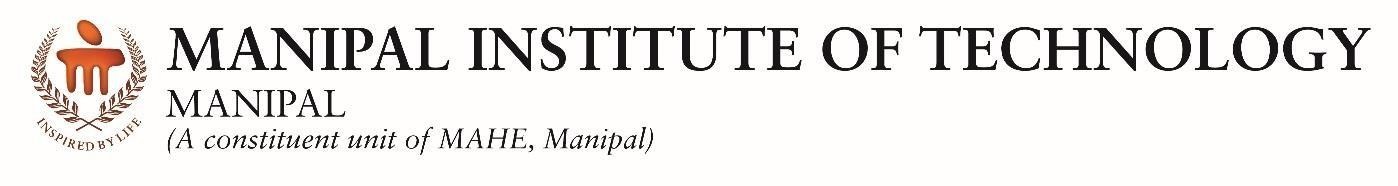
**DEPARTMENT OF INFORMATION & COMMUNICATION TECHNOLOGY**



#### MANIPAL INSTITUTE OF TECHNOLOGY MANIPAL

**CERTIFICATE**

This is to certify that Ms./Mr. …………………...…………………………………… Reg.No.

…..…………………… Section: ……… Roll No has satisfactorily completed the lab exercises

prescribed for Data Mining and Predictive Analysis Lab [ICT 3262] of Fifth Semester B. Tech. (CCE) Degree at MIT, Manipal, in the academic year July-December 2020.

Date: ……...................................

Signature of the faculty

**CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB NO.** | **TITLE** | **PAGE NO.** | **SIGNATURE** | **REMARKS** |
|  | COURSE OBJECTIVES, OUTCOMES AND EVALUATION PLAN | I |  |  |
|  | INSTRUCTIONS TO THE STUDENTS | II |  |  |
| 1 | NORMALIZE THE DATA USING MIN-MAX NORMALIZATION | 3 |  |  |
| 2 | CREATING NEW ARFF FILE | 18 |  |  |
| 3 | DATA PROCESSING TECHNIQUES ON DATA SET | 24 |  |  |
| 4 | DATA CUBE CONSTRUCTION – OLAP OPERATIONS | 37 |  |  |
| 5 | MINI PROJECT SYNOPSIS SUBMISSION | 43 |  |  |
| 6 | APRIORI ALGORITHM | 44 |  |  |
| 7 | K-MEANS ALGORITHM | 48 |  |  |
| 8 | DECISION TREE ID3 ALGORITHM FOR CLASSIFICATION | 50 |  |  |
| 9 | NAÏVE BAYES CLASSIFIER | 53 |  |  |
| 10 | MNIPROJECT : IMPLEMENTATION | 56 |  |  |
| 11 | MINIPROJECT : PROGRESS | 57 |  |  |
| 12 | MINIPROJECT : RESULT AND ANALYSIS | 58 |  |  |
|  | REFERENCES | 59 |  |  |

**Course Objectives**

* Familiarization with Rapid miner and InfoSphere
* Implementation of frequent pattern finding algorithms for association rule mining
* Implementation of clustering algorithms
* Implementation of the classification and predictive algorithms
* Implementation of a mini project on application of the above

#### Course Outcomes

At the end of this course, students will be able to

* Compare three main categories of data mining such as association rule mining, clustering, and classification
* Implement algorithms under each category
* Gain exposure to various tools and techniques for data miing and predictive analysis
* Implement a mini project by applying data mining algorithms on bench mark data sets

#### Evaluation plan

|  |
| --- |
| **Split up of 60 marks for Regular Lab Evaluation** |
| Lab 1 to Lab 4: (10 Marks ) [Record:8M Execution:4M Viva/Execution Test:8M]  Lab 5 : (20 Marks) [Report]  Lab 6 to Lab 10: (10 Marks)[Record:12M Execution:6M Viva/Execution Test:12M]  Mid Term: (20 Marks) |
| **End Semester Lab evaluation: 40 marks (Duration 2 hrs)** |
| Miniproject : 40 Marks |

**INSTRUCTIONS TO THE STUDENTS**

#### Pre- Lab Session Instructions

1. Students should carry the Lab Manual Book and the required stationery to every lab session
2. Be in time and follow the institution dress code
3. Must sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance
5. Adhere to the rules and maintain the decorum

#### In- Lab Session Instructions

* Follow the instructions on the allotted exercises
* Show the program and results to the instructors on completion of experiments
* Prescribed textbooks and class notes can be kept ready for reference if required

#### General Instructions for the exercises in Lab

* Implement the given exercise individually and not in a group.
* The programs should meet the following criteria:
  + Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
  + Comments should be used to give the statement of the problem.
  + Statements within the program should be properly indented.
* Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
* In case a student misses a lab, he/ she must ensure that the experiment is completed before the next evaluation with the permission of the faculty concerned.
* Students missing out lab on genuine reasons like conference, sports or activities assigned by the Department or Institute will have to take **prior permission** from the HOD to attend **additional lab** (with other batch) and complete it **before** the student goes on leave. The student could be awarded marks for the write up for that day provided he submits it during the **immediate** next lab.
* Students who fall sick should get permission from the HOD for evaluating the lab records. However attendance will not be given for that lab.
* Students will be evaluated only by the faculty with whom they are registered even though they carry out additional experiments in other batch.
* Presence of the student during the lab end semester exams is mandatory even if the student assumes he has scored enough to pass the examination
* Minimum attendance of 75% is mandatory to write the final exam.
* If the student loses his book, he/she will have to rewrite all the lab details in the lab record.
* Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.

#### THE STUDENTS SHOULD NOT

* Bring mobile phones or any other electronic gadgets to the lab.
* Go out of the lab without permission.

#### LAB NO. 1 Date

# Introduction

Weka (pronounced to rhyme with Mecca) is a workbench that contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and Make file-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:

* Free availability under the GNU General Public License.
* Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform
* A comprehensive collection of data preprocessing and modeling techniques
* Ease of use due to its graphical user interfaces

## Description:

Open the program. Once the program has been loaded on the user’s machine it is opened by navigating to the programs start option and that will depend on the users operating system. Figure

1.1 is an example of the initial opening screen on a computer. There are four options available on this initial screen:

A screenshot of a computer

Description automatically generated

Fig: 1.1 Weka GUI

**1. Explorer** - the graphical interface used to conduct experimentation on raw data After clicking the Explorer button the weka explorer interface appears.

A screenshot of a computer

Description automatically generated

Fig: 1.2 Pre-processor

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Inside the weka explorer window there are six tabs:

1. **Preprocess-** used to choose the data file to be used by the application.

**Open File**- allows for the user to select files residing on the local machine or recorded medium **Open URL**- provides a mechanism to locate a file or data source from a different location specified by the user

**Open Database**- allows the user to retrieve files or data from a database source provided by user

1. **Classify-** used to test and train different learning schemes on the preprocessed data file under experimentation

A screenshot of a computer

Description automatically generated

Fig: 1.3 choosing Zero set from classify

Again there are several options to be selected inside of the classify tab. Test option gives the user the choice of using four different test mode scenarios on the data set.

* 1. Use training set
  2. Supplied training set
  3. Cross validation
  4. Split percentage

1. **Cluster-** used to apply different tools that identify clusters within the data file.

The Cluster tab opens the process that is used to identify commonalties or clusters of occurrences within the data set and produce information for the user to analyze.

A screenshot of a computer

Description automatically generated

1. **Association-** used to apply different rules to the data file that identify association within the data. The associate tab opens a window to select the options for associations within thedataset.

A screenshot of a computer

Description automatically generated

1. **Select attributes-**used to apply different rules to reveal changes based on selected attributes inclusion or exclusion from the experiment
2. **Visualize-** used to see what the various manipulation produced on the data set in a 2D format, in scatter plot and bar graph output.
3. **Experimenter** - this option allows users to conduct different experimental variations on data sets and perform statistical manipulation. The Weka Experiment Environment enables the user to create, run, modify, and analyze experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyze the results to determine if one of the schemes is (statistically) better than the other schemes.

A screenshot of a computer

Description automatically generated

Fig: 1.6 Weka experiment

**Results destination**: ARFF file, CSV file, JDBC database.

**Experiment type**: Cross-validation (default), Train/Test Percentage Split (data randomized).

**Iteration control**: Number of repetitions, Data sets first/Algorithms first.

**Algorithms**: filters

1. **Knowledge Flow** -basically the same functionality as Explorer with drag and drop functionality. The advantage of this option is that it supports incremental learning from previous results
2. **Simple CLI** - provides users without a graphic interface option the ability to execute commands from a terminal window.

## b. Explore the default datasets in weka tool.

Click the “***Open file…***” button to open a data set and double click on the “***data***” directory. Weka provides a number of small common machine learning datasets that you can use to practice on. Select the “***iris.arff***” file to load the Iris dataset.

A screenshot of a computer

Description automatically generated

Fig: 1.7 Different Data Sets in weka

Exercise:

* 1. Normalize the data using min-max normalization

**LAB NO. 2 Date**

# Creating new ARFF file

## Creating a new ARFF file

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software in WEKA, each data entry is an instance of the java class weka.core. Instance, and each instance consists of a For loading datasets in WEKA, WEKA can load ARFF files. Attribute Relation File Format has two sections:

* + 1. The Header section defines relation (dataset) name, attribute name, and type.
    2. The Data section lists the data instances.

A screenshot of a computer

Description automatically generated

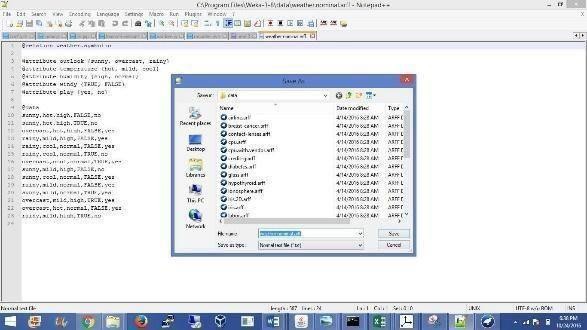
The figure above is from the textbook that shows an ARFF file for the weather data. Lines beginning with a % sign are comments. And there are three basic keywords:





The external representation of an Instances class Consists of:

* **A header:** Describes the attribute types
* **Data section:** Comma separated list of data



Exercise:

1. Creating a sample dataset for supermarket (supermarket.arff)

#### LAB NO: 3 Date:

#### Data Processing Techniques on Data Set

## Pre-process a given dataset based on Attribute selection

To search through all possible combinations of attributes in the data and find which subset of attributes works best for prediction, make sure that you set up attribute evaluator to „Cfs Subset Val‟ and a search method to „Best First‟. The evaluator will determine what method to use toassign a worth to each subset of attributes. The search method will determine what style of search to perform. The options that you can set for selection in the „Attribute Selection Mode‟ fig no: 3.2

1. **Use full training set.** The worth of the attribute subset is determined using the full set of training data.
2. **Cross-validation.** The worth of the attribute subset is determined by a process of cross- validation. The „Fold‟ and „Seed‟ fields set the number of folds to use and the random seed used when shuffling the data.

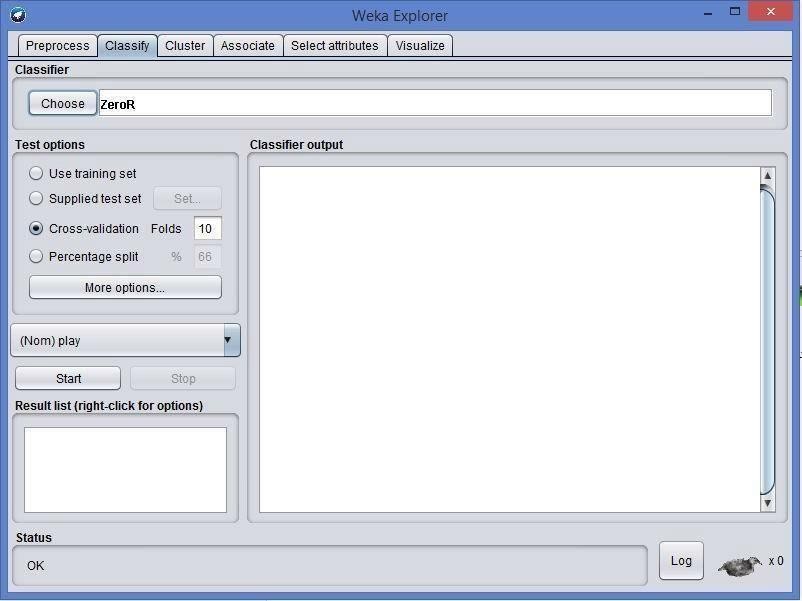
Specify which attribute to treat as the class in the drop-down box below the test options. Once all the test options are set, you can start the attribute selection process by clicking on „Start‟ button.

Fig: 3.1 Choosing Cross validation

When it is finished, the results of selection are shown on the right part of the window and entry is added to the „Result list‟.

# 2. Visualizing Results

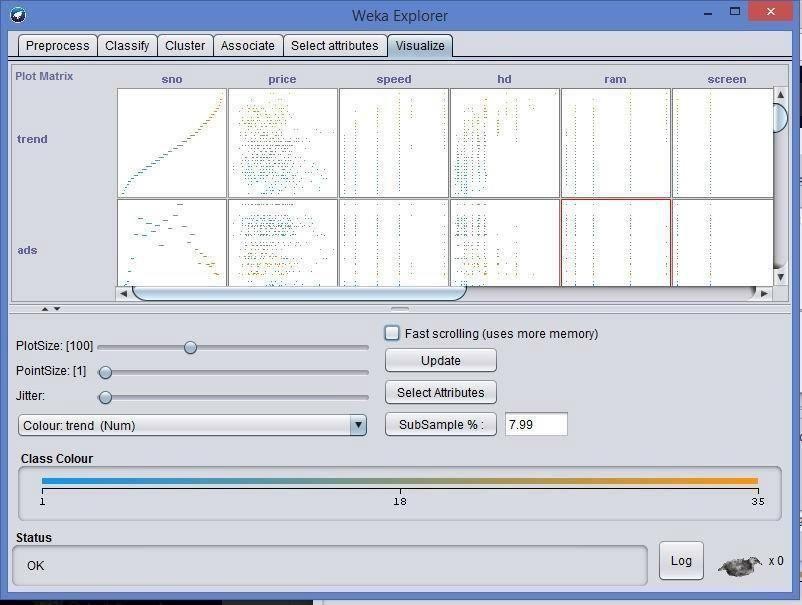


Fig: 3.2 Data Visualization

WEKA‟s visualization allows you to visualize a 2-D plot of the current working relation. Visualization is very useful in practice; it helps to determine difficulty of the learning problem. WEKA can visualize single attributes (1-d) and pairs of attributes (2-d), rotate 3-d visualizations (Xgobi-style). WEKA has “Jitter” option to deal with nominal attributes and to detect “hidden” data points.

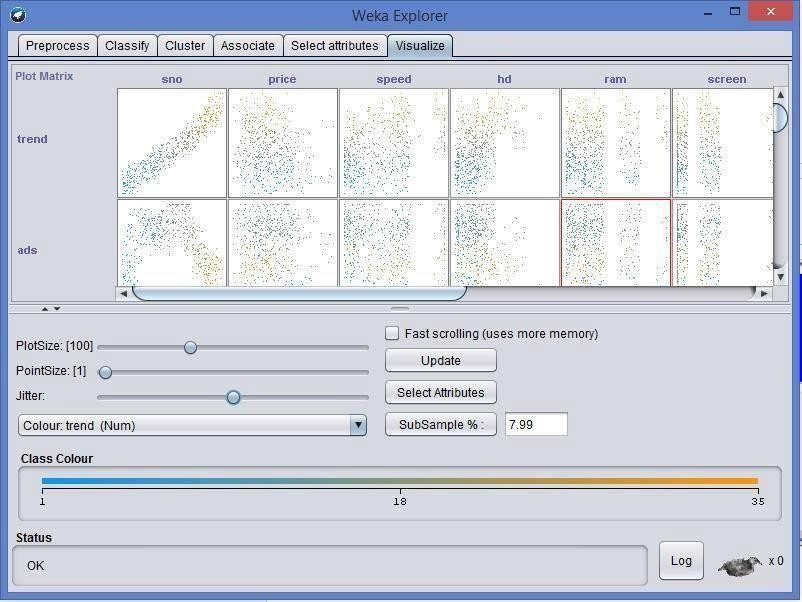
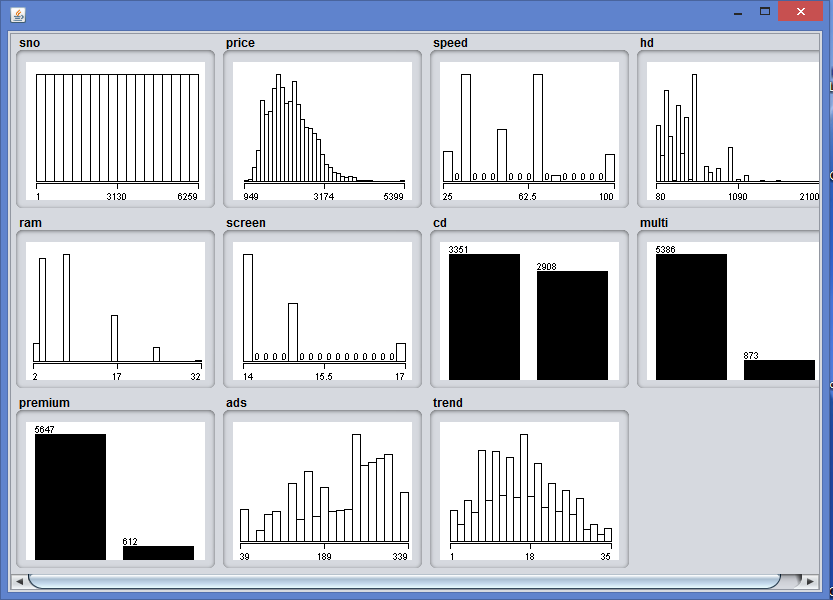


Fig 3.3: Preprocessing with jitter



Exercise

Fig: 3.3 Data visualization

1. Explain data preprocessing steps for heart disease dataset.

# Aim: B. Pre-process a given dataset based on Handling Missing Values

**Process**: Replacing Missing Attribute Values by the Attribute Mean. This method is used for data sets with numerical attributes. An example of such a data set is presented in fig no: 3.4

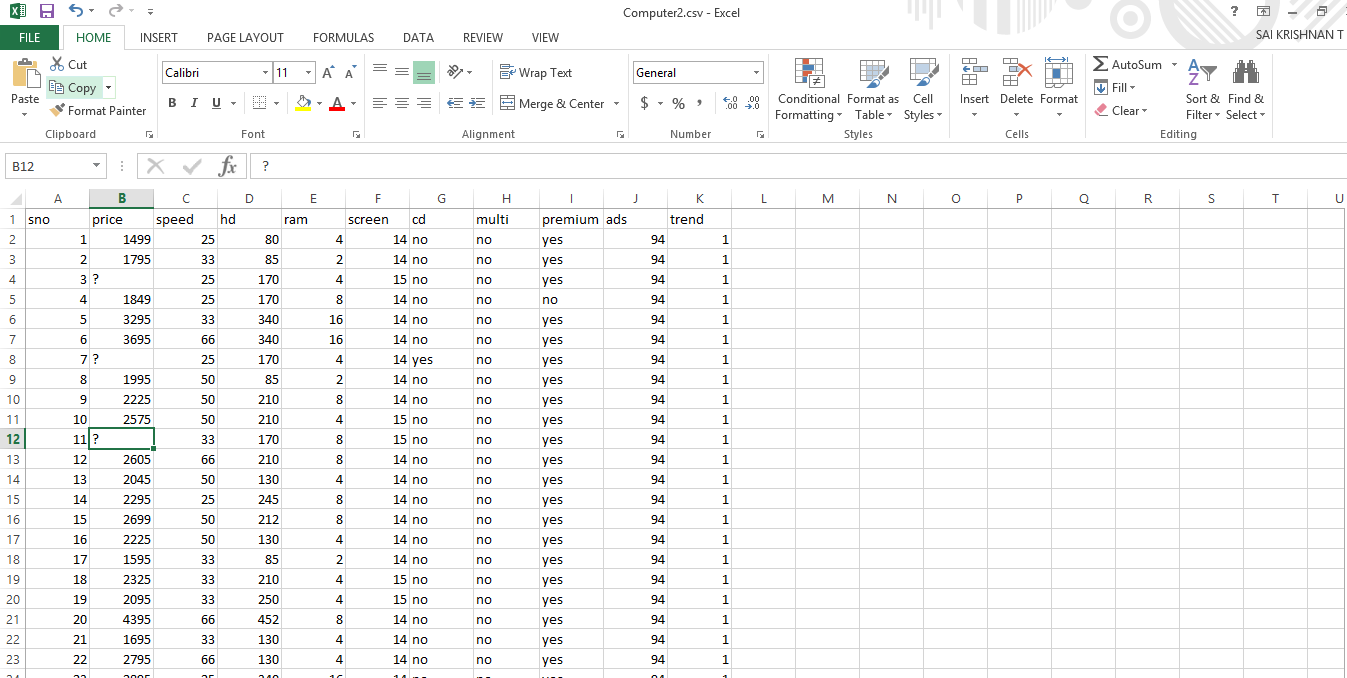


Fig: 3.4 Missing values

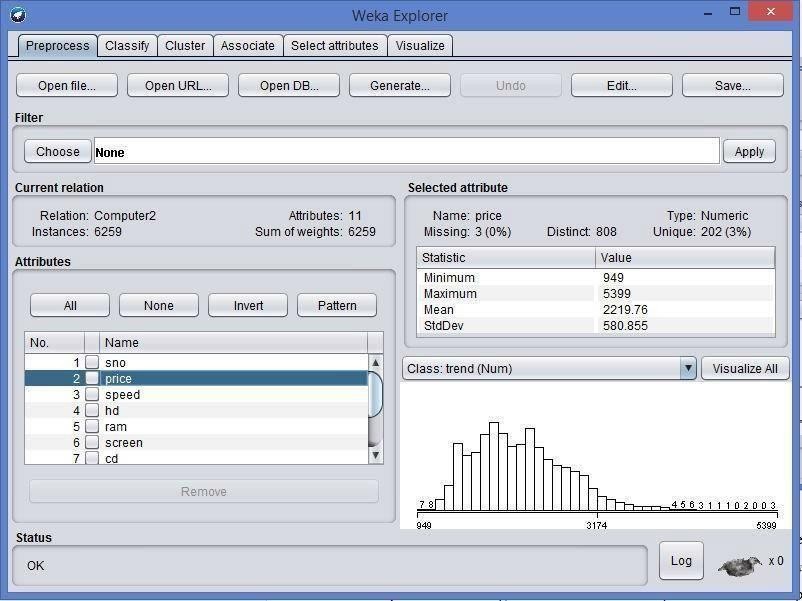
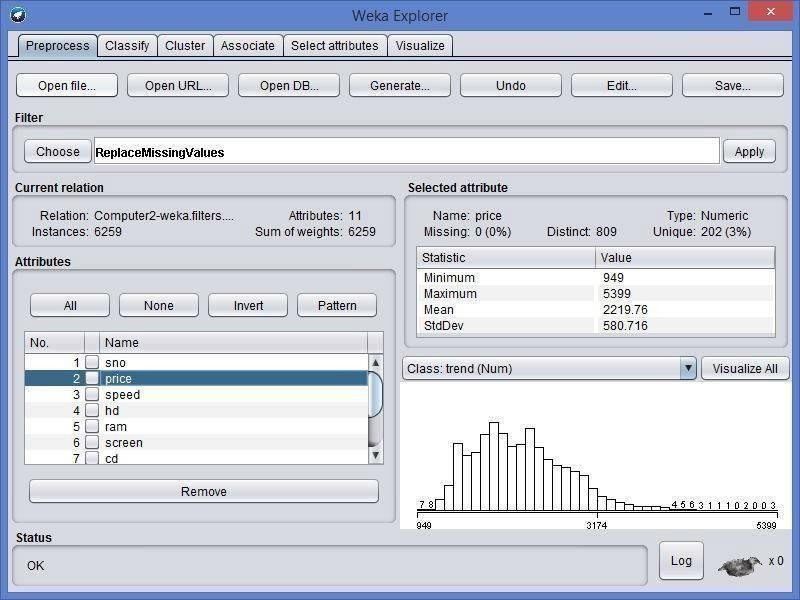
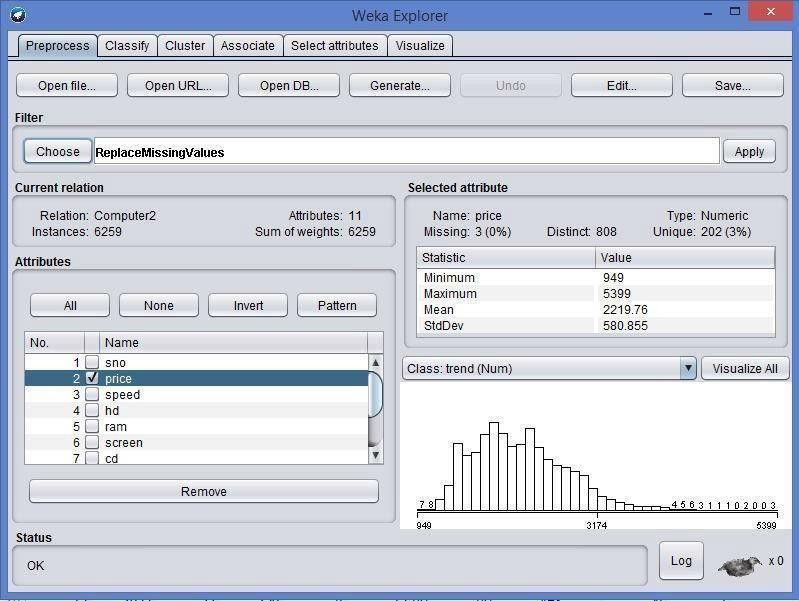
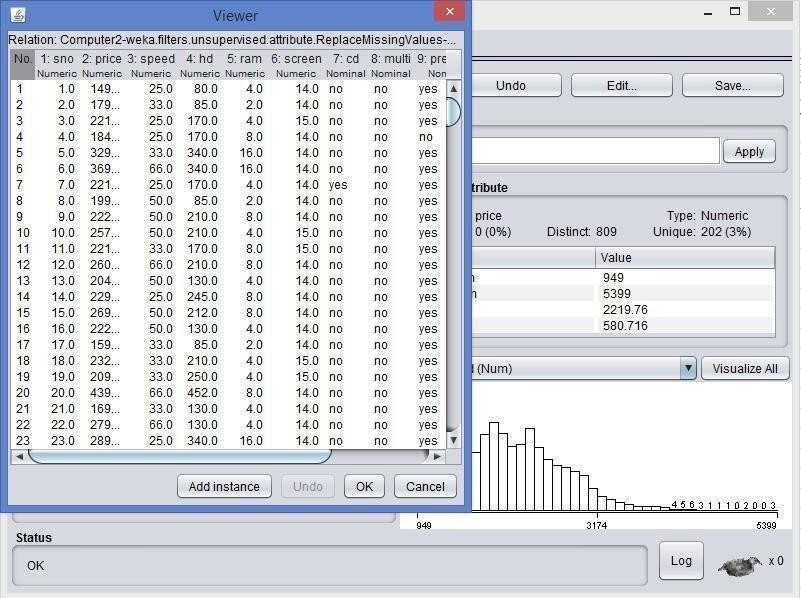


Fig: 3.5 Choosing a dataset

In this method, every missing attribute value for a numerical attribute is replaced by the arithmetic mean of known attribute values. In Fig, the mean of known attribute values for Temperature is 99.2, hence all missing attribute values for Temperature should be replaced by The table with missing attribute values replaced by the mean is presented in fig. For symbolic attributes Headache and Nausea, missing attribute values were replaced using the most common value of the Replace Missing Values.





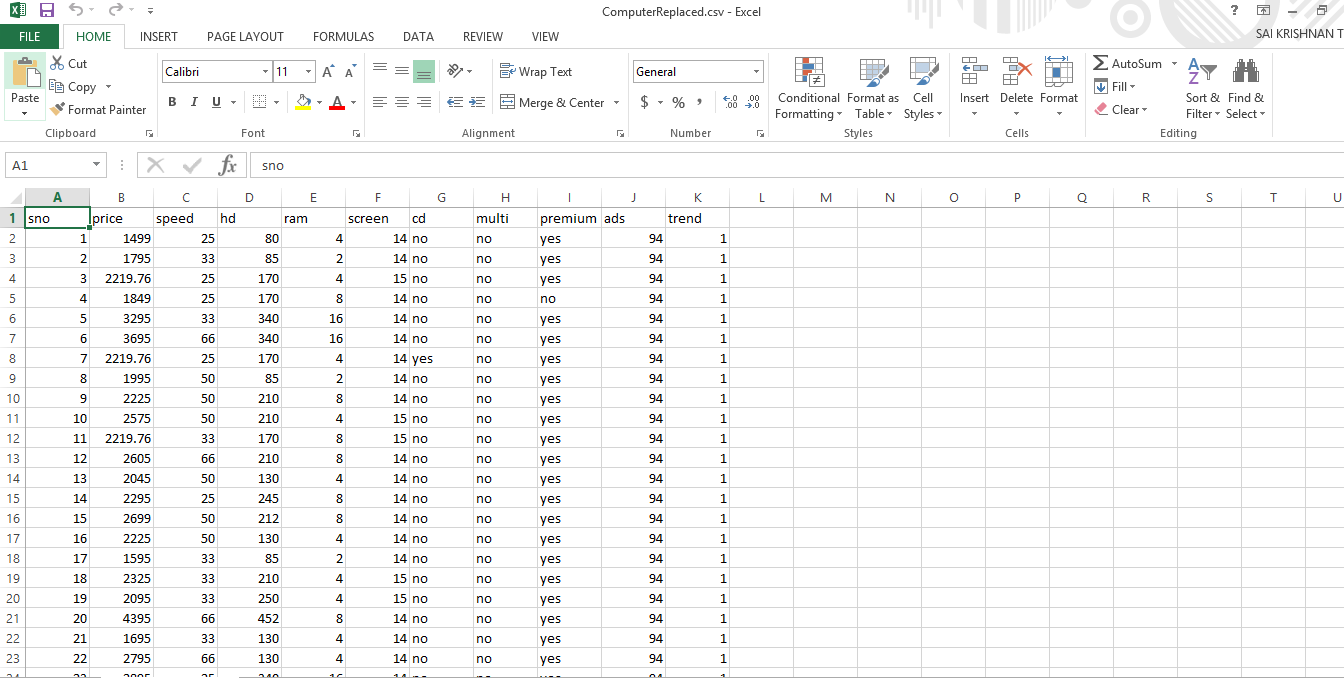


Fig: 3.6 Replaced values

Exercise

Create your own dataset having missing values included.

#### LAB NO: 4 Date:

**Data cube construction – OLAP operations**

An OLAP cube is a term that typically refers to multi-dimensional array of data. OLAP is an acronym for online analytical processing,[1]which is a computer-based technique of analyzing data to look for insights. The term cube here refers to a multi-dimensional dataset, which is also sometimes called a hypercube if the number of dimensions is greater than 3.

## Operations:

1. **Slice** is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension.[4] The picture shows a slicing operation: The sales figures of all sales regions and all product categories of the company in the year 2005 and 2006 are "sliced" out of the data cube.
2. **Dice:** The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions.[5]The picture shows a dicing operation: The new cube shows the sales figures of a limited number of product categories, the time and region dimensions cover the same range as before.
3. **Drill Down/Up** allows the user to navigate among levels of data ranging from the most summarized (up) to the most detailed (down).[4] The picture shows a drill-down operation: The analyst moves from the summary category "Outdoor-Schutzausrüstung" to see the sales figures for the individual products.
4. **Roll-up:** A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as "profit = sales

- expenses".

1. **Pivot** allows an analyst to rotate the cube in space to see its various faces. For example, cities could be arranged vertically and products horizontally while viewing data for a particular quarter. Pivoting could replace products with time periods to see data across time for a single product.

Exercise:

Apply the OLAP operations for the above banking application.

#### LAB NO. 5 Date:

**MINI PROJECT – SYNOPSIS SUBMISSION**

**Objective**

## 1. To finalize the title of miniproject

### Students are required to submit the synopsis of the mini project. They are encouraged to select the topic based on the Scopus/WoS indexed paper in data mining area. They can also develop any application based on the data mining algorithms.

**SAMPLE PROJECT SYNOPSIS TITLE**

### Project Description:

Objectives:

### Scope:

Software Requirements:

### Submitted by

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Registration number | Roll Number | Semester & Branch | Section |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Specifications to be followed:

### Font size: 12 for regular text, font size: 14 for side headings and font size: 16 for title

1. Font Type: Times New Roman

### Text alignment : Justified(Both Sides)

#### LAB NO. 6 Date:

**APRIORI ALGORITHM**

#### Objectives

1. To implement Apriori algorithm
2. To generate association rules based on frequent item sets

#### Apriori Algorithm

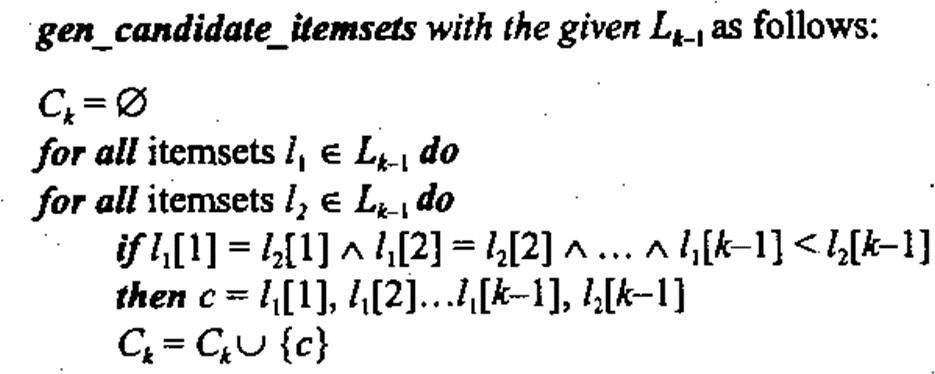
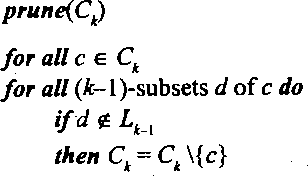
Agarwal and Srikant proposed the Apriori algorithm in 1994. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation)*, and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search to count candidate item sets efficiently. This algorithm uses downward closure property, which states that, “Any subset of a frequent itemset must be frequent”. It is called as apriori because it uses prior knowledge of frequent item set properties.

It uses level-wise search, where k-itemsets (an itemset containing k number of items is called as a k-itemset) are used to explore (k+1) itemsets to mine frequent itemsets from transactional database. First, the set of frequent 1-temset (L1) is found. L1 is used to find L2, which is used to find L3 and so on, until no more frequent k-itemsets can be found.

The candidate-gen function takes *Lk*-1 and returns a superset (called the candidates) of the set of all frequent

1. itemsets. It has two steps
   * *join* step: Generate all possible candidate itemsets *Ck* of length *k*
   * *prune* step: Remove those candidates in *Ck* that cannot be frequent.

Figure 7.1 shows the pseudocode of the algorithm.



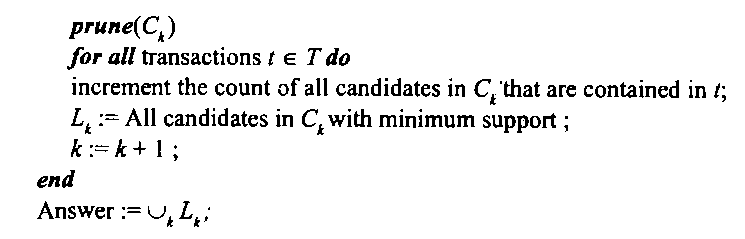
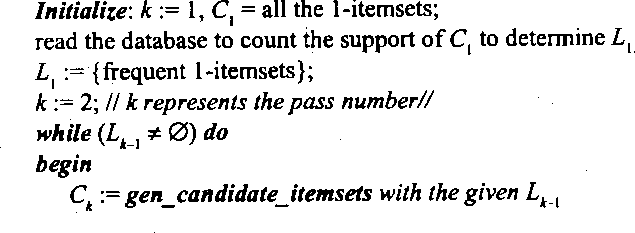


Figure 7.1 Apriori algorithm

#### Example

Consider a database consisting of 9 transactions as given in Table 7.1. Find the frequent itemsets using apriori algorithm. Assume the minimum support count as 2.

* Scan the database to get the support of each candidate item C1 = { {A} – 6, {B} – 7, {C} – 6, {D} – 2, {E} – 2 }
* Determine L1 from C1

L1 = { {A}, {B}, {C}, {D}, {E} }

Table 7.1 Database

|  |  |
| --- | --- |
| Transaction ID | List of Items |
| 1 | A, B, E |
| 2 | B, D |
| 3 | B, C |
| 4 | A, B, D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, C |

* Generate C2 from L1 using apriori join step

C2 = { {A,B}, {A,C}, {A,D}, {A,E}, {B,C}, {B,D}, {B,E}, {C,D}, {C,E}, {D,E}}

* Scan the database to get the support of each candidate item of C2

C2 = { {A,B} – 4, {A,C} – 4, {A,D} – 1, {A,E} – 2, {B,C} – 4, {B,D} – 2, {B,E} – 2, {C,D} – 0, {C,E}

– 1, {D,E} – 0 }

* Determine L2 from C2

L2 = { {A,B}, {A,C}, {A,E}, {B,C}, {B,D}, {B,E} }

* Generate C3 from L2 using apriori join step

C3 = { {A,B,C}, {A,B,E}, {A,C,E},{ B,C,D}, {B,C,E}, {B,D,E} }

* Prune C3 using apriori prune step C3 = { {A,B,C}, {A,B,E} }
* Scan the database to get the support of each candidate item of C3 C3 = { {A,B,C} – 2, {A,B,E} – 2 }
* Determine L3 from C3

L3 = { {A,B,C}, {A,B,E} }

* Generate C4 from L3 using apriori join step C4 = { {A,B,C, E} }
* Prune C4 using apriori prune step C4 = { }

L4 = { }

The algorithm stops, as L4 is empty.

Answer: Frequent itemsets: {{A}, {B}, {C}, {D}, {E}, {A, B}, {A, C}, {A, E}, {B, C}, {B,D}, {B,E},

{A,B,C}, {A, B, E}}

#### Association Rules

An association rule is an implication of the form X  Y, where X & Y are transactions with set of items from a transactional database ‘D’.

* + The rule X  Y holds in ‘D’ with confidence c if c% of transactions in D that contain X also contain Y
  + The rule X  Y has support s in D if s% of transactions in D contain X U Y

Find all rules that have support and confidence greater than user-specified minimum support and minimum confidence. The steps to generate association rules is as given below:

* + For each frequent itemset *l*, generate all nonempty subsets of *l*.
  + For every nonempty subset *s* of *l*, output the rule “*s* *l-s* ” if (*support count*(*l*) / *support count*(*s*))

≥ *min conf*, where *min conf* is the minimum confidence threshold.

* + As the rules are generated from the frequent itemsets, each one automatically satisfies minimum support.

Example: Consider one of the frequent itemset {A, B, C} for the database given in Table 7.1 to generate association rule by considering the minimum confidence threshold as 75%

* + *l* ={A, B, E} , Subsets of *l* ={{A, B},{A, E},{B, E},{A},{B},{E}}
  + s = {A, B}, {A, B}  {E} , conf({A, B} {E})=2/4=50%
  + s = {A, E}, {A, E}  {B} , conf({A, C}  {B})=2/2=100%
  + s = {B, E}, {B, E}  {A} , conf({B, E}  {A})=2/2=100%
  + s = {A}, {A} {B, E} , conf({A}  {B, E})=2/6=33%
  + s = {B}, {B}  {A,E} , conf({B}  {A,E})=2/7=29%
  + s = {E}, {E}  {A,B} , conf({E} {A,B})=2/2=100%

Association rules satisfying the minimum support and threshold are as follows:

{A, E}  {B}, {B, E}  {A}, {E}  {A, B}

#### Lab Exercise

1. Find the frequent itemsets by using the Apriori algorithm for a given transactional database and determine the association rules by considering suitable minimum support and confidence values

#### Additional Exercise

1. Implement partition based Apriori algorithm for a database stored as a file and find the association rules.

#### LAB NO: 7 Date:

**K-MEANS ALGORITHM**

#### Objective

* 1. To implement K-means algorithm for clustering

#### k-means algorithm

Clustering is a process of partitioning various objects into groups called as clusters, with the aim of having high intra-cluster similarity and low inter cluster similarity. It is an example of unsupervised learning. Clustering is a form of learning by observation, rather than learning by examples.

K-means is a well-known and commonly used partitioning method. The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured concerning the mean value of the objects in a cluster, which can be viewed as the cluster’s centroid or center of gravity.

The k-means algorithm proceeds as follows:

First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the square-error criterion is used, as given in equation 8.1.

𝑘

𝐸 = ∑

𝑖=1

∑𝑝∈𝐶𝑖 |𝑝 − 𝑚𝑖|2 (8.1)

where, E is the sum of the square error for all objects in the data set; p is the point in space representing a given object; and mi is the mean of cluster Ci (both p and mi are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This criterion tries to make the resulting k clusters as compact and as separate as possible. The k-means procedure is summarized in Figure 8.1

The algorithm attempts to determine k partitions that minimize the square-error function. It works well when the clusters are compact clouds that are well separated from one another. The method is relatively scalable and efficient in processing large data sets because the computational complexity of the algorithm is O(nkt), where n is the total number of objects, k is the number of clusters, and t is the number of iterations. Normally, k << n and t << n. The method often terminates at a local optimum.

The K-means method, however, can be applied only when the mean of a cluster is defined. The necessity for users to specify k, the number of clusters, in advance can be seen as a disadvantage. The K-means method is not suitable for discovering clusters with nonconvex shapes or clusters of very different size. It is sensitive to noise and outlier data points because a small number of such data can substantially influence the mean value.

Suppose that there is a set of objects located in space as depicted in the rectangle shown in Figure 8.2(a). Let k = 3; that is, the user would like the objects to be partitioned into three clusters. According to the algorithm in Figure 8.1, we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a “+”. Each object is distributed to a cluster based on the cluster center to which it is

the nearest. Such a distribution forms silhouettes encircled by dotted curves, as shown in Figure 8.2(a). Next, the cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster. Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest. Such a redistribution forms new silhouettes encircled by dashed curves, as shown in Figure 8.2(b).

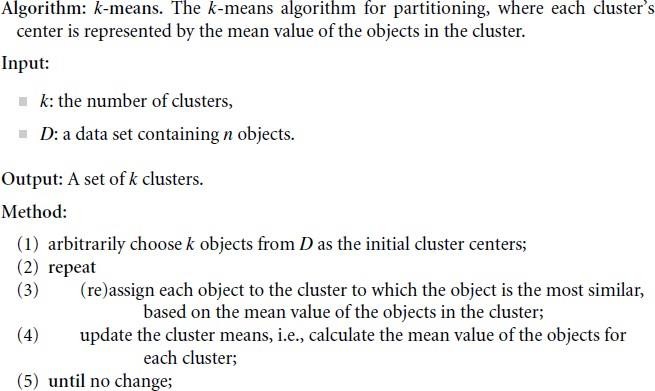


Figure 8.1 K-means algorithm

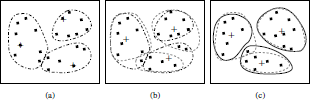
.This process iterates, leading to Figure 8.2(c). The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as iterative relocation. Eventually, no redistribution of the objects in any cluster occurs, and so the process terminates. The resulting clusters are returned by the clustering process.

Figure 8.2 Clustering of a set of objects based on the k-means method

#### Lab Exercise

**1.** Implement K-means algorithm for a given dataset using Euclidean distance as a similarity measure**.**

#### Additional Exercise

1. Use Manhattan as a similarity measure to cluster the given dataset using K-means algorithm

#### Lab 8 Date:

**DECISION TREE ID3 ALGORITHM FOR CLASSIFICATION**

#### Objectives:

1. To understand the working of decision tree for classification
2. Implement ID3 algorithm to construct the decision tree.

#### Introduction:

ID3 builds a decision tree from a fixed set of examples. The resulting tree is used to classify future samples. The example has several attributes and belongs to a class (like yes or no). The leaf nodes of the decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a possible value of the attribute. ID3 uses information gain to help it decide which attribute goes into a decision node. The advantage of learning a decision tree is that a program, rather than a knowledge engineer, elicits knowledge from an expert.

How does ID3 decide which attribute is the best? A statistical property, called **information gain**, is used. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. In order to define gain, we first borrow an idea from information theory called entropy. Entropy measures the amount of information in an attribute.

Given a collection S of c outcomes, Entropy is calculated as given in equation 9.1

Entropy(S) = S -p(I) log2 p(I) (9.1)

where, p(I) is the proportion of S belonging to class I. S is over c. Log2 is log base2. Note that S is not an attribute but the entire sample set. Gain(S, A) is information gain of example set S on attribute A is defined as given in equation 9.2.

Gain(S, A) = Entropy(S) - S ((|Sv| / |S|) \* Entropy(Sv)) (9.2)

where,

S is each value v of all possible values of attribute A Sv = subset of S for which attribute A has value v

|Sv| = number of elements in Sv

|S| = number of elements in S

ID3 Algorithm:

ID3 (Examples, Target\_Attribute, Attributes) Create a root node for the tree

If all examples are positive, Return the single-node tree Root, with label = +. If all examples are negative, Return the single-node tree Root, with label = -.

If number of predicting attributes is empty, then Return the single node tree Root, with label = most common value of the target attribute in the examples.

Otherwise Begin

A ← The Attribute that best classifies examples. Decision Tree attribute for Root = A.

End

For each possible value, *vi*, of A,

Add a new tree branch below Root, corresponding to the test A = *vi*. Let Examples(*vi*) be the subset of examples that have the value *vi* for A If Examples(*vi*) is empty

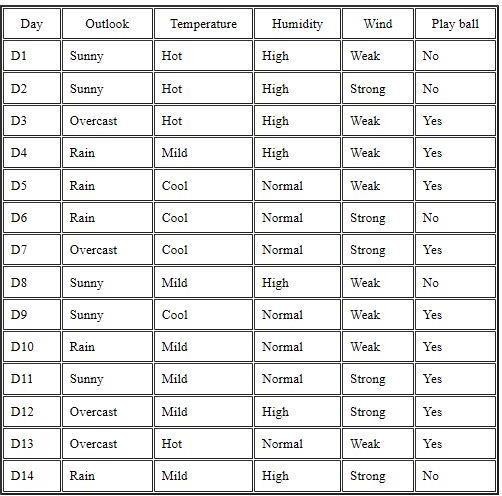
Then below this new branch add a leaf node with label = most common target value in the examples

Else below this new branch add the subtree ID3 (Examples(*vi*), Target\_Attribute, Attributes – {A})

Return Root

Consider the following example:

Table 9.1: Play ball dataset



Suppose Table 9.1 is a set of 14 examples in which one of the attributes is wind speed. The values of Wind can be *Weak* or *Strong*. The classification of these 14 examples are 9 YES and 5 NO. For attribute Wind, suppose there are 8 occurrences of Wind = Weak and 6 occurrences of Wind = Strong. For Wind = Weak, 6 of the examples are YES and 2 are NO. For Wind = Strong, 3 are YES and 3 are NO. Therefore, Gain(S,Wind) = Entropy(S)-(8/14)\*Entropy(Sweak)-(6/14)\*Entropy(Sstrong)

= 0.940 - (8/14)\*0.811 - (6/14)\*1.00

= 0.048

Entropy(Sweak) = - (6/8)\*log2(6/8) - (2/8)\*log2(2/8) = 0.811 Entropy(Sstrong) = - (3/6)\*log2(3/6) - (3/6)\*log2(3/6) = 1.00

For each attribute, the gain is calculated and the highest gain is used in the decision node. This process goes on until all data is classified perfectly or we run out of attributes.

#### Lab Exercises

1. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use data set given in table 1 for building the decision tree.
2. Classify new samples using the rules obtained from the decision tree in exercise1.

#### Additional Exercises

1. Construct the decision tree for the following dataset of mushroom

<https://archive.ics.uci.edu/ml/machine-learning-databases/>mushroom/agaricus-lepiota.data

**Lab 9 Date:**

# NAIVE BAYES CLASSIFIER

#### Objectives:

* 1. To understand the Naïve Bayesian classifier.
  2. To build Naïve Bayes Gaussian classifier to classify data.
  3. To build Multinomial Naïve Bayes Gaussian classifier to classify text.

#### Naïve Bayesian Gaussian Classifier Introduction:

Naive Bayes is a supervised learning algorithm used for classification tasks. Hence, it is called as Naive Bayes Classifier. As other supervised learning algorithms, Naive Bayes uses features to make a prediction on a target variable. The key difference is that Naive Bayes assumes that features are independent of each other and there is no correlation between features

Bayes comes from the famous [Bayes’ Theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) of Thomas Bayes as shown in equation 10.1. To get a comprehensive understanding of Bayes’ Theorem, we should talk about probability and conditional probability first.

P(A|B)=

𝑃(𝐴).𝑃(𝐵)

𝑃(𝐵)

(10.1)

Naive Bayes classifier calculates the probability of a class given a set of feature values (i.e. p(yi | x1, x2 ,

… , xn)). Input this into Bayes’ theorem as given in equation 10.2.

P(yi|x1,x2,….xn)=P(x1, x2, … . xn|yi).P(yi)

P(X1,X2,….Xn)

(10.2)

**p(x1, x2 , … , xn | yi)** means the probability of a specific combination of features given a class label. To be able to calculate this, we need extremely large datasets to have an estimate on the probability distribution for all different combinations of feature values. To overcome this issue, **naive bayes algorithm assumes that all features are independent of each other.** Furthermore, denominator (p(x1,x2, … , xn)) can be removed to simplify the equation because it only normalizes the value of conditional probability of a class given an observation ( p(yi | x1,x2,… … , xn)).The probability of a class ( p(yi) ) is very simple to calculate as shown in equation 10.3.

P(yi)=𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑜𝑏𝑠𝑒𝑟𝑣𝑎𝑡𝑖𝑜𝑛𝑠 𝑤𝑖𝑡ℎ 𝑐𝑙𝑎𝑠𝑠 𝑦𝑖

𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑎𝑙𝑙 𝑜𝑏𝑠𝑒𝑟𝑣𝑎𝑡𝑖𝑜𝑛𝑠

(10.3)

Under the assumption of features being independent, **p(x1, x2 ,.. , xn | yi)** can be written as given in equation 10.4.

x1,x2,….xn|yi)=P(x1|yi).P(x2|yi)………P(xn|yi) (10.4)

The conditional probability for a single feature given the class label (i.e. p(x1 | yi) ) can be more easily estimated from the data. The algorithm needs to store probability distributions of features for each class independently. The type of distributions depends on the characteristics of features:

1. For binary features (Y/N, True/False, 0/1): Bernoulli distribution
2. For discrete features (i.e. word counts): Multinomial distribution
3. For continuous features: Gaussian (Normal) distribution

#### Multinomial Naive Bayes Classifier to Classify Text Introduction:

Multinomial naive Bayes works similar to Gaussian naive Bayes, however the features are assumed to be multinomial distributed. In practice, this means that this classifier is commonly used when we have discrete data (e.g. movie ratings ranging 1 and 5).

For sentiment analysis, a Naive Bayes classifier is one of the easiest and most effective ways to hit the ground running for sentiment analysis.

Deriving the prior probability of a class is rather trivial, as it is simply the sum of all words in *doc* that are assigned to *c* divided by the number of words in *doc* as shown in equation 10.5.

https://miro.medium.com/max/114/1*-PkUQ4n42T1YLaK-jXI9VQ.png(10.5)

How do we learn all of the probabilities of that make up each feature? The solution is again rather simple: for a given word *w* in words *W* from *d* we count how many of *w* belong in class *c*. We then divide this by all the words in *d* that belong to *c*. This gives us a probability for a word *w* given *c* as in equation 10.6.

(10.6)

#### Computing Error Rate, Accuracy, Precision and Recall from confusion matrix:

A confusion matrix is a technique for summarizing the performance of a classification algorithm. The number of correct and incorrect predictions are summarized with count values and broken down by each class. These numbers are then organized into a table, or a matrix as shown in Table 10.1.

Table 10.1: Confusion matrix for 2-class problem

|  |  |  |  |
| --- | --- | --- | --- |
| ACTUAL |  | PREDICTED | |
| **Positive** | **negative** |
| **Positive** | True Positive | False negative |
| **negative** | False Positive | True Negative |

From confusion matrix one can get the following measures.

* 1. Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset as in equation 10.7.

Accuracy=(TP+TN)/(TP+TN+FP+FN) (10.7)

* 1. Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset as in equation 10.8.

Error rate= (FP+FN)/(TP+TN+FP+FN) (10.8)

* 1. Precision evaluates the fraction of correct classified instances among the ones classified as positive as in equation 10.9.

Precision=TP/(TP+FP) (10.9)

* 1. Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made as in equation 10.10.

Recall=TP/(TP+FN) (10.10)

#### sklearn to fetch measures

sklearn can be used to fetch these measures using the following codes:

*from sklearn import metrics metrics.confusion\_matrix metrics.classification\_report(predicted, expected)*

#### Lab Exercises

1. Write a program to implement the naïve Bayesian classifier(Gaussian) for *Pima indians diabetes*

training data set. Compute the accuracy of the classifier, considering few test data sets.

1. Use the naïve Bayesian(Multinomial) Classifier model to perform text classification task on 20newsgroups dataset.
2. Calculate the accuracy, precision, and recall for your data set on the confusion matrix obtained.

#### Additional Exercises

* 1. Write a program to implement the naïve Bayesian classifier(Gaussian) for scikit-learn wine training data set. Compute the accuracy of the classifier, considering few test data sets.
  2. Use the naïve Bayesian(Multinomial) Classifier model to perform text classification task on [Reuters News Dataset.](https://archive.ics.uci.edu/ml/datasets/Reuters-21578%2BText%2BCategorization%2BCollection)

#### LAB NO. 10 Date:

**MINI PROJECT – IMPLEMENTATION**

**Objective**

* + 1. **To implement the miniproject**

### Students are required to implement the mini project.

#### LAB NO. 11 Date:

**MINI PROJECT – PROGRESS**

**Objective**

## 1. To discuss the progress of miniproject implementation

### Students are required to show the progress of mini project implementation and discuss with faculty

#### LAB NO. 12 Date:

**MINI PROJECT – PROGRESS**

**Objective**

## 1. To discuss the progress of miniproject result and analysis

### Students are required to show the progress of mini project result and analysis discuss with faculty

#### REFERENCES

1. Jiawei Han and Micheline Kamber, “Data Mining- Concepts and Techniques”, 3rd Edition, Morgan Kaufmann Publishers, 2011.
2. Arun K. Pujari, ”Data Mining Techniques”, University press, 2006.
3. G.K. Gupta,”Introduction to data mining with Case Studies”, Easter Economy edition, Prentice Hall of India,2006.
4. Pang-Ning Tan,Michael Steinbach and Vipin Kumar,”Introduction to Data Mining” Pearson Education,2007.
5. Infosphere Documentation.
6. Rapid miner Documentation.