DATA MINING

MODULE 2

Data Visualization

Data visualization aims to communicate data clearly and effectively through graphical representation. More popularly, we can take advantage of visualization techniques to discover data relationships that are otherwise not easily observable by looking at the raw data. Nowadays, people also use datavisualization to create fun and interesting graphics.

Visualize: "To form a mental vision, image, or picture of (something not visible or present to the sight, or of an abstraction); to make visible to the mind or imagination."

Visualization: It is the use of computer graphics to create visual images which aid in the understanding of complex, often massive representations of data.

Visual Data Mining: It is the process of discovering implicit but useful knowledge from large data sets using visualization techniques.

Tablev/s Graph

Table	Graph
You need to lookup specific values	The message is contained in the shape of the values
Users need precise values	You want to reveal relationships among multiple values(similarities and differences)
You need to precisely compare related values	Show general trends

Data Visualization Methods

Univariate Analysis

- Univariate analysis is basically the simplest form of data analysis or visualization where we are only concerned with analyzing one data attribute or variable and visualizing the same (one dimension).
- **Example:** Analyzing the heights of a group of individuals or the scores of students in a single subject are univariate analyses.

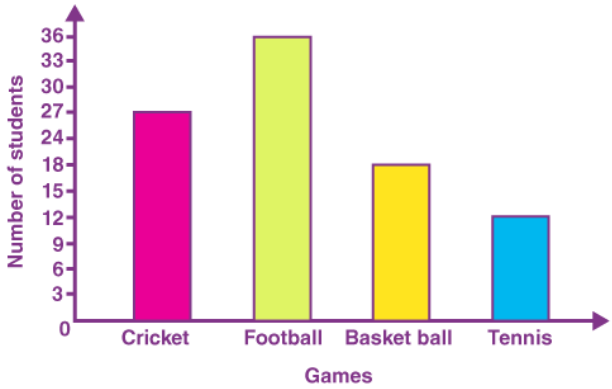
Multivariate Analysis

- Multivariate analysis involves the simultaneous analysis of two or more variables.It aims to understand the relationships and interactions between multiple variables. Multivariate techniques include regression analysis, factor analysis, principal component analysis, and clustering algorithms. Visualization techniques like scatter plots, heatmaps, and 3D plots are also used in multivariate analysis.
- **Example:** Examining the relationship between both height and weight of individuals or analyzing the performance of students considering multiple subjects are examples of multivariate analyses.

Methods

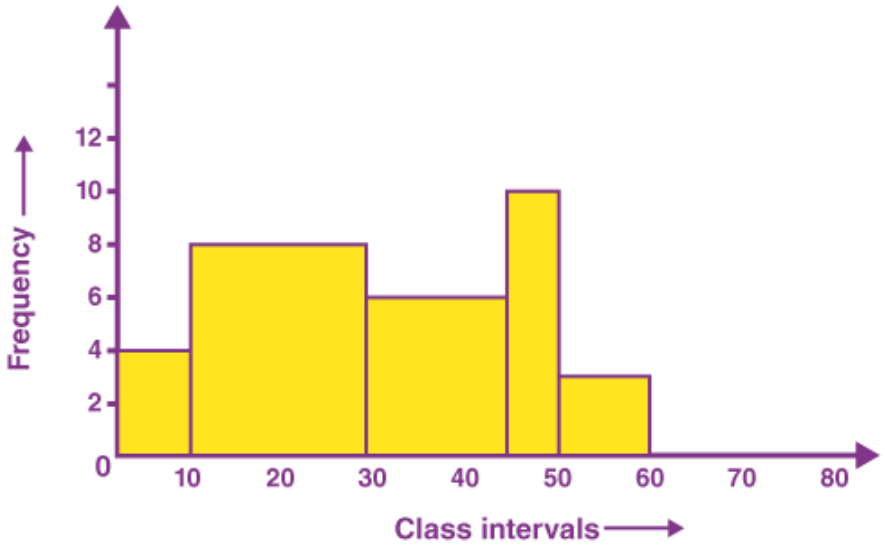
Bar Charts:

Represent data using rectangular bars, where the length of each bar corresponds to the value it represents. Useful for comparing values across categories.



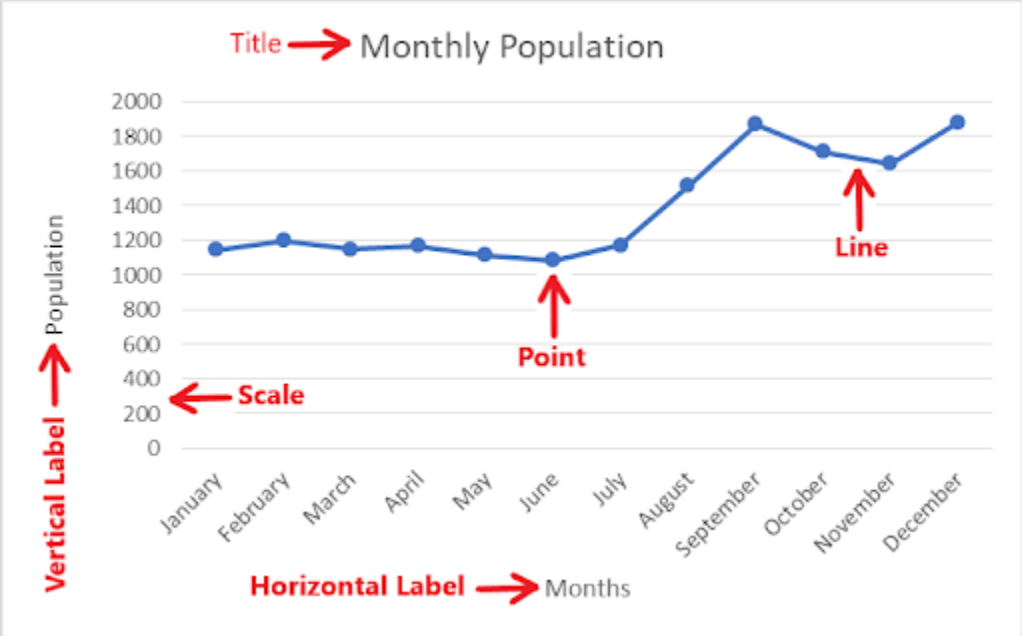
Histograms:

Display the distribution of a continuous variable by dividing it into bins and counting the number of occurrences in each bin. Provide insights into the data's shape and central tendencies.



Line Charts:

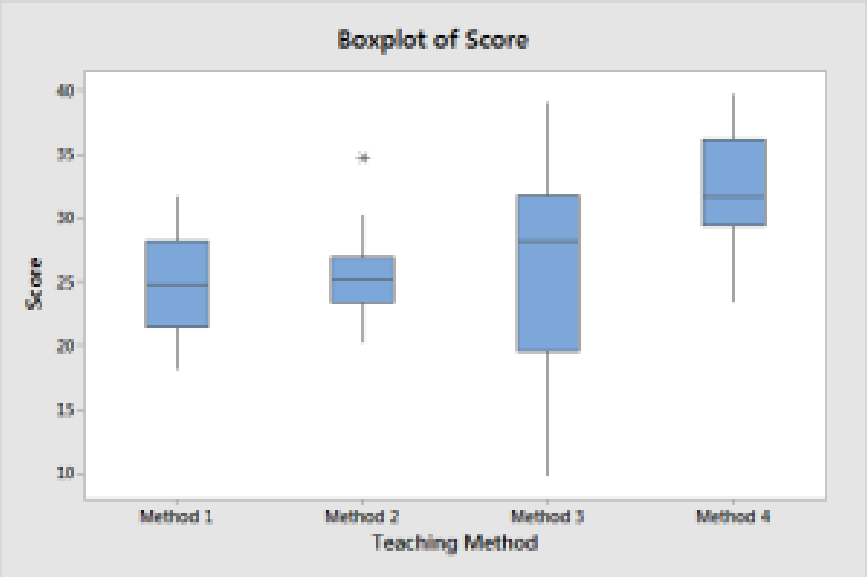
Connect data points with lines, illustrating trends over a continuous variable (usually time). Effective for showing patterns and changes in data over time.



Box Plots (Box-and-Whisker Plots):

Display the distribution of a dataset's summary statistics, including median, quartiles, and outliers, along a single axis. Useful for identifying the spread and skewness of the data.

- Box plots provide a visual summary of the data with which we can quickly identify the average value of the data, how dispersed the data is, whether the data is skewed or not (skewness).
- The Median gives you the average value of the data.



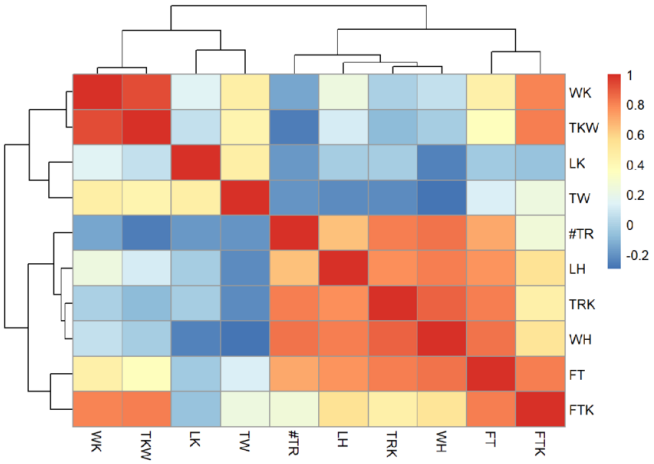
Dot Plots:

Represent individual data points with dots along a single axis. Useful for showing the distribution and density of points.

Heatmap

One of the best ways to checkout potential relationships or correlations amongst the different data attributes is to leverage a ***pair-wise correlation matrix*** and depict it as a ***heatmap***.

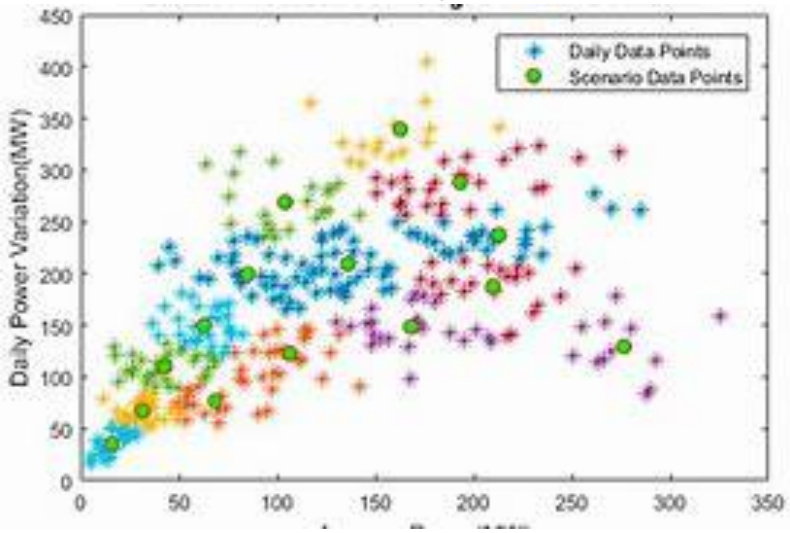
The gradients in the heatmap vary based on the strength of the correlation and you can clearly see it is very easy to spot potential attributes having strong correlations amongst themselves.



Scatter plots

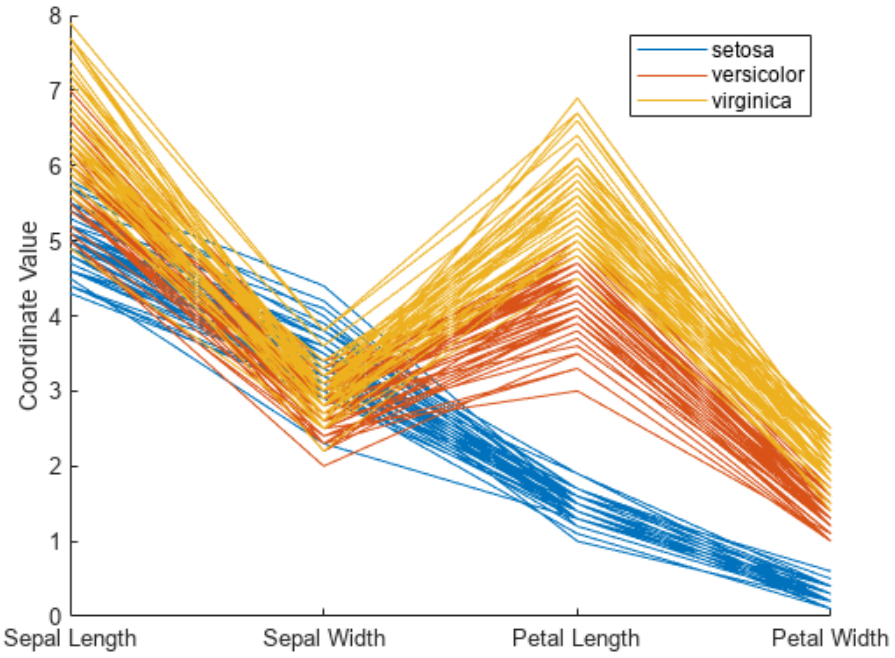
Another way to visualize the same is to use pairwise ***scatter plots*** amongst attributes of interest.The gradients in the heatmap vary based on the strength of the correlation and you can clearly see it is very easy to spot potential attributes having strong correlations amongst themselves.

A **scatter plot** displays 2-D data points using Cartesian coordinates. A third dimension can be added using different colors or shapes to represent different data points. A 3-D scatter plot uses three axes in a Cartesian coordinate system. If it also uses color, it can display up to 4-D data points. For data sets with more than four dimensions, scatter plots are usually ineffective. The **scatter-plot matrix** technique is a useful extension to the scatterplot. For an n-dimensional dataset, a scatter-plot matrix is an n × n grid of 2-D scatter plots that provides a visualization of each dimension with every other dimension. The scatter-plot matrix becomes less effective as the dimensionality increases.



Parallel coordinates

Another popular technique, called **parallel coordinates**, can handle higher dimensionality. To visualize n-dimensional data points, the parallel coordinates technique draws n equally spaced axes, one for each dimension, parallel to one of the display axes. A data record is represented by a polygonal line that intersects each axis at the point corresponding to the associated dimension value. A major limitation of the parallel coordinates technique is that it cannot effectively show a data set of many records. Even for a data set of several thousand records, visual clutter and overlap often reduce the readability of the visualization and make the patterns hard to find.



Association Analysis

- Frequent patterns are patterns (e.g., itemsets, subsequences, or substructures) that appear frequently in a data set.
- Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data.
- Moreover, it helps in data classification, clustering, and other data mining tasks.Thus, frequent pattern mining has become an important data mining task and a focused theme in data mining research.
- Market Basket Analysis is a data mining technique used to discover associations and relationships between items in a dataset.

Market Basket Analysis

- This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets".
- The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.
- For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket?
- This information can lead to increased sales by helping retailers do selective

marketing and plan their shelf space.

Given a set of transactions, we can find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Support Count(□) – Frequency of occurrence of a itemset.

Here, □ ({Milk, Bread, Diaper})=2

Frequent Itemset – An itemset whose support is greater than or equal to minsupport threshold

Association Rule – An implication expression of the form X -> Y, where X and Y are any 2 itemsets.

Example: {Milk, Diaper}->{Beer}

Rule Evaluation Metrics

1. **Support(s)**

   The number of transactions that include items in the {X} and {Y} parts of the rule as a percentage of the total number of transaction.It is a measure of how frequently the collection of items occur together as a percentage of all transactions.

Support (X) = (Number of transactions containing X) / (Total number of transactions)

2. **Confidence(c)**

   It is defined as the proportion of cases in which the association rule holds true, or in other words, the percentage of times that the items in the antecedent (the "if" part of the rule) appear in the same transaction as the items in the consequent (the "then" part of the rule).

   Confidence(X => Y) = (Number of transactions containing X and Y) / (Number of transactions containing X)

3. **Lift(l)**

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

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

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.(ie, Knowing the occurrence of one item does not provide any information about the occurrence of the other.)
- **Lift>1**: Indicates a positive correlation. The items are more likely to be bought together than if they were bought independently. The higher the lift, the stronger the association.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

➔ Rule **support and confidence** are two measures of rule interestingness.
➔ They respectively reflect the usefulness and certainty of discovered rules. Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold.**
➔ These thresholds can be set by users or domain experts. Additional analysis can be performed to discover interesting statistical correlations between associated items.

Example:

Consider the rule, **Apple** —-> **Orange**

Support (Apple) = 2/4

Support (Orange) = 3/4

Support (Apple, Orange) = 2/4

Confidence (Apple —> Orange) = Support (Apple, Orange) / Support (Apple)

= (2/4) / (2/4)
= 1 means 100%

4. Leverage

Leverage computes the probability of A and B occurring together and the frequency that would be expected if A and B were independent.

Ex.
Leverage(Bread →Milk) = 0.4-(0.8*0.6) = 0.08
Leverage(Milk →Jam) = 0.2-(0.6*0.2) = 0.08

- Leverage is similar to lift but easier to interpret since it ranges from -1 to 1 while lift ranges from 0 to infinity.
- A leverage value of 0 indicates independence.

5. Conviction

- *Conviction helps to judge if the rule happened to be there by chance or not.*

Ex.
Conviction(Bread →Milk) = (1-0.4)/(1–0.5) = 1.2
Conviction(Milk →Jam) = (1-0.2)(1–0.33) = 1.19

- A high conviction value means that the consequent is highly dependent on the antecedent (A). It can be interpreted as lift. If items are independent, the conviction is 1.

Frequent Itemsets, Closed Itemsets, and Association Rules

- **The rule A→B holds in the D, if both a minimum support threshold (min_sup) and a minimum confidence(min_conf) exists.**
-

Association Rule Mining

- Two step process
  - **Find all the frequent itemset:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
  - **Generate strong association rules from the frequent itemset**: By definition, these rules must satisfy minimum support and minimum confidence

A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support (min sup) threshold, especially when min sup is set low. This is because if an itemset is frequent, each of its subsets is frequent as well. A long itemset will contain a combinatorial number of shorter, frequent sub-itemsets. For example, a frequent itemset of length 100, such as {a1, a2,..., a100}, contains 100 C 1 =100 frequent 1-itemsets: {a1}, {a2}, . . . , {a100}; 100 C 2 frequent itemsets and so on. Thus the total number of frequent itemsets that it contains is 2^100-1 frequent itemsets.

This is too huge a number of itemsets for any computer to compute or store. To overcome this difficulty, we introduce the concepts of closed frequent itemset and **maximal frequent itemset**. An itemset X is **closed** in a data set Difthereexistsno proper super-itemset Y^5 such that Y has the same support count as X in D. An itemset X is a **closed frequent itemset** in set D if X is both closed and frequent in

D. An itemset X is a **maximal frequent itemset** (or max-itemset) in a data set D if X is frequent, and there exists no super-itemset Y such that X⊂Y and Y is frequent in D.

Apriori Algorithm

- As an increment of the number of item sets leads to an exponential growth of the association rule, it is not efficient to calculate all rules. If we have many items, calculating all rules can be computationally expensive.
- The Apriori algorithm is designed to find only important association rules with minimum computation.
- The Apriori algorithm was proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties.
- Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k + 1)- itemsets.

- First,the set of frequent 1-item sets is found by scanning the database to accumulate the count for each item(C1), and collecting those items that satisfy minimum support.
- The resulting set is denoted by L1.
- Next, L1 is used to find L2, the set of frequent - itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found.
- The finding of each Lk requires one full scan of the database.
- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

Apriori Property

All non-empty subsets of a frequent item set must also be frequent. If an itemset is infrequent, all its supersets will be infrequent.

- The Apriori property is based on the following observation.
- By definition, if an itemset I does not satisfy the minimum support threshold, min sup, then I is not frequent, that is, P(I) < min sup.

- If an item A is added to the itemset I, then the resulting itemset (i.e.,I∪A)cannot occur more frequently than I.
- Therefore, I∪A is not frequent either, that is, P(I∪A) < min sup.
- This property belongs to a special category of properties called **anti monotonicity** in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called anti monotonicity because the property is monotonic in the context of failing a test.

Steps for Apriori Algorithm

- **The join step:**To find Lk,a set of candidate k-itemsets is generated by joining Lk−1 with itself. This set of candidates is denoted Ck.
- Let l1 and l2 be itemsets in Lk−1. The notation li[j] refers to the jth item in li (e.g., l1[k − 2] refers to the second to the last item in l1).
- For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.
- For the(k−1)-itemset, li, this means that the items are sorted such that li[1]<li[2]<···<li[k−1]. The join, Lk−1✶Lk−1,is performed, where members of Lk−1 are joinable if their first (k − 2) items are in common. That is, members l1 and l2 of Lk−1 are joined if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ··· ∧ (l1[k − 2] = l2[k − 2]) ∧(l1[k − 1] < l2[k − 1]).
- The condition l1[k − 1] < l2[k − 1] simply ensures that no duplicates are generated.Theresultingitemsetformedbyjoiningl1andl2is{l1[1],l1[2],...,l1[k− 2],l1[k− 1],l2[k− 1]}.
- **The prune step**: Ck is a superset of Lk , that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in Ck.
- A database scan to determine the count of each candidate in Ck would result in the determination of Lk(i.e.,all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk).
- Ck , however, can be huge, and so this could involve heavy computation. To reduce the sizeof Ck, the Apriori property.

Algorithm:

In Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.

1) In the first iteration of the algorithm, each item is taken as a1-itemsets candidate.

The algorithm will count the occurrences of each item.

2) Let there be some minimum support, min_sup (eg 2). The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

3) Next, 2-itemset frequent items with min_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.

4) The2-itemsetcandidatesareprunedusingmin-supthresholdvalue.Nowthetable will have 2 –itemsets with min-sup only.

5) The next iteration will form 3 –itemsets using the join and prune step. This iteration will follow an anti monotone property where the subsets of 3-itemsets, that is the 2 – itemset subsets of each group fall in min_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.

6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.

Advantages

- Easy to understand algorithm
- Join and Prune steps are easy to implement on large itemsets in large databases

Disadvantages

- It requires high computation if the itemsets are very large and the minimum support is kept very low.
- The entire database needs to be scanned multiple times.
- Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets

Applications of Apriori Algorithm

- In Education Field: Extracting association rules in data mining of admitted students through characteristics and specialties.
- In the Medical Field: For example Analysis of the patient's database.

- In Forestry: Analysis of probability and intensity of forest fire with the forest fire data.
- Apriori is used by many companies like Amazon in the Recommender System and by Google for the auto-complete feature.

Methods to Improve Apriori's Efficiency

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

Frequent Pattern Growth Tree Algorithm(FP Tree Algorithm)

The two primary drawbacks of the Apriori Algorithm are:-

1. At each step, candidate sets have to be built.
2. To build the candidate sets, the algorithm has to repeatedly scan the database.

- These two properties inevitably make the algorithm slower.
- To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm.
- It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a tree Data Structure.

★ A frequent pattern is generated without the need for candidate generation.
★ FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.
★ This tree structure will maintain the association between the itemsets.
★ The database is fragmented using one frequent item.
★ This fragmented part is called a "pattern fragment".
★ The itemsets of these fragmented patterns are analyzed. Thus with this method,

the search for frequent itemsets is reduced comparatively.

★ ***The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.***

FP Tree

- The frequent-pattern tree (FP-tree) is a compact data structure that stores quantitative information about frequent patterns in a database.
- Each transaction is read and then mapped onto a path in the FP-tree. This is done until all transactions have been read.
- Different transactions with common subsets allow the tree to remain compact because their paths overlap.
- A frequent Pattern Tree is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the item set.
- The root node represents null, while the lower nodes represent the item sets. The associations of the nodes with the lower nodes, that is, the item sets with the other item sets, are maintained while forming the tree.

Frequent Pattern Algorithm Steps

The frequent pattern growth method lets us find the frequent patterns without candidate generation.

Steps:

#1) The first step is to scan the database to find the occurrences of the item sets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.

#2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

#3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The item set with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.

#4) The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.
This means that the common itemset is linked to the new node of another itemset in this transaction.

#5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.

#6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length1. From this, traverse the path in the FP Tree. This Path or paths are called a conditional pattern base. Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

#7) Construct a Conditional FP Tree, which is formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.

#8) Frequent Patterns are generated from the Conditional FP Tree.

- **Conditional Pattern Base:** The Conditional Pattern Base is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree.
- From the Conditional Frequent Pattern tree, the Frequent Pattern rules are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item

Example:

FP GROWTH TREE ALGORITHM

Transaction ID	Items
T1	E,K,M,N,O,Y
T2	D,E,K,N,O,Y
T3	A,E,K,M
T4	C,K,M,U,Y
T5	C,E,I,K,O,O

Minimum support count= 3

Step 1:

Compute the frequency of each item

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

Step 2:

A **Frequent Pattern Set (L)** is built which will contain all the elements whose frequency is greater than or equal to the minimum support count.

Here minimum support count =3

Transaction ID	Frequency
E	4
K	5
M	3
O	3
Y	3

Step 3:

Arrange the transactions in descending order of frequency in order to generate L. And the transactions with the same frequency should be considered based on FIFO.

So, **L= {K:5, E:4, M:3, O:3, Y:3}**

Step 4:

Build **Ordered- Item** set based on **L**

Consider the order of items in the L and arrange the items in the same order in each transaction.

L= {K:5, E:4, M:3, O:3, Y:3}

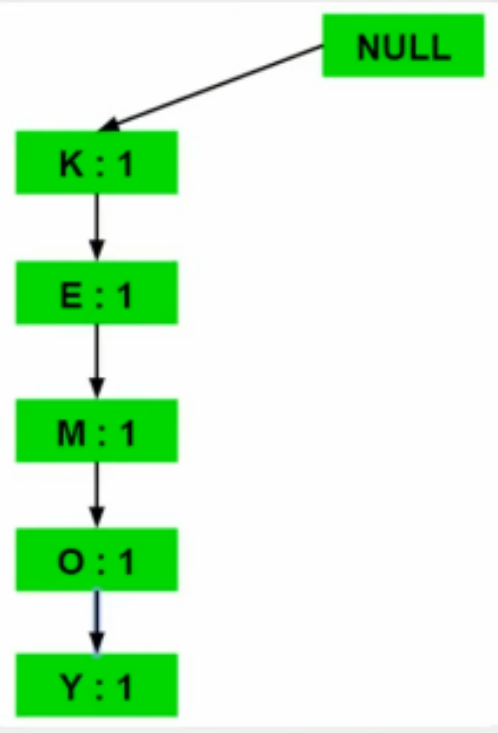
Transaction ID	Items	Ordered- Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M

T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O,O	K,E,O

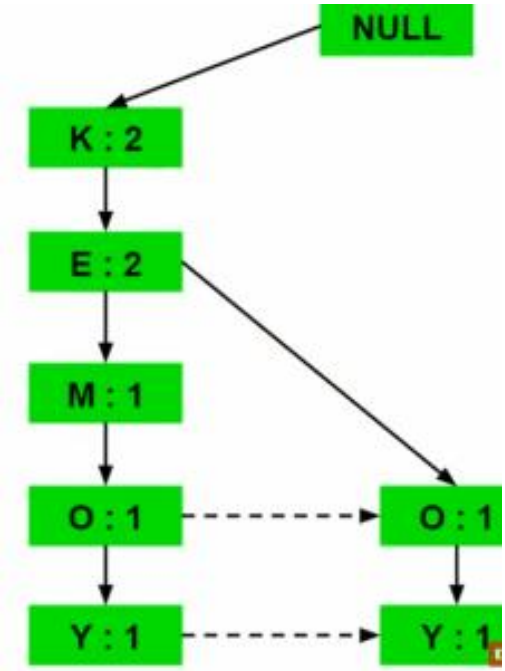
Step 5:

Insert all the transactions in the Tree.

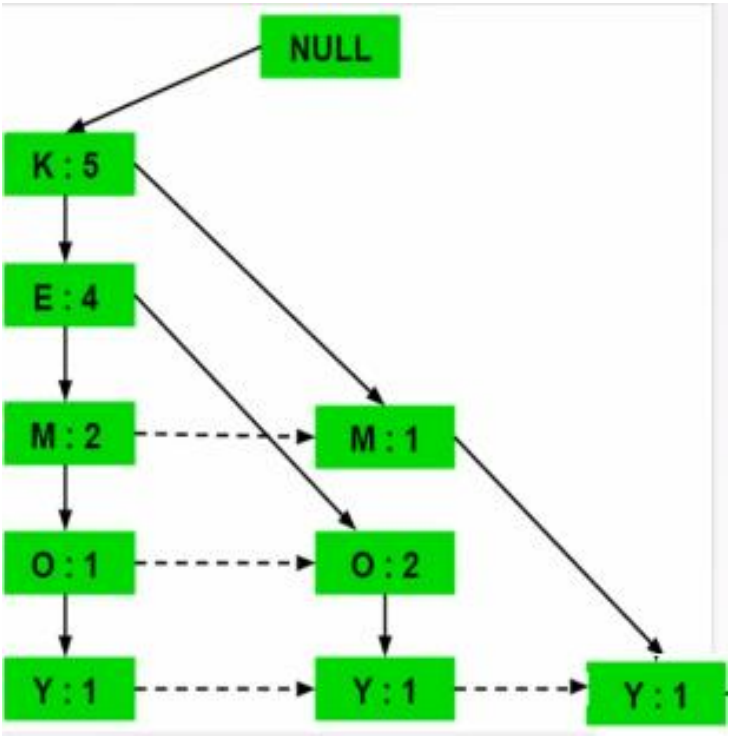
- **Inserting the set K, E, M, O, Y**



- **Inserting the set K, E, O, Y**



- **Finally, we will get the below given tree**



Step 6:

Compute **Conditional Pattern Base** for each item. It is the path labels of all the paths which lead to any node of the given item in the frequent pattern tree.

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {k :1}}
E	{{K : 4}}
K	

Step 7:

Now for each item, a Conditional **Frequent Pattern Tree** is built. It is done by taking the set of elements which is common in all the paths in the **Conditional Pattern Base** of that item and calculating its support count by summing the support counts for all the paths in the conditional pattern base.

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K :1}}	{K : 3}
E	{{K : 4}}	{K : 4}
K		

Step 8:

From the conditional Frequent Pattern Tree, the **Frequent Pattern Rules** are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding items.

Items	Frequent Patterns Generated

Y	{<K,Y : 3>}
O	{<K,O :3>, <E,O :3>, <K,E,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

Step 9:

For each frequent item set(pattern), possible rules can be generated and check whether it meets the minimum threshold.

i) K—>Y

Y—>K

Calculate the confidence of both the rules and repeat the same for others.

Select the rule which satisfies the given confidence threshold.

Advantages of FP Growth Algorithm

1. This algorithm needs to scan the database only twice when compared to Apriori which scans the transactions for each iteration.
2. The pairing of items is not done in this algorithm and this makes it faster.
3. The database is stored in a compact version in memory.
4. It is efficient and scalable, forming both long and short frequent patterns.

Disadvantages of FP-Growth Algorithm

1. FP Tree is more cumbersome and difficult to build than Apriori.
2. It may be expensive.
3. When the database is large, the algorithm may not fit in the shared memory.

FP Growth vs Apriori

FP Growth	Apriori
Pattern Generation	
FP growth generates pattern by constructing a FP tree	Apriori generates patterns by pairing the items into singletons, pairs and triplets.
Candidate Generation	
There is no candidate generation	Apriori uses candidate generation
Process	
The process is faster as compared to Apriori. The runtime of the process increases linearly with an increase in the number of itemsets.	The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets
Memory Usage	
A compact version of database is saved	The candidates combinations are saved in memory

Major Steps to Mine FP-tree

1) Construct conditional pattern base for each node in the FP-tree

2) Construct conditional FP-tree from each conditional pattern-base

3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far

▪ If the conditional FP-tree contains a single path, simply enumerate all the patterns

Properties of FP-tree for Conditional Pattern Base Construction

• **Node-link property:** For any frequent item ai , all the possible frequent patterns thatcontainaicanbeobtainedbyfollowingai'snode-links,startingfromai'shead in the FP-tree header

• **Prefix path property:** To calculate the frequent patterns for a node ai in a path P, only the prefix sub path of ai in P need to be accumulated, and its frequency count should carry the same count as node ai.