(rules.head())
###Portugal
frq_items = apriori(basket_Por, min_support = 0.05, use_colnames =
True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False,  False])
print(rules.head())
### Sweden
frq_items = apriori(basket_Sweden, min_support = 0.05, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False,  False])
print(rules.head())
```

```
2019  (SUKI  SHOULDER BAG, JAM MAKING SET PRINTED)     (DOTCOM POSTAGE)
2295        (HERB MARKER MINT, HERB MARKER THYME)  (HERB MARKER ROSEMARY)
2300   (HERB MARKER PARSLEY, HERB MARKER ROSEMARY)     (HERB MARKER THYME)
2301      (HERB MARKER PARSLEY, HERB MARKER THYME)  (HERB MARKER ROSEMARY)

      antecedent support  consequent support   support  confidence      lift  \
116             0.011036             0.037928  0.010768    0.975728  25.725872
2019            0.011625             0.037928  0.011196    0.963134  25.393807
2295            0.010714             0.012375  0.010232    0.955000  77.173095
2300            0.011089             0.012321  0.010553    0.951691  77.240055
2301            0.011089             0.012375  0.010553    0.951691  76.905682

      leverage  conviction
116   0.010349   39.637371
2019  0.010755   26.096206
2295  0.010099   21.947227
2300  0.010417   20.444951
2301  0.010416   20.443842
                           antecedents                          consequents  \
1170  (SET 12 COLOUR PENCILS DOLLY GIRL)     (SET 12 COLOUR PENCILS SPACEBOY)
1171    (SET 12 COLOUR PENCILS SPACEBOY)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1172  (SET 12 COLOUR PENCILS DOLLY GIRL)   (SET OF 4 KNICK KNACK TINS LONDON)
1173  (SET OF 4 KNICK KNACK TINS LONDON)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1174  (SET 12 COLOUR PENCILS DOLLY GIRL)  (SET OF 4 KNICK KNACK TINS POPPIES)
```

```
      antecedent support  consequent support   support  confidence       lift  \
1170            0.051724            0.051724  0.051724         1.0  19.333333
1171            0.051724            0.051724  0.051724         1.0  19.333333
1172            0.051724            0.051724  0.051724         1.0  19.333333
1173            0.051724            0.051724  0.051724         1.0  19.333333
1174            0.051724            0.051724  0.051724         1.0  19.333333

      leverage  conviction
1170  0.049049         inf
1171  0.049049         inf
1172  0.049049         inf
1173  0.049049         inf
1174  0.049049         inf
                        antecedents                       consequents  \
0     (12 PENCILS SMALL TUBE SKULL)      (PACK OF 72 SKULL CAKE CASES)
1     (PACK OF 72 SKULL CAKE CASES)      (12 PENCILS SMALL TUBE SKULL)
4           (36 DOILIES DOLLY GIRL)     (ASSORTED BOTTLE TOP  MAGNETS)
5    (ASSORTED BOTTLE TOP  MAGNETS)           (36 DOILIES DOLLY GIRL)
180  (CHILDRENS CUTLERY DOLLY GIRL)  (CHILDRENS CUTLERY CIRCUS PARADE)

     antecedent support  consequent support   support  confidence  lift  \
0              0.055556            0.055556  0.055556         1.0  18.0
1              0.055556            0.055556  0.055556         1.0  18.0
4              0.055556            0.055556  0.055556         1.0  18.0
5              0.055556            0.055556  0.055556         1.0  18.0
```

```
                        antecedents                       consequents  \
0     (12 PENCILS SMALL TUBE SKULL)      (PACK OF 72 SKULL CAKE CASES)
1     (PACK OF 72 SKULL CAKE CASES)      (12 PENCILS SMALL TUBE SKULL)
4           (36 DOILIES DOLLY GIRL)     (ASSORTED BOTTLE TOP  MAGNETS)
5    (ASSORTED BOTTLE TOP  MAGNETS)           (36 DOILIES DOLLY GIRL)
180  (CHILDRENS CUTLERY DOLLY GIRL)  (CHILDRENS CUTLERY CIRCUS PARADE)

     antecedent support  consequent support   support  confidence  lift  \
0              0.055556            0.055556  0.055556         1.0  18.0
1              0.055556            0.055556  0.055556         1.0  18.0
4              0.055556            0.055556  0.055556         1.0  18.0
5              0.055556            0.055556  0.055556         1.0  18.0
180            0.055556            0.055556  0.055556         1.0  18.0

     leverage  conviction
0    0.052469         inf
1    0.052469         inf
4    0.052469         inf
5    0.052469         inf
180  0.052469         inf
```

16.6 CLUSTERING ALGORITHMS

We learned about the theoretical aspects of various clustering algorithms like K-Means, DBSCAN etc. in the respective unit of this course. The K-Means algorithm was quite simple and hence its implementation is given below:

16.6.1 K-Means - Implementation code in Python

You have already discussed this classifier in detail in Block 4 Unit 15 of this course, you may refer to Block 4 Unit 15 to understand the concept.

Implementation code in Python

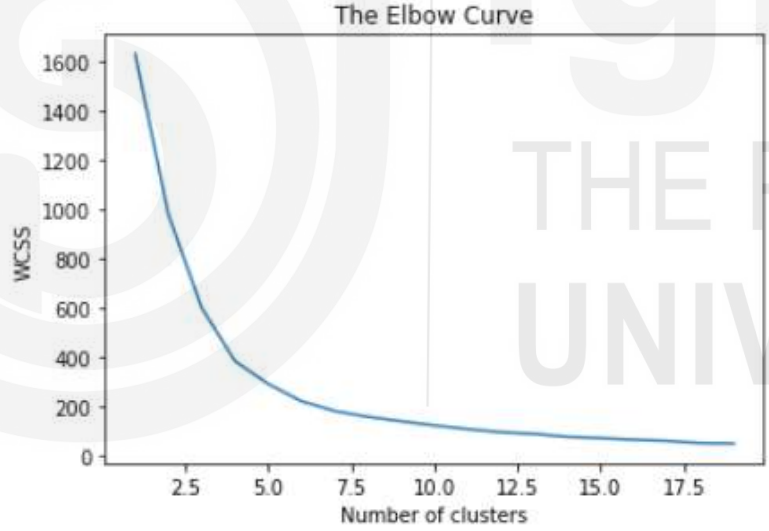
The screenshot of the executed code is given below

```
# K-MEANS CLUSTERING
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x1=10*np.random.rand(100,2)
x1.shape
kmean=KMeans(n_clusters=3)
kmean.fit(x1)
kmean.cluster_centers_
kmean.labels_
wcss = []
for i in range(1,20):
 kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
 kmeans.fit(x1)
 wcss.append(kmeans.inertia_)
 print('Cluster', i, 'Inertia', kmeans.inertia_)
plt.plot(range(1,20),wcss)
plt.title('The Elbow Curve')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') ##WCSS stands for total within-cluster sum of square
plt.show()
```

```
plt.ylabel('WCSS') ##WCSS stands for total within-cluster sum of square
plt.show()
```

```
Cluster 1 Inertia 1633.4729386366648
Cluster 2 Inertia 980.7251659226347
Cluster 3 Inertia 602.6606235675433
Cluster 4 Inertia 386.8266954126674
Cluster 5 Inertia 293.52522127511526
Cluster 6 Inertia 223.57829467348864
Cluster 7 Inertia 183.55842327586788
Cluster 8 Inertia 160.52571627742793
Cluster 9 Inertia 142.12347605903437
Cluster 10 Inertia 125.1405682359277
Cluster 11 Inertia 110.07801216378505
Cluster 12 Inertia 97.71840448709966
Cluster 13 Inertia 91.0468655863335
Cluster 14 Inertia 78.97168184469689
Cluster 15 Inertia 73.78690617997606
Cluster 16 Inertia 67.33331767461436
Cluster 17 Inertia 62.26772230658193
Cluster 18 Inertia 53.44137427335066
Cluster 19 Inertia 51.55574132217379
```

```
Cluster 14 Inertia 78.97168184469689
Cluster 15 Inertia 73.78690617997606
Cluster 16 Inertia 67.33331767461436
Cluster 17 Inertia 62.26772230658193
Cluster 18 Inertia 53.44137427335066
Cluster 19 Inertia 51.55574132217379
```



Check Your Progress - 4

4. Make Suitable assumptions and modify the python code of K-Means algorithm

16.8 SUMMARY

In this unit we understood the implementation of various machine learning algorithms for Classification, Regression, Dimension Reductionality and clustering. The theoretical aspects of the respective algorithm were already discussed in the respective units of this course.

16.9 SOLUTIONS/ANSWERS

Check Your Progress - 1

1. Make Suitable assumptions and modify the python code of following Classification algorithms:
    a. K-NN
    b. Decision Tree
    c. Logistic Regression
    d. Support Vector Machines

Solution : Refer to section 16.2

Check Your Progress - 2

2. Make Suitable assumptions and modify the python code of following Regression algorithms:
    a. Linear regression
    b. Polynomial egression

Solution : Refer to section 16.3

Check Your Progress - 3

3. Make Suitable assumptions and modify the python code of Principal Component Analysis, for dimensionality reduction.

Solution : Refer to section 16.4

Check Your Progress - 4

4. Make Suitable assumptions and modify the python code of K-Means algorithm

Solution : Refer to section 16.6

16.10 FURTHER READINGS

- https://www.kaggle.com/
- https://www.github.com/
- https://towardsdatascience.com
- https://machinelearningmastery.com