A Comparative Study of Classification Techniques in Data Mining

[1]Kishansingh Rajput,[2]Bhavesh A. Oza
[1]Student,[2]Professor
[1]Computer Engineering Department
[1]L. D. College of Engineering (Gujarat Technological University), Ahmedabad,India

Abstract :In today's world of information, tons of data exist for everyone. The processing power and storage power has increased dramatically in last decade and now we have capacity to store this large amount of data. The motivation behind storage and processing of this data is to produce knowledge from them. Classification algorithms are widely used to extract knowledge from available data. This paper focuses primarily on some most frequently used classification techniques in data mining. In this paper most popular classification techniques like Decision Tree, K-Nearest Neighbor, Apriori and Support Vector Machine are discussed, and compared on the basis of their performance.

IndexTerms **- Decision tree, Apriori, Support Vector Machine, KNN, Classification, Data mining**

I. INTRODUCTION

In today's world data is generated or collected by almost every device, we use in our daily life, this data is very useful if we can effectively use it for gaining knowledge. Data Mining or Knowledge Discovery is needed to make sense and use of data. Knowledge Discovery in Data is the process of identifying valid, new, potentially useful and ultimately understandable patterns in data. Data mining is not just collection and managing data; it also includes analysis of data for deriving useful knowledge from it. Machine learning can often be successfully applied to practical problems, improving the efficiency of systems and the designs of machines. Machine learning mainly does predictions whether it can be prediction of continuous values or prediction of class amongst set of pre-defining classes. There are various techniques for prediction of class; among them most popular techniques are Decision Tree algorithm, K-nearest neighbor algorithm, apriori algorithm and Support Vector Machine. We will discuss these four algorithms in detail.

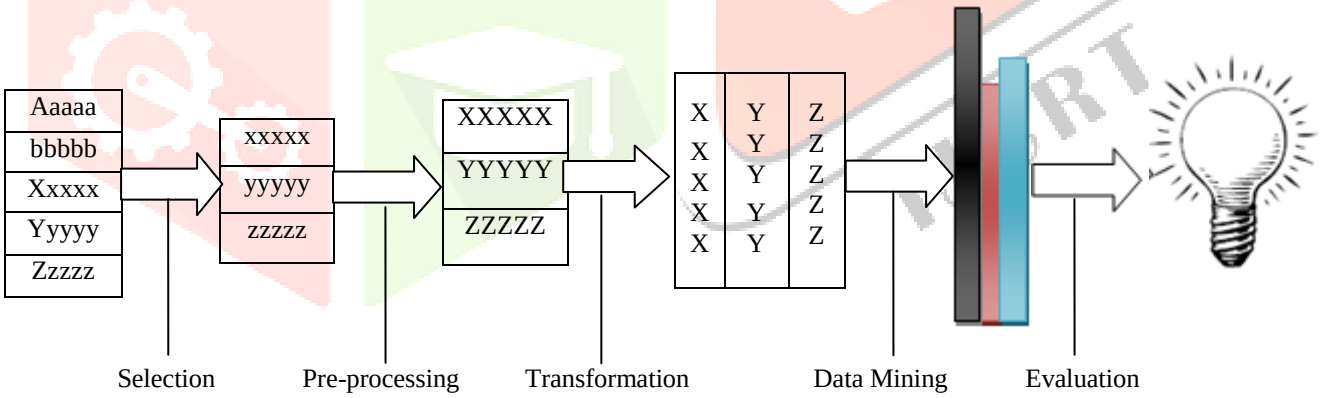

Fig: Process of Data Mining

II. CLASSIFICATION

Some time we need to classify the data into some definite classes. A doctor needs to classify the stage of cancer based on the symptoms; it can be Stage-1, Stage-2 or Stage-3. Similarly he may need to classify a tumor, whether it is malignant or not and all these can be done using Machine Learning or so called classification techniques. Classification algorithms take data set as input and place given record in one of the pre-defined classes.

One another model in data mining is prediction which takes attributes value as input and gives value of event which is most likely, based on the given value of attributes according to the previous knowledge. For example, it may take different attributes value of house to be sold in particular area and gives us its Price as output. Prediction model predicts continuous values which can be anything. Predictions are often (but not always) about the future. Prediction models are mathematical or statistical like, linear regression, non-linear regression, logistic regression or Neural networks etc. On the other hand Classification problems can be seen as prediction of class labels where number of classes is fixed and pre-defined.

III. CLASSIFICATION ALGORITHMS

Systems that construct classifiers are one of the commonly used tools in data mining. Such systems take as input a collection of cases, each belonging to one of a small number of classes and ascribed by its values for a fixed set of attributes, and output a classifier that can accurately predict the class to which a new case belongs.

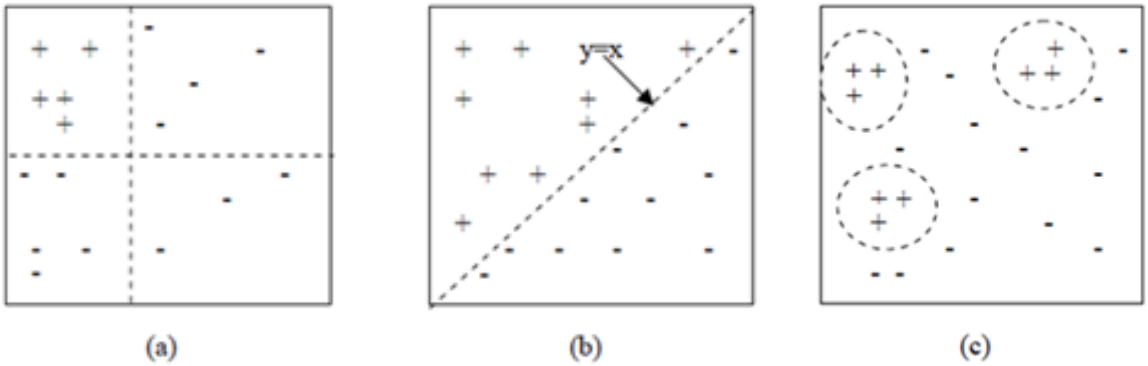

Fig: (a) Asymmetric Classification, (b) Symmetric Classification, (c) Clustering

IV. DECISION TREE ALGORITHMS

A decision tree is a tree like structure, where rectangles are used to denote internal nodes and ovals are used to denote leaf nodes. All internal nodes can have two or more child nodes. All internal nodes contain splits, which test the value of an expression of the attributes. Connections from an internal node to its children are labeled with distinct outcomes of the test and each leaf node has a class label associated with it.

Given a set D of cases, this algorithm first grows an initial tree using the divide-and-conquer algorithm as follows:

- If all the cases in D belong to the same class or D is small, the tree is a leaf labeled with the most frequent class in D.
- Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test the root of the tree with one branch for each outcome of the test, partition D into corresponding subsets D1, D2, . . . according to the outcome for each case, and apply the same procedure recursively to each subset.[15]

Attributes can be either numeric or nominal and this determines the format of the test outcomes. For a numeric attribute A they are $\{A \leq h, A > h\}$ where the threshold h is found by sorting D on the values of A and choosing the split between successive values that maximizes the criterion above. An attribute A with discrete values has by default one outcome for each value, but an option allows the values to be grouped into two or more subsets with one outcome for each subset.[15] Decision trees are usually unvaried since they are based on a single feature at each internal node. Most decision tree algorithms cannot perform well with problems that require diagonal partitioning.[6]

Sample:

Let's assume that a student is making his weekend plans and finds out that his parents might come to town. He'd like to have plans in place, but there are a few unknown factors that will determine what he can, and can't, do. So it's time for a decision tree.

First, he will draw his decision box. This is the box that includes the event that starts his decision tree. In this case it is his parents coming to town. Out of that box, he has a branch for each possible outcome. In this example, it's easy: yes or no - either his parents come or they don't. His parents love the movies, so if they come to town, he'll go to the cinema. Since the goal of the decision tree is to decide his weekend plans, he has an answer. But what about if his parents don't come to town? He can go back up to the 'no branch' from the decision box and finish his decision tree. If his parents don't come to the town, he needs to decide what he is going to do. As he thinks of options, he realizes the weather is an important factor. Weather becomes his next box. Since it's spring time, he knows it will either be rainy, sunny, or windy. Those three possibilities become his branches. If it's sunny or rainy, he knows what he'll do - play tennis or stay in, respectively. But, what if it's windy? If it's windy, he wants to get out of the house, but he probably won't be able to play tennis. He could either go to the movies or go shopping. What will determine if he go shopping or go see a movie? Money is another factor. If he has enough money in the pocket, he would prefer shopping or else he would opt for movie. So this way he can decide between different options he has for the weekend using the decision tree.

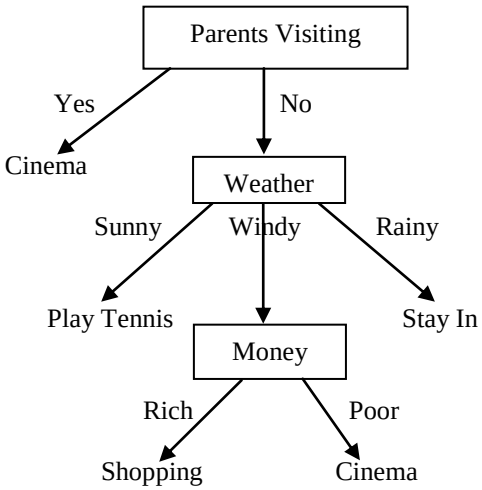

Fig: Decision Tree Algorithm

Characteristics of Decision Tree Algorithm:

Decision Tree algorithm makes use of tree data structure to predict the result i.e. class. Generated Tree, decides the prediction. As others, this algorithm also has some pros as well as cons they are as follow:

Pros:

- Simple for understanding and relate properly to a set of production rules.
- Decision trees can be efficiently approached for practical problems.
- No prior predictions about the behaviour of the data are to be made.
- Efficient enough to create models with data containing numerical and also categorical values.

Cons:

- *Empty branches:* Constructing tree with meaningful value is one of the crucial steps for rule generation by this algorithm.
- *Insignificant branches:* Numbers of selected discrete attributes create equal number of potential branches to build a decision tree. But all of them are not significant for classification task. These insignificant branches not only reduce the usability of decision.
- *Over fitting:* Over fitting happens when algorithm model picks up data with uncommon characteristics. This cause many fragmentations in the process distribution. Statistically insignificant nodes with very few samples are known as fragmentations. Generally this algorithm constructs trees and grows it branches just deep enough to perfectly classify the training examples'. This strategy performs well with noise free data. But most of the time this approach over fits the training examples with noisy data.[9] Currently there are two approaches widely used to bypass this over-fitting in decision tree learning. Those are:
    1. If tree grows very large, stop it before it reaches maximal point of perfect classification of the training data.
    2. Allow the tree to over-fit the training data then post-prune tree.[9]

V. K-NEAREST NEIGHBOUR ALGORITHM

Given an example the nearest-neighbor method first determines the k most similar examples in the training data and then determines the prediction based on the class values associated with these k examples, where k is a user specified parameter. The simplest scheme is to predict the class value that occurs most frequently in the k examples, while more sophisticated schemes might use weighted voting, where those examples most similar to the example to be classified are more heavily weighted. People naturally use this type of technique in everyday life.[13]K-Nearest neighbor algorithm (KNN) is one of the supervised learning algorithms that have been used in many applications in the field of data mining, statistical pattern recognition and many others. It follows a method for classifying objects based on closest training examples in the feature space.[5]

The k-nearest neighbors' algorithmis amongst the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors. *k*is a positive integer, typically small. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes.

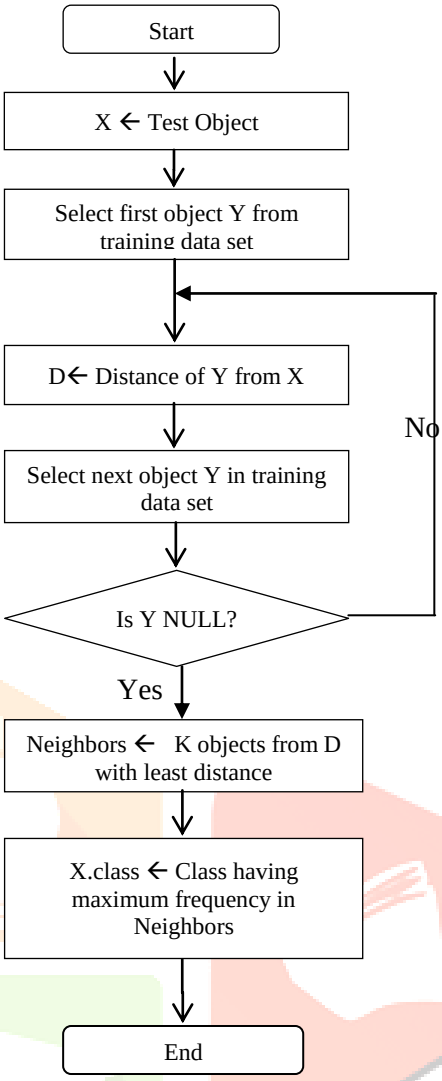

Fig: KNN Algorithm

Sample:

Let a person needs to predict whether it will rain or not on a particular day in a particular area based on conditions of humidity, temperature and pressure in that area. He can use K-Nearest Neighbor algorithm to predict this result. He has to classify given record into two classes one is 'It will rain' and another one being 'No rain'. Let he has training data as below

Table 1: Training data for Prediction of rain

#	Humidity (%)	Temp. (Celsius)	Pressure (bar*100)	Result (Yes/No)
1	30	30	110	No
2	80	40	101	No
3	90	32	97	Yes
4	70	28	117	Yes

The current temperature is 32 degree Celsius, Humidity 58% and pressure is 1.00 bar (i.e. 100 after normalization) then he can predict whether it will rain or not using K-Nearest Neighbour algorithm, for this he need to first calculate the distance of his records point from the record points of training data set.

Calculation of Distance with X = (58, 32, 100)

Table 2: Calculation of Distance value for KNN

#	Humidity (%)	Temp. (Celsius)	Pressure (bar*100)	Distance from (x)
1	30	30	110	888
2	80	40	101	549
3	90	32	97	1033
4	70	28	117	609

Now if he takes K as 3 then, 3 nearest neighbours of point X are 1,2 and 4 and among them 1, 2 says 'No rain' whereas only 4 says 'it will rain'. Hence his prediction will be 'No rain' for the given situation.

Characteristics of K-Nearest Neighbor Algorithm:

K Nearest Neighbor algorithm is a supervised machine learning algorithm, it requires training data and while prediction it uses the pre-defined classes of training data. This algorithm uses distance approach to match the new record with any record of training record.

Pros:

- *K*NN classification is an easy to understand and easy to implement classification technique.
- Despite its simplicity, it can perform well in many situations. In particular, a well-known result by Cover and Hart [15] shows that the error of the nearest neighbor rule is bounded above by twice the Bayes error under certain reasonable assumptions. Also, the error of the general *k*NN method asymptotically approaches that of the Bayes error and can be used to approximate it.[15]

Cons:

- If *k* is too small, then the result can be sensitive to noise points. On the other hand, if *k* is too large, then the neighborhood may include too many points from other classes.
- The simplest method is to take a majority vote, but this can be a problem if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the class of the object.[15]

VI. SUPPORT VECTOR MACHINE

Given SVM was first introduced by Vapnik and has been very effective method for regression, classification and general pattern recognition. It is considered a good classifier because of its high generalization performance without the need to add a priori knowledge, even when the dimension of the input space is very high. It is considered a good classifier because of its high generalization performance without the need to add a priori knowledge, even when the dimension of the input space is very high. The aim of SVM is to find the best classification function to distinguish between members of the two classes in the training data.[15]

A Support Vector Machine (SVM) separates the data into two categories of performing classification and constructing an N-dimensional hyper plane. These models are closely related to neural networks. In fact, this model uses a sigmoid kernel function which is equivalent to a two-layer, perceptron neural network. Because there are many such linear hyper planes, what SVM additionally guarantee is that the best such function is found by maximizing the margin between the two classes. Intuitively, the margin is defined as the amount of space, or separation between the two classes as defined by the hyper plane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyper plane. Having this geometric definition allows us to explore how to maximize the margin, so that even though there are an infinite number of hyper planes, only a few qualify as the solution to SVM.[15] To ensure that the maximum margin hyper planes are actually found, an SVM classifier attempts to maximize the following function with respect to vector w and b:

$$L_P = \frac{1}{2} \parallel \vec{\mathbf{w}} \parallel - \sum_{i=1}^{t} \alpha_i y_i (\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_\mathbf{i} + b) + \sum_{i=1}^{t} \alpha_i$$

where t is the number of training examples, and αi , i = 1, . . . , t, are non-negative numbers such that the derivatives of L P with respect to αi are zero. αi are the Lagrange multipliers and L P is called the Lagrangian. In this equation, the vectors _w and constant b define the hyper plane.

Sample:

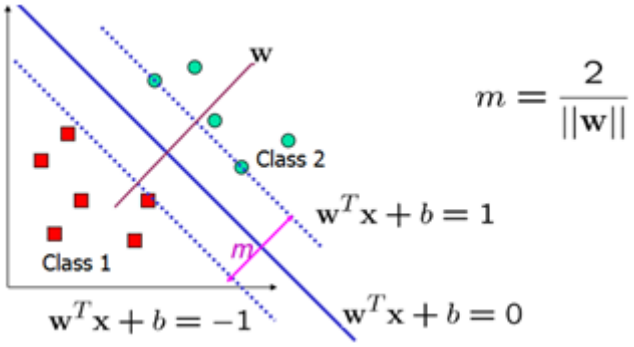

Fig: Support Vector Machine

Characteristics of Support Vector Machine Algorithm:

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. SVM works by forming hyper plane and accuracy of each hyper plane is very important for the final prediction. SVM is comparatively complex to implement.

Pros:

- By introducing the kernel, SVMs gain flexibility in the choice of the form of the threshold separating solvent from insolvent companies, which needs not be linear and even needs not have the same functional form for all data, since its function is non-parametric and operates locally. [1]
- Since the kernel **implicitly** contains a non-linear transformation, no assumptions about the functional form of the transformation, which makes data linearly separable, is necessary. The transformation occurs implicitly on a robust theoretical basis and human expertise judgment beforehand is not needed.[1]
- By choosing an appropriate generalization grade, SVMs can be robust, even when the training sample has some bias.[1]
- SVMs deliver a unique solution, since the optimality problem is convex. This is an advantage compared to Neural Networks, which have multiple solutions associated with local minima and for this reason may not be robust over different samples.[1]

Cons:

- The biggest limitation of the support vector approach lies in choice of the kernel.[8]
- Although SVMs have good generalization performance, they can be abysmally slow in test phase, a problem addressed in (Burges, 1996; Osuna and Girosi, 1998).
- The most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements of the required quadratic programming in large-scale tasks.

VII. APRIORI ALGORITHM

Given Apriori algorithm is the originality algorithm of Boolean association rules of mining frequent item sets, raised by R. Agrawa and R. Srikan in 1994. Apriori is a frequent pattern mining algorithm which classifies frequent patterns and infrequent patterns from bunch of possible item sets. The core principles of this theory are the subsets of frequent item sets are frequent item sets and the supersets of infrequent item sets are infrequent item sets. This theory is regarded as the most typical data mining theory of all the time.[4]

Apriori is a seminal algorithm for finding frequent item sets using candidate generation. It is characterized as a level-wise complete search algorithm using anti-monotonicity of item sets, "if an item set is not frequent, any of its superset is never frequent". By convention, Apriori assumes that items within a transaction or item set are sorted in lexicographic order. Let the set of frequent item sets of size k be F_k and their candidates be C_K. Apriori first scans the database and searches for frequent item sets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent item sets :[15]

1. Generate C_{k+1}, candidates of frequent item sets of size k +1, from the frequent item sets of size k.
2. Scan the database and calculate the support of each candidate of frequent item sets.
3. Add those item sets that satisfies the minimum support requirement to F_{k+1}.

The Apriori achieves good performance by reducing the size of candidate sets. However, in situations with very many frequent item sets, large item sets, or very low minimum support, it still suffers from the cost of generating a huge number of

candidate sets and scanning the database repeatedly to check a large set of candidate item sets. In fact, it is necessary to generate 2^100 candidate item sets to obtain frequent item sets of size 100.[4]

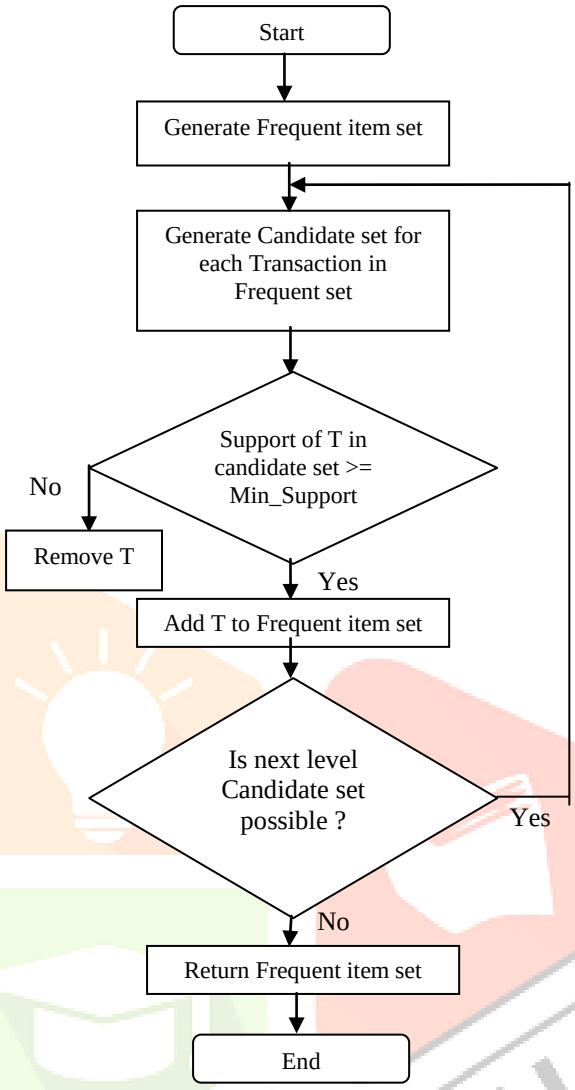

Fig: Apriori Algorithm

Sample:

Let a large supermarket tracks sales data by Stock-keeping unit (SKU) for each item, and thus is able to know what items are typically purchased together. Apriori is a moderately efficient way to build a list of frequent purchased item pairs from this data. Let the database of transactions consist of the sets {1,2,3,4}, {1,2,3,4,5}, {2,3,4}, {2,3,5}, {1,2,4}, {1,3,4}, {2,3,4,5}, {1,3,4,5}, {3,4,5}, {1,2,3,5}. Each number corresponds to a product such as "butter" or "water". The first step of Apriori is to count up the frequencies, called the supports, of each member item separately:

Table 3: Support table for different items

Item	Support
1	6
2	7
3	9
4	8
5	6

We can define a minimum support level to qualify as "frequent", which depends on the context. For this case, let min support = 4. Therefore, all are frequent. The next step is to generate a list of all 2-item pairs of the frequent items. Had any of the above items not been frequent, they wouldn't have been included as a possible member of possible 2-item pairs. In this way, Apriori

prunes the tree of all possible sets. In next step we again select only these items (now 2-item pairs are items) which are frequent (written in bold text):

Table 4: Candidate Item sets

Item set	Support
{1,2}	**4**
{1,3}	**5**
{1,4}	**5**
{1,5}	3
{2,3}	**6**
{2,4}	**5**
{2,5}	**4**
{3,4}	**7**
{3,5}	**6**
{4,5}	**4**

We generate the list of all triples of the frequent items (by connecting frequent pair with frequent single item).

Table 5: Frequent Item sets

Item set	Support
{1,3,4}	**4**
{2,3,4}	**4**
{2,3,5}	**4**
{3,4,5}	**4**

The algorithm will end here because the pair { 2, 3, 4, 5} generated at the next step does not have the desired support. Hence there are four Item sets which are most frequent according to our desired support i.e. 4.

Characteristics of Apriori Algorithm:

Apriori algorithm uses table data structure to find association rule. This algorithm is frequently used technique for association rule mining in data mining. It classifies frequent patterns.

Pros:

- Apriori Algorithm uses large item set property which makes it more accurate in pattern mining.
- Apriori Algorithm can be easily parallelized hence its speed can be increased using parallel operations.
- Apriori Algorithm is comparatively easy to understand, learn and implement.

Cons:

- Apriori Algorithm assumes that transaction database is memory resident.
- This algorithm requires many database scans which makes it little slow.

VIII. COMPARATIVE STUDY

Comparison of Decision tree algorithm and K-Nearest Neighbor algorithm:

Table 6: Theoretical comparison between DT and KNN.[11]

Property	Decision Tree	KNN
Accuracy in general	Good	Good
Speed of learning	V. Good	Excellent
Speed of classification	Excellent	Average
Tolerance to missing values	V. Good	Average
Tolerance to irrelevant attributes	V. Good	Good

Tolerance to noise	Good	Average
Attempts for incremental learning	Good	Excellent
Explanation ability/ transparency of knowledge/ classification	Excellent	Good
Support Multi Classification	Excellent	Excellent

Comparison of Decision tree algorithm and Support Vector Machine:

From the above comparison, it can be deduced that if Speed is a concern then KNN technique is better than decision tree technique but if the training data contains many missing values then decision tree will work better than KNN. KNN technique proves to be better in incremental learning but it is not as transparent as Decision tree technique.

Table 7: Simulation Results[12]

Algorithm (Total Instance 337)	Decision tree	Support Vector Machine
Correctly Classified Instances (% Value)	276 (81.8991)	269 (79.8220)
Incorrectly Classified Instances (% Value)	61 (18.1009)	68 (20.1780)
Time Taken (in seconds)	0.23	0.67
Kappa Statistic	0.6132	0.5302

After applying both decision tree algorithm and Support Vector machine, it looks like decision tree algorithm performed better in classification. Decision tree algorithm classifies more accurately while taking less amount of time. In SVM, accuracy and speed depends on selection of kernel function, if kernel function is selected properly then it can give more accurate results in less time, than Decision tree.

Table 8: Training and Simulation Error[12]

Algorithm (Total Instances 337)	Decision Tree	Support Vector Machine
Mean Absolute Error	0.2318	0.2018
Root Mean Squared Error	0.3772	0.4492
Relative Absolute Error (%)	50.5261	43.9748
Root Relative Squared Error (%)	78.7789	93.8086

The mean absolute error and relative absolute error of decision tree algorithm is more as compared to support vector machine while in case of root mean squared error and root relative squared error, they are more for Support vector machine.

IX. CONCLUSION

This brief paper discusses about what data mining is, and comparison of different classification techniques in data mining. Decision tree, K-nearest neighbor Support Vector Machine and apriorialgorithm(for classification of frequent patterns) are most frequently used classification algorithms in practice. After comparison it can be said that all four of them have their own advantages and disadvantages and they can best be applied in different situations.

REFERENCES

[1] Laura Auria and Rouslan A. Moro, Support Vector Machine (SVM0 as a Technique for Solvency Analysis, Berlin August 2008

[2] S. Muthuselvan and Dr. K. Soma Sundaram, A Survey of Sequence patterns in Data Mining Techniques, International Journal of Applied Engineering Research, 2015
[3] Babu C. Lakshamanan, ValarmathiShrinivasan, ChinnaiyanPonnuraja, Data Mining with Decision Tree to Evaluate the Pattern on Effectiveness of Treatment for Pulmonary Tuberculosis: A Clustering and Classification Techniques, Scientific Research Journal, Vol-3,
[4] Jiao Yabing, Research of an Improved Apriori Algorithm in Data
Mining Association Rules, International Journal of Computer and Communication Engineering, Vol. 2, No. 1, January 2013
[5] BendiVenkataRamana, Prof. M.Surendra Prasad Babu, Prof. N. B. Venkateswarlu, A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis. International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011
[6] Raj Kumar, Dr. Rajesh Verma, Classification Algorithms for Data Mining: A Survey, International Journal of Innovations in Engineering and Technology (IJIET), Vol. 1 Issue 2 August 2012
[7] Ashish Kumar Dogra, TanujWala, A Comparative Study of Selected Classification Algorithms of Data Mining, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.6, June- 2015
[8] Donghui Li, Mahmood R. Azimi-Sadjadi, and Marc Robinson, Comparison of Different Classification Algorithms for Underwater Target Discrimination, IEEE transactions on neural networks, vol. 15, no. 1, january 2004
[9] JignaAshish Patel, Classification Algorithms and Comparison in Data Mining, International Journal of Innovations & Advancement in Computer Science, ISSN 2347 – 8616, Volume 4, Special Issue, May 2015
[10] AbdolrashidRezvani and JavadHosseinkhani, Enhancing the Performance of BitApriori Algorithm in Data Mining using an Effective Data Structure, International Journal of Advanced Computer Science and Information Technology (IJACSIT) Vol. 4, No. 3, 2015, Page: 85-92, ISSN: 2296-1739
[11] OmkarArdhapure, GayatriPatil, DishaUdani and KamleshJetha, comparative study of classification algorithm for Text based categorization, IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308, 2016
[12] Devendra Kumar Tiwary, a comparative study of classification algorithms for credit card approval using weka, GALAXY International Interdisciplinary Research Journal, Vol.2 (3), MARCH (2014), ISSN 2347-6915
[13] Gary M. Weiss and Brian D. Davison, Data Mining, The Handbook of Technology Management, H. Bidgoli (Ed.), John Wiley and Sons, 2010.
[14] PreetiPatidar, JitendraDangra and M.K. Rawar, Decision Tree C4.5 algorithm and its enhanced approach for Educational Data Mining, Engineering Universe for Scientific Research and Management, (International Journal), Vol. 7 Issue 2, February 2015
[15] XindongWu , Vipin Kumar , J. Ross Quinlan , JoydeepGhosh , Qiang Yang , Hiroshi Motoda , Geoffrey J. McLachlan , Angus Ng , Bing Liu , Philip S. Yu , Zhi-Hua Zhou , Michael Steinbach , David J. Hand and Dan Steinberg, Top 10 algorithms in data mining, KnowlInfSyst (2008) 14:1–37, Springer-Verlag London Limited 2007
[16] QIANG YANG and XINDONG WU, 10 challenging problems in data mining research, International Journal of Information Technology & Decision Making, Vol. 5, No. 4 (2006) 597–604