VEMU INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

15A05610- DATA WAREHOUSING & MINING LABORATORY

Regulation – R15

Year / Semester: III / II

INDEX

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech III-II Sem. (CSE)

L	T	P	C
0	0	4	2

15A05610 DATA WAREHOUSING & MINING LABORATORY

Course Objectives:

Learn how to build a data warehouse and query it (using open source tools like Pentaho Data Integration and Pentaho Business Analytics), Learn to perform data mining tasks using a data mining toolkit (such as open source WEKA), Understand the data sets and data preprocessing, Demonstrate the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression, Exercise the data mining techniques with varied input values for different parameters.

Course Outcomes:

- Ability to build Data Warehouse and Explore WEKA
- Ability to perform data preprocessing tasks and Demonstrate performing association rule mining on data sets
- Ability to perform classification, clustering and regression on data sets
- Ability to design data mining algorithms

Data Warehousing

Experiments:

Build Data Warehouse and Explore WEKA

A. Build a Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration tool, Pentoaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects, etc.).

(i). Identify source tables and populate sample data

(ii). Design multi-dimensional data models namely Star, snowflake and Fact constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, Manufacturing, Automobile, etc.).

(iii). Write ETL scripts and implement using data warehouse tools

(iv). Perform various OLAP operations such slice, dice, roll up, drill up and pivot

(v). Explore visualization features of the tool for analysis like identifying trends etc.

B. Explore WEKA Data Mining/Machine Learning Toolkit

(i). Downloading and/or installation of WEKA data mining toolkit,

(ii). Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface.

(iii). Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, Classify panel, Cluster panel, Associate panel and Visualize panel)

(iv). Study the arff file format

(v). Explore the available data sets in WEKA.

(vi). Load a data set (ex. Weather dataset, Iris dataset, etc.)
(vii). Load each dataset and observe the following:
   i. List the attribute names and they types
   ii. Number of records in each dataset
   iii. Identify the class attribute (if any)
   iv. Plot Histogram
   v. Determine the number of records for each class.
   vi. Visualize the data in various dimensions

Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets

A. Explore various options available in Weka for preprocessing data and apply (like Discretization Filters, Resample filter, etc.) on each dataset
B. Load each dataset into Weka and run Aprori algorithm with different support and confidence values. Study the rules generated.
C. Apply different discretization filters on numerical attributes and run the Apriori association rule algorithm. Study the rules generated. Derive interesting insights and observe the effect of discretization in the rule generation process.

Demonstrate performing classification on data sets

A. Load each dataset into Weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.
B. Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix and derive Accuracy, F-measure, TPrate, FPrate, Precision and Recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results.
C. Load each dataset into Weka and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.
D. Plot RoC Curves
E. Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

Demonstrate performing clustering on data sets

A. Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters)**.** Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.
B. Explore other clustering techniques available in Weka.
C. Explore visualization features of Weka to visualize the clusters. Derive interesting insights and explain.

Demonstrate performing Regression on data sets

A. Load each dataset into Weka and build Linear Regression model. Study the clusters formed. Use Training set option. Interpret the regression model and derive patterns and conclusions from the regression results.
B. Use options cross-validation and percentage split and repeat running the Linear Regression Model. Observe the results and derive meaningful results.
C. Explore Simple linear regression technique that only looks at one variable

Resource Sites:

1. **http://www.pentaho.com/**
2. **http://www.cs.waikato.ac.nz/ml/weka/**

Data Mining

Task 1: Credit Risk Assessment

Description:
The business of banks is making loans. Assessing the credit worthiness of an applicant is of crucial importance. You have to develop a system to help a loan officer decide whether the credit of a customer is good, or bad. A bank's business rules regarding loans must consider two opposing factors. On the one hand, a bank wants to make as many loans as possible. Interest on these loans is the banks profit source. On the other hand, a bank cannot afford to make too many bad loans. Too many bad loans could lead to the collapse of the bank. The bank's loan policy must involve a compromise: not too strict, and not too lenient.
To do the assignment, you first and foremost need is some knowledge about the world of credit. You can acquire such knowledge in a number of ways.

1. Knowledge Engineering. Find a loan officer who is willing to talk. Interview her and try to represent her knowledge in the form of production rules.
2. Books. Find some training manuals for loan officers or perhaps a suitable textbook on finance. Translate this knowledge from text form to production rule form.
3. Common sense. Imagine yourself as a loan officer and make up reasonable rules which can be used to judge the credit worthiness of a loan applicant.
4. Case histories. Find records of actual cases where competent loan officers correctly judged when, and when not to, approve a loan application.

The German Credit Data:

Actual historical credit data is not always easy to come by because of confidentiality rules. Here is one such dataset, consisting of 1000 actual cases collected in Germany. credit dataset (original) Excel sp readsheet version of the German credit data.
In spite of the fact that the data is German, you should probably make use of it for this assignment. (Unless you really can consult a real loan officer !)
A few notes on the German dataset

- DM stands for Deutsche Mark, the unit of currency, worth about 90 cents Canadian (but looks and acts like a quarter).

- Owns_telephone. German phone rates are much higher. So fewer people own telephones.
- Foreign_worker. There are millions of these in Germany (many from Turrkey). It is very hard to get German citizenship if you were not born of German parents.
- There are 20 attributes used in judging a loan applicant. The goal is to classify the applicant into one of two categories, good or bad.

Subtasks: (Turn in your answers to the following tasks)

1. List all the categorical (or nominal) attributes and the real-valued attributes separately.

2. What attributes do you think might be crucial in making the credit assessment ? Come up with some simple rules in plain English using your selected attributes.

3. One type of model that you can create is a Decision Tree - train a Decision Tree using the complete dataset as the training data. Report the model obtained after training.

4. Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly? (This is also called testing on the training set) Why do you think you cannot get 100 % training accuracy?

5. Is testing on the training set as you did above a good idea? Why or Why not ?

6. One approach for solving the problem encountered in the previous question is using cross-validation? Describe what is cross-validation briefly. Train a Decision Tree again using cross-validation and report your results. Does your accuracy increase/decrease? Why?

7. Check to see if the data shows a bias against "foreign workers" (attribute 20), or "personal-status" (attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the preprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect?

8. Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). Try out some combinations. (You had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)

9. Sometimes, the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifcations equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and cross-validation results. Are they significantly different from results obtained in problem 6 (using equal cost)?

10. Do you think it is a good idea to prefer simple decision trees instead of having long complex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?

11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning. Try reduced error pruning for training your Decision Trees using cross-validation (you can do this in Weka) and report the Decision Tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increase?

12.(Extra Credit): How can you convert a Decision Trees into "if-then-else rules". Make up your own small Decision Tree consisting of 2-3 levels and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules - one such classifier in Weka is rules. PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this dataset ? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a one R classifier. Rank the performance of j48, PART and oneR.

Task Resources:

- Andrew Moore's Data Mining Tutorials (See tutorials on Decision Trees and Cross Validation)
- Decision Trees (Source: Tan, MSU)
- Tom Mitchell's book slides (See slides on Concept Learning and Decision Trees)
- Weka resources:
  - Introduction to Weka (html version) (download ppt version)
  - Download Weka
  - Weka Tutorial
  - ARFF format
  - Using Weka from command line

Task 2: Hospital Management System
Data Warehouse consists Dimension Table and Fact Table.
REMEMBER The following
Dimension
The dimension object (Dimension):
_ Name
_ Attributes (Levels) , with one primary key
_ Hierarchies
One time dimension is must.
About Levels and Hierarchies

Dimension objects (dimension) consist of a set of levels and a set of hierarchies defined over those levels. The levels represent levels of aggregation. Hierarchies describe parent-child relationships among a set of levels.
For example, a typical calendar dimension could contain five levels. Two hierarchies can be defined on these levels:
H1: YearL > QuarterL > MonthL > WeekL > DayL
H2: YearL > WeekL > DayL

The hierarchies are described from parent to child, so that Year is the parent of Quarter, Quarter the parent of Month, and so forth.

About Unique Key Constraints

When you create a definition for a hierarchy, Warehouse Builder creates an identifier key for each level of the hierarchy and a unique key constraint on the lowest level (Base Level)

Design a Hospital Management system data warehouse (TARGET) consistig of Dimensions Patient, Medicine, Supplier, Time. Where measures are ' NO UNITS', UNIT PRICE.

Assume the Relational database (SOURCE) table schemas as follows

TIME (day, month, year),

PATIENT (patient_name, Age, Address, etc.,)

MEDICINE (Medicine_Brand_name, Drug_name, Supplier, no_units, Uinit_Price, etc.,)

SUPPLIER :(Supplier_name, Medicine_Brand_name, Address, etc.,)

If each Dimension has 6 levels, decide the levels and hierarchies, Assume the level names suitably.

Design the Hospital Management system data warehouse using all schemas. Give the example 4-D cube with assumption names.

(15A05610) DATA WAREHOUSING & MINING LABORATORY

Data Warehousing

Experiments:

1. Build Data Warehouse and Explore WEKA

A. Build a Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects, etc.).

(i). Identify source tables and populate sample data

(ii). Design multi-dimensional data models namely Star, snowflake and Fact constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, Manufacturing, Automobile, etc.).

(iii). Write ETL scripts and implement using data warehouse tools

(iv). Perform various OLAP operations such slice, dice, roll up, drill down and pivot

(v). Explore visualization features of the tool for analysis like identifying trends etc.

B. Explore WEKA Data Mining/Machine Learning Toolkit

(i). Downloading and/or installation of WEKA data mining toolkit,

(ii). Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface.

(iii). Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, classify panel, Cluster panel, Associate panel and Visualize panel)

(iv). Study the arff file format

(v). Explore the available data sets in WEKA.

(vi). Load a data set (ex. Weather dataset, Iris dataset, etc.)

(vii). Load each dataset and observe the following:

i. List the attribute names and they types

ii. Number of records in each dataset

iii. Identify the class attribute (if any)

iv. Plot Histogram

v. Determine the number of records for each class.

vi. Visualize the data in various dimensions

2. Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets

A. Explore various options available in Weka for preprocessing data and apply (like Discretization Filters, Resample filter, etc.) on each dataset

B. Load each dataset into Weka and run Aprori algorithm with different support and confidence values. Study the rules generated.

C. Apply different discretization filters on numerical attributes and run the Apriori association rule algorithm. Study the rules generated. Derive interesting insights and observe the effect of discretization in the rule generation process.

3. Demonstrate performing classification on data sets

A. Load each dataset into Weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.

B. Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix and derive Accuracy, F-measure, TPrate, FPrate, Precision and Recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results.

C. Load each dataset into Weka and perform Naïve-bayes classification and k- Nearest Neighbour classification. Interpret the results obtained.
D. Plot RoC Curves
E. Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

4. Demonstrate performing clustering on data sets

A. Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters)**.** Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.
B. Explore other clustering techniques available in Weka.
C. Explore visualization features of Weka to visualize the clusters. Derive interesting insights and explain.

5. Demonstrate performing Regression on data sets

A. Load each dataset into Weka and build Linear Regression model. Study the clusters formed. Use Training set option. Interpret the regression model and derive patterns and conclusions from the regression results.
B. Use options cross-validation and percentage split and repeat running the Linear Regression Model. Observe the results and derive meaningful results.
C. Explore Simple linear regression technique that only looks at one variable

Resource Sites:

1. http://www.pentaho.com/
2. http://www.cs.waikato.ac.nz/ml/weka/

Data Mining

Task 1: Credit Risk Assessment

Description:

The business of banks is making loans. Assessing the credit worthiness of an applicant is of crucial importance. You have to develop a system to help a loan officer decide whether the credit of a customer is good, or bad. A bank's business rules regarding loans must consider two opposing factors. On the one hand, a bank wants to make as many loans as possible. Interest on these loans is the banks profit source. On the other hand, a bank cannot afford to make too many bad loans. Too many bad loans could lead to the collapse of the bank. The bank's loan policy must involve a compromise: not too strict, and not too lenient.
To do the assignment, you first and foremost need is some knowledge about the world of credit. You can acquire such knowledge in a number of ways

1. Knowledge Engineering. Find a loan officer who is willing to talk. Interview her and try to represent her knowledge in the form of production rules.
2. Books. Find some training manuals for loan officers or perhaps a suitable textbookon finance. Translate this knowledge from text form to production rule form.
3. Common sense. Imagine yourself as a loan officer and make up reasonable rules which can be used to judge the credit worthiness of a loan applicant.
4. Case histories. Find records of actual cases where competent loan officerscorrectly judged when, and when not to, approve a loan application.

The German Credit Data:

Actual historical credit data is not always easy to come by because of confidentialityrules. Here is one such dataset, consisting of 1000 actual cases collected in Germany. credit dataset

(original) Excel spreadsheet version of the German credit data. In spite of the fact that the data is German, you should probably make use of it for this assignment. (Unless you really can consult a real loan officer!)

A few notes on the German dataset

•DM stands for Deutsche Mark, the unit of currency, worth about 90 cents Canadian (but looks and acts like a quarter).

•Owns_telephone. German phone rates are much higher. So fewer people own telephones.

•Foreign_worker. There are millions of these in Germany (many from Turrkey). It is very hard to get German citizenship if you were not born of German parents.

•There are 20 attributes used in judging a loan applicant. The goal is to classify the applicant into one of two categories, good or bad.

Subtasks: (Turn in your answers to the following tasks)

1. List all the categorical (or nominal) attributes and the real-valued attributesseparately.

2. What attributes do you think might be crucial in making the credit assessment? Come up with some simple rules in plain English using your selected attributes.

3. One type of model that you can create is a Decision Tree - train a Decision Tree using the complete dataset as the training data. Report the model obtained after training.

4. Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly? (This is also called testing on the training set) Why do you think you cannot get 100 % training accuracy?

5. Is testing on the training set as you did above a good idea? Why or Why not?

6. One approach for solving the problem encountered in the previous question is using cross-validation? Describe what is cross-validation briefly. Train a Decision Tree again using cross-validation and report your results. Does your accuracy increase/decrease? Why?

7. Check to see if the data shows a bias against "foreign workers" (attribute 20), or"personal-status" (attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute, you can use the preprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect?

8. Another question might be, do you really need to input so many attributes to getgood results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). Try out some combinations. (You had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)

9. Sometimes, the cost of rejecting an applicant who actually has a good credit (case1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and crossvalidation results. Are they significantly different from results obtained in problem 6 (using equal cost)?

10. Do you think it is a good idea to prefer simple decision trees instead of having longcomplex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?

11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning. Try reduced error pruning for training your Decision Trees using cross-

validation (you can do this in Weka) and report the Decision Tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increase?

12.(Extra Credit): How can you convert a Decision Trees into "if-then-else rules". Make up your own small Decision Tree consisting of 2-3 levels and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules - one such classifier in Weka is rules. PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this dataset? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a one R classifier. Rank the performance of j48, PART and oneR.

Task 2: Hospital Management System

Data Warehouse consists Dimension Table and Fact Table.

REMEMBER The following

Dimension

The dimension object (Dimension):

_ Name

_ Attributes (Levels) , with one primary key

_ Hierarchies

One time dimension is must.

About Levels and Hierarchies

Dimension objects (dimension) consist of a set of levels and a set of hierarchies defined over those levels. The levels represent levels of aggregation. Hierarchies describe parent-child relationships among a set of levels.

For example, a typical calendar dimension could contain five levels. Two hierarchies can be defined on these levels:

H1: YearL > QuarterL > MonthL > WeekL > DayL

H2: YearL > WeekL > DayL

The hierarchies are described from parent to child, so that Year is the parent of Quarter, Quarter the parent of Month, and so forth.

About Unique Key Constraints

When you create a definition for a hierarchy, Warehouse Builder creates an identifier key for each level of the hierarchy and a unique key constraint on the lowest level (Base Level)

Design a Hospital Management system data warehouse (TARGET) consisting of Dimensions Patient, Medicine, Supplier, Time. Where measures are _ NO UNITS ‘,

UNIT PRICE.

Assume the Relational database (SOURCE) table schemas as follows

TIME (day, month, year),

PATIENT (patient_name, Age, Address, etc.,)

MEDICINE (Medicine_Brand_name, Drug_name, Supplier, no_units, Uinit_Price, etc.,)

SUPPLIER :(Supplier_name, Medicine_Brand_name, Address, etc.,)

If each Dimension has 6 levels, decide the levels and hierarchies, Assume the level names suitably.

Design the Hospital Management system data warehouse using all schemas. Give the example 4-D cube with assumption names.

Data Warehousing

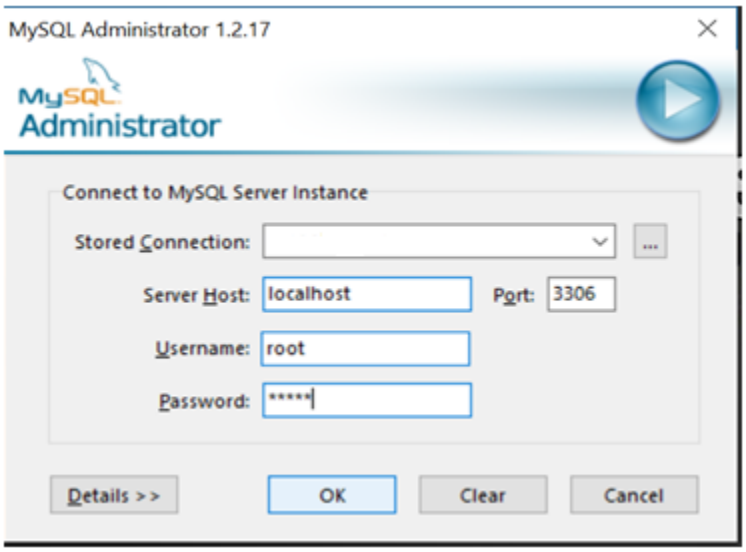
Experiments:

1. Build Data Warehouse and Explore WEKA

A. Build a Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects, etc.).

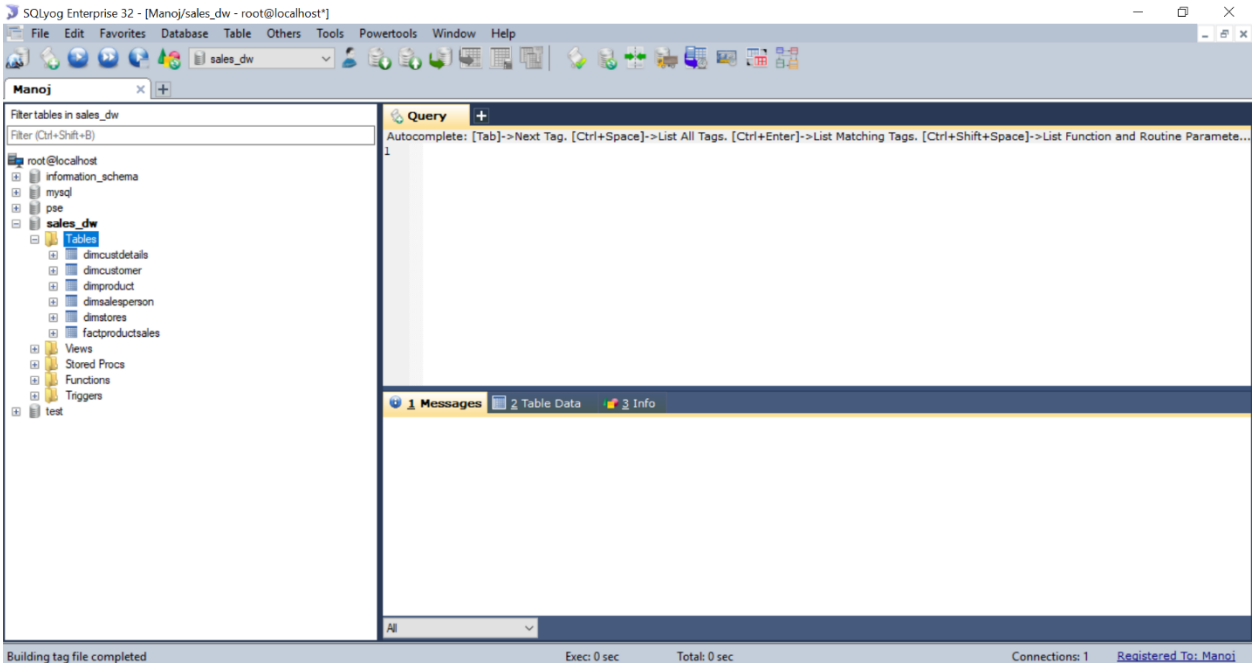
(i). Identify source tables and populate sample data

In this task, we are going to use **MySQL administrator, SQLyog Enterprise tools** for building & identifying tables in database & also for populating (filling) the sample data in those tables of a database.A data warehouse is constructed by integrating data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries and decision making. We are building a data warehouse by integrating all the tables in database & analyzing those data. In the below figure we represented MySQL Administrator connection establishment.

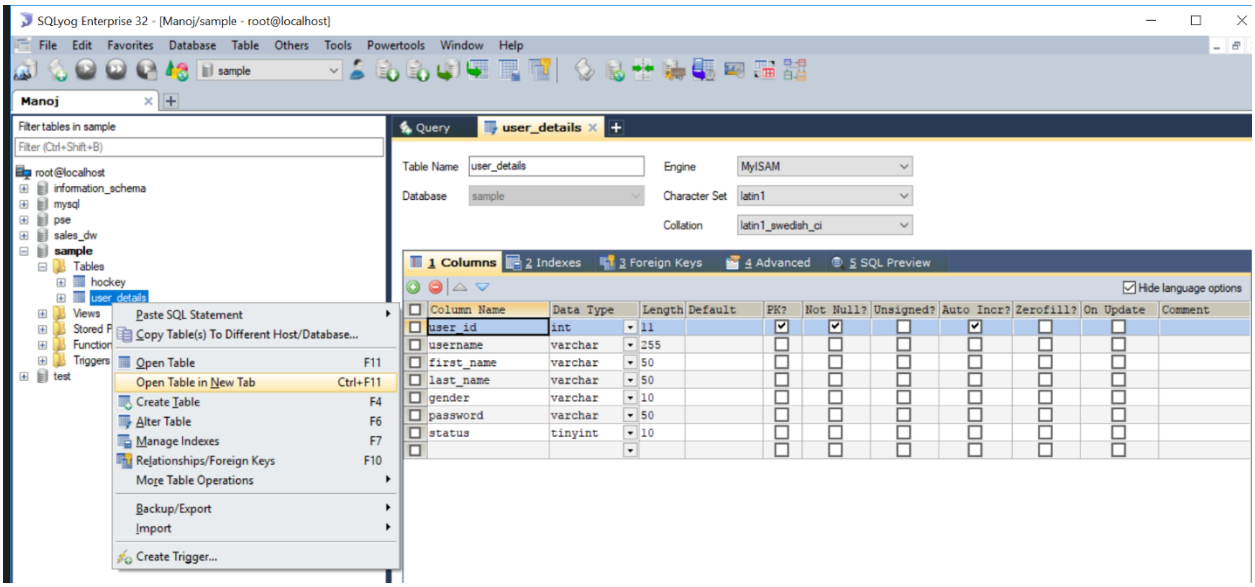


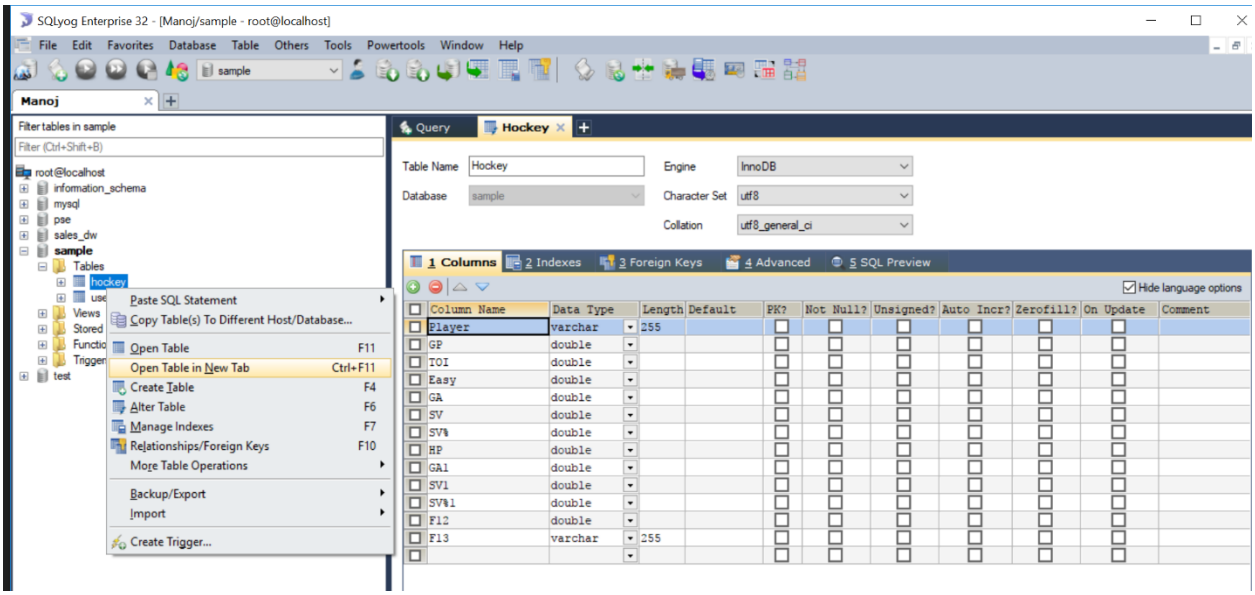
After successful login, it will open new window as shown below.

There are different options available in MySQL administrator. Another tool SQLyog Enterprise, we are using for building & identifying tables in a database after successful connection establishment through MySQL Administrator. Below we can see the window of SQLyog Enterprise.



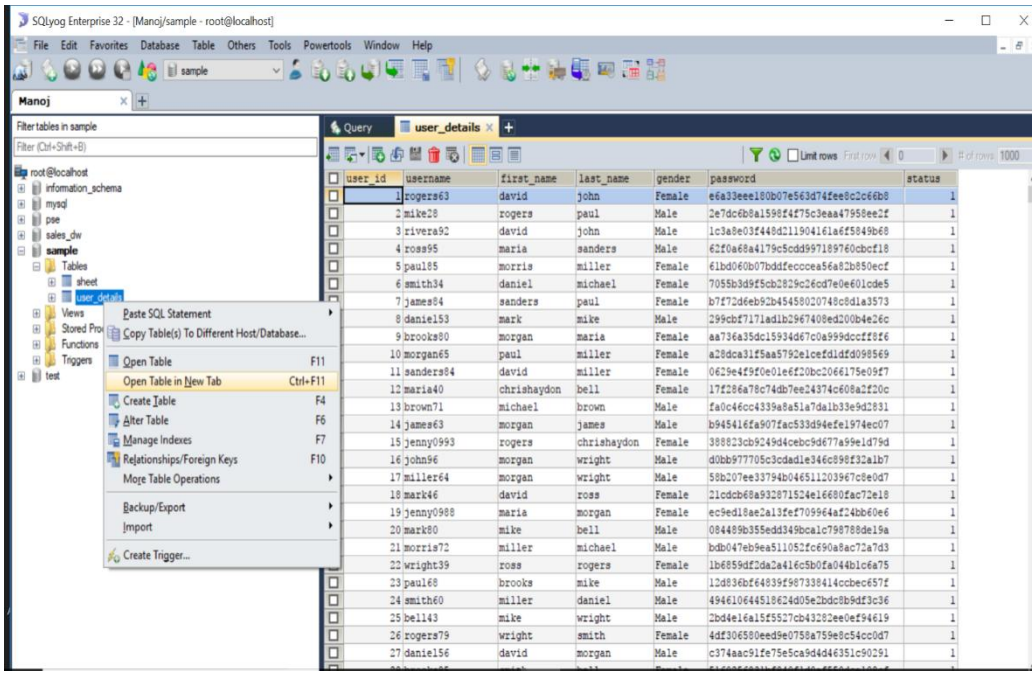
On left-side navigation, we can see different databases & it's related tables. Now we are going to build tables & populate table's data in database through SQL queries. These tables in database can be used further for building data warehouse.

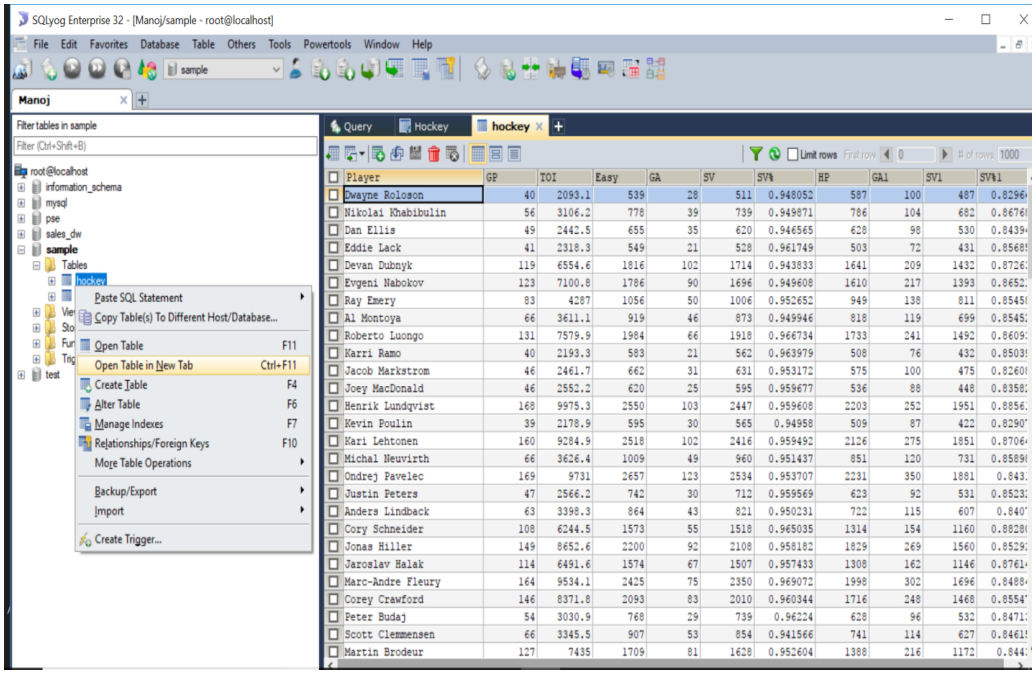




In the above two windows, we created a database named **"sample"**& in that database we created two tables named as **"user_details"**& **"hockey"**through SQL queries.

Now, we are going to populate (filling) sample data through SQL queries in those two created tables as represented in below windows.





Through MySQL administrator & SQLyog, we can import databases from other sources (.XLS, .CSV, .sql) & also we can export our databases as backup for further processing. We can connect MySQL to other applications for data analysis & reporting.

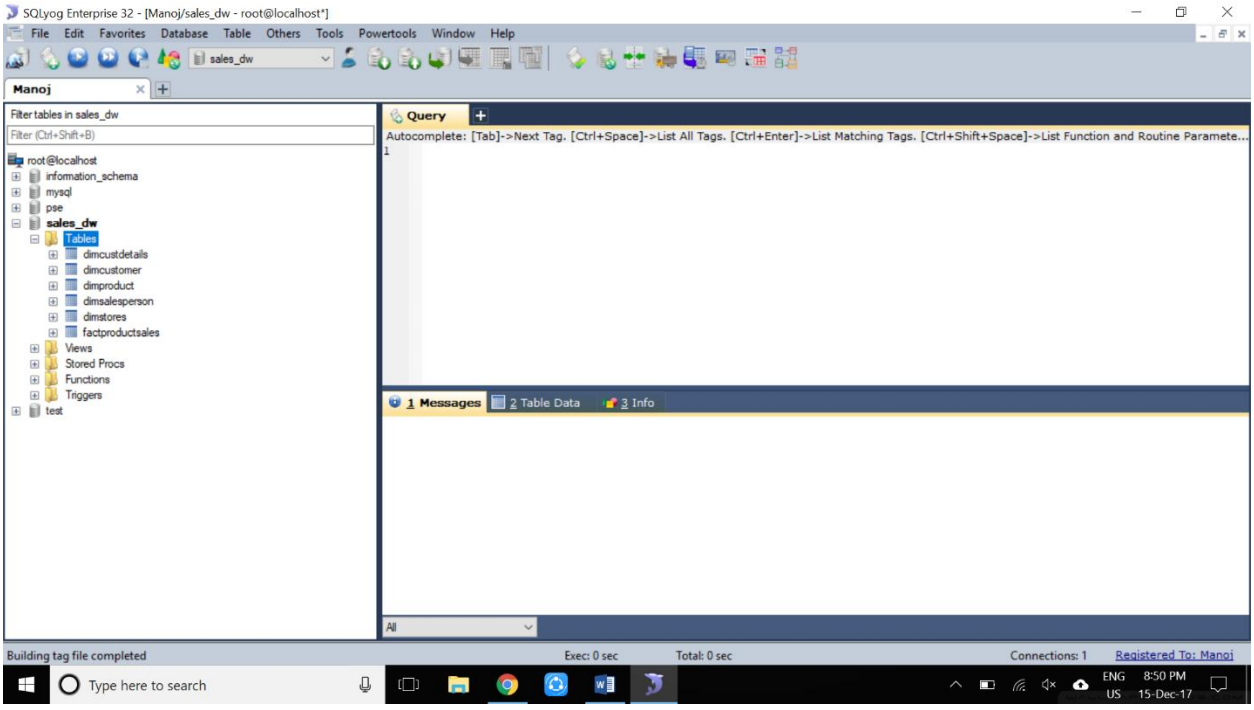
(ii). Design multi-dimensional data models namely Star, snowflake and Fact constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, Manufacturing, Automobile, etc.).

Multi-Dimensional model was developed for implementing data warehouses & it provides both a mechanism to store data and a way for business analysis. The primary components of dimensional model are dimensions & facts. There are different of types of multi-dimensional data models. They are:

1. Star Schema Model
2. Snow Flake Schema Model
3. Fact Constellation Model.

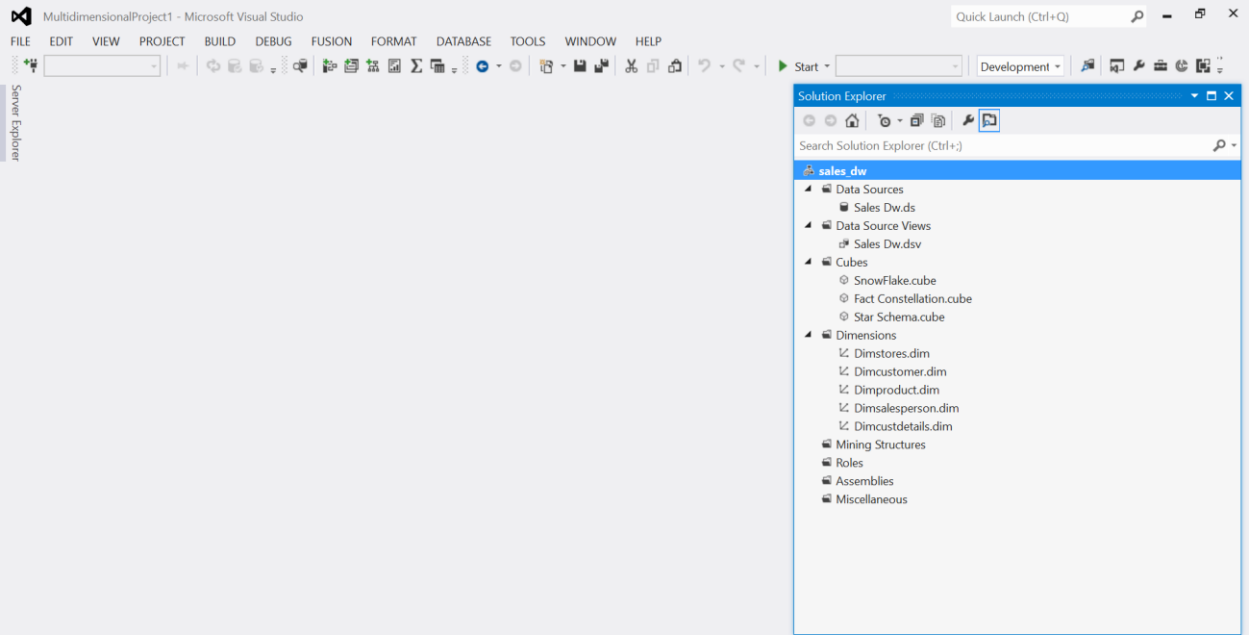
Now, we are going to design these multi-dimensional models for the Marketing enterprise.

First, we need to built the tables in a database through SQLyog as shown below.



In the above window, left side navigation bar consists of a database named as "sales_dw" in which there are six different tables (dimcustdetails, dimcustomer, dimproduct, dimsalesperson, dimstores, factproductsales) has been created.

After creating tables in database, here we are going to use a tool called as **"Microsoft Visual Studio 2012 for Business Intelligence"** for building multi-dimensional models.



In the above window, we are seeing Microsoft Visual Studio before creating a project In which right side navigation bar contains different options like Data Sources, Data Source Views, Cubes, Dimensions etc.

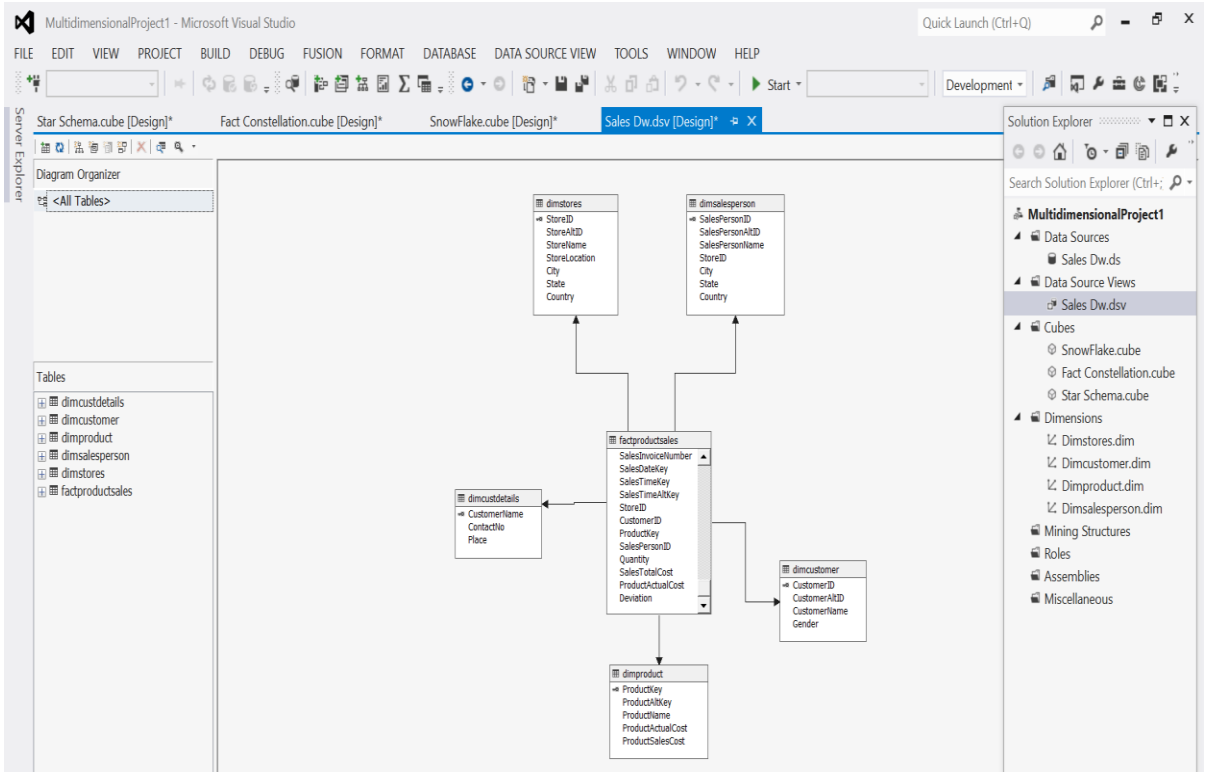
Through Data Sources, we can connect to our MySQL database named as **"sales_dw".** Then, automatically all the tables in that database will be retrieved to this tool for creating multi-dimensional models.

By data source views & cubes, we can see our retrieved tables in multi-dimensional models. We need to add dimensions also through dimensions option. In general, Multi-dimensional models consists of dimension tables & fact tables.

Star Schema Model:

A Star schema model is a join between a fact table and a no. of dimension tables. Each dimensional table are joined to the fact table using primary key to foreign key join but dimensional tables are not joined to each other. It is the simplest style of dataware house schema.

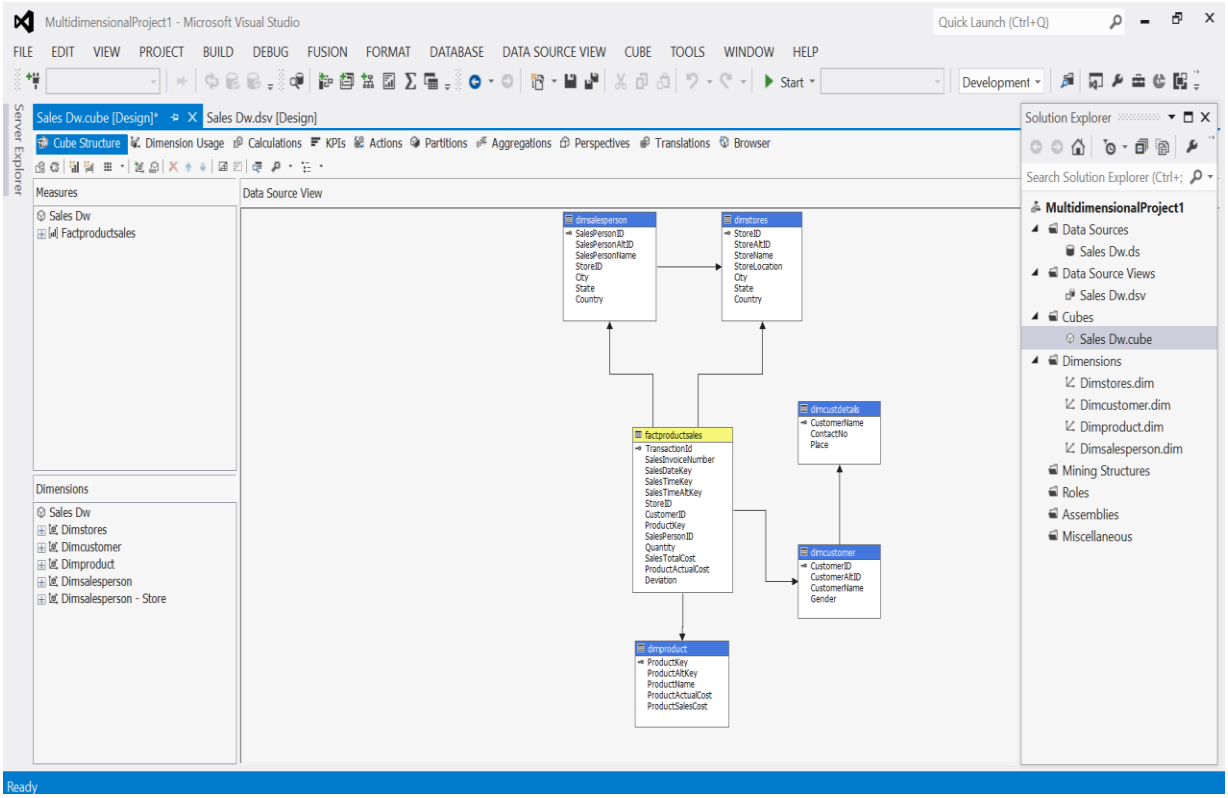
Star schema is a entity relationship diagram of this schema resembles a star with point radiating from central table as we seen in the below implemented window in visual studio.



Snow Flake Schema:

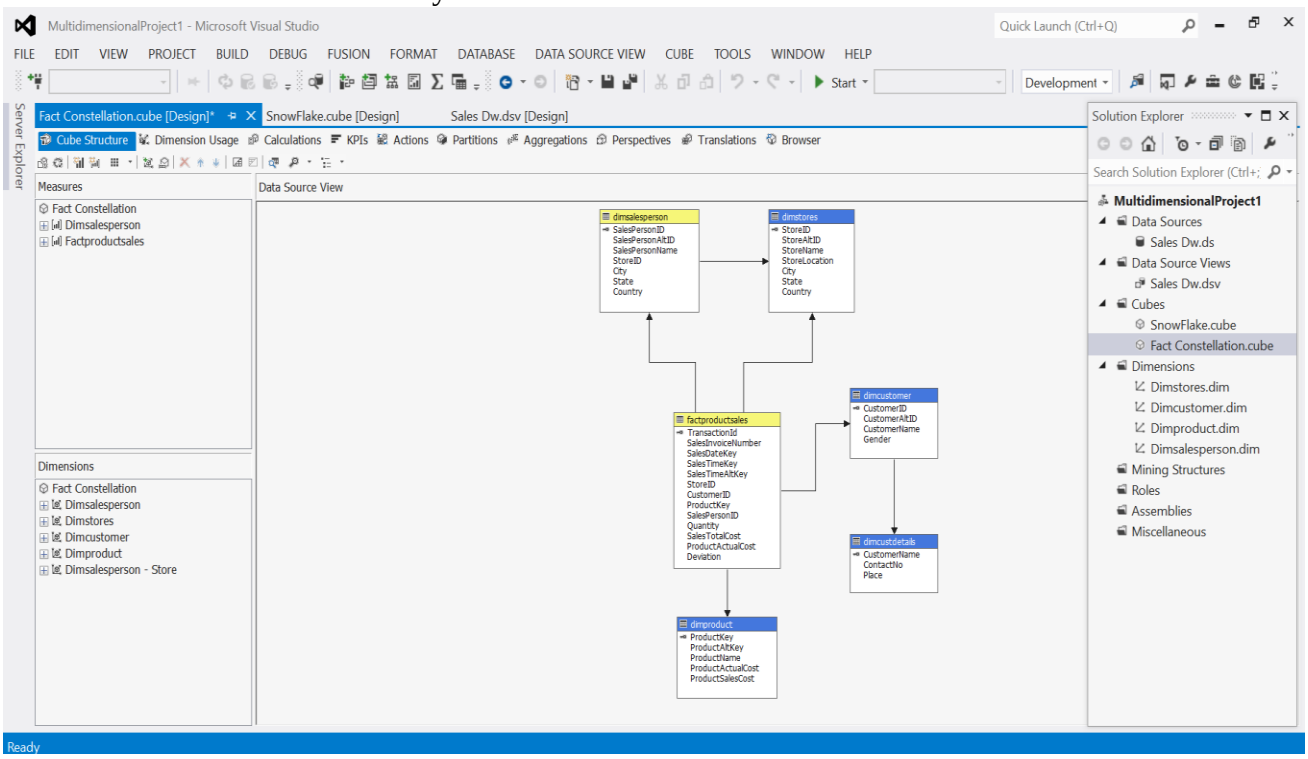
It is slightly different from star schema in which dimensional tables from a star schema are organized into a hierarchy by normalizing them.

Snow flake schema is represented by centralized fact table which are connected to multiple dimension tables. Snow flake effects only dimension tables not fact tables. we developed a snowflake schema for sales_dw database by visual studio tool as shown below.



Fact Constellation Schema:

Fact Constellation is a set of fact tables that share some dimension tables. In this schema there are two or more fact tables. We developed fact constellation in visual studio as shown below. Fact tables are labelled in yellow color.



2. Write ETL scripts and implement using data warehouse tools

ETL (Extract-Transform-Load):

ETL comes from Data Warehousing and stands for Extract-Transform-Load. ETL covers a process of how the data are loaded from the source system to the data warehouse. Currently, the ETL encompasses a cleaning step as a separate step. The sequence is then Extract-Clean-Transform-Load. Let us briefly describe each step of the ETL process.

Process

Extract:

The Extract step covers the data extraction from the source system and makes it accessible for further processing. The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible. The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

- Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.
- Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.
- Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.

When using Incremental or Full extracts, the extract frequency is extremely important. Particularly for full extracts; the data volumes can be in tens of gigabytes.

Clean:

The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

- Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
- Convert null values into standardized Not Available/Not Provided value
- Convert phone numbers, ZIP codes to a standardized form
- Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
- Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

Transform:

The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load:

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Managing ETL Process:

The ETL process seems quite straight forward. As with every application, there is a possibility that the ETL process fails. This can be caused by missing extracts from one of the systems, missing values in one of the reference tables, or simply a connection or power outage. Therefore, it is necessary to design the ETL process keeping fail-recovery in mind.

Staging:

It should be possible to restart, at least, some of the phases independently from the others. For example, if the transformation step fails, it should not be necessary to restart the Extract step. We can ensure this by implementing proper staging. Staging means that the data is simply dumped to the location (called the Staging Area) so that it can then be read by the next processing phase. The staging area is also used during ETL process to store intermediate results of processing. This is ok for the ETL process which uses for this purpose. However, tThe staging area should is be accessed by the load ETL process only. It should never be available to anyone else; particularly not to end users as it is not intended for data presentation to the end-user.may contain incomplete or in-the-middle-of-the-processing data.

ETL Tool Implementation:

When you are about to use an ETL tool, there is a fundamental decision to be made: will the company build its own data transformation tool or will it use an existing tool?

Building your own data transformation tool (usually a set of shell scripts) is the preferred approach for a small number of data sources which reside in storage of the same type. The reason for that is the effort to implement the necessary transformation is little due to similar data structure and common system architecture. Also, this approach saves licensing cost and there is no need to train the staff in a new tool. This approach, however, is dangerous from the TOC point of view. If the transformations become more sophisticated during the time or there is a need to integrate other systems, the complexity of such an ETL system grows but the manageability drops significantly. Similarly, the implementation of your own tool often resembles re-inventing the wheel.

There are many ready-to-use ETL tools on the market. The main benefit of using off-the-shelf ETL tools is the fact that they are optimized for the ETL process by providing connectors to common data sources like databases, flat files, mainframe systems, xml, etc. They provide a means to implement data transformations easily and consistently across various data sources. This includes filtering, reformatting, sorting, joining, merging, aggregation and other operations ready to use. The tools also support transformation scheduling, version control, monitoring and unified metadata management. Some of the ETL tools are even integrated with BI tools.

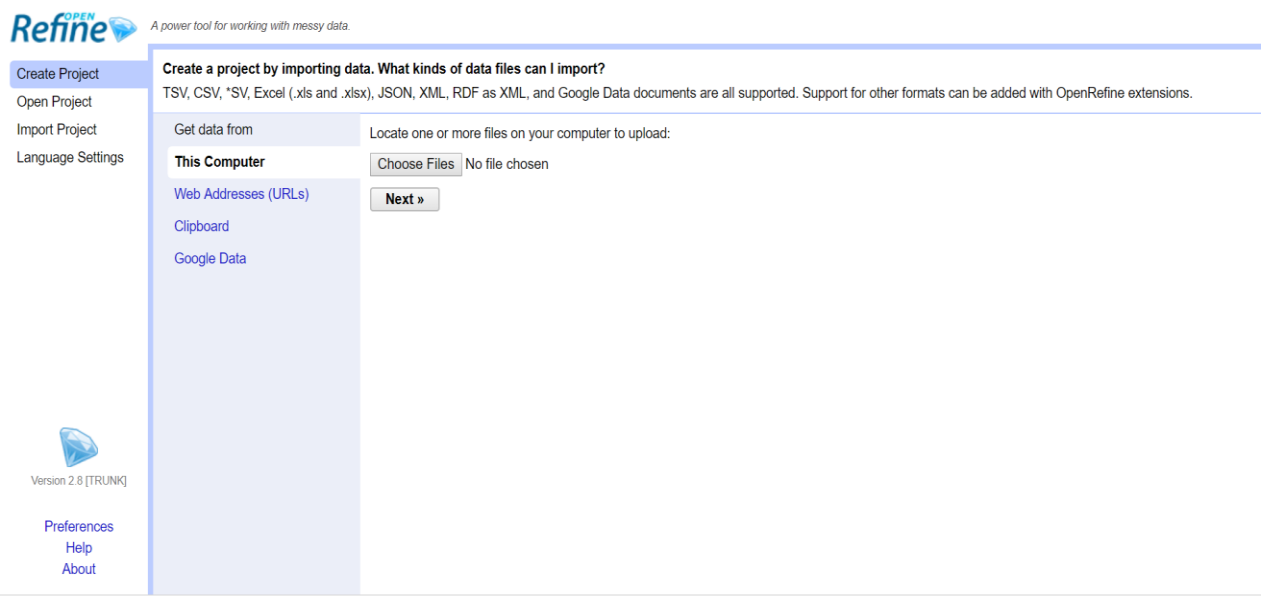
Some of the Well Known ETL Tools:

The most well-known commercial tools are Ab Initio, IBM InfoSphere DataStage, Informatica, Oracle Data Integrator, and SAP Data Integrator.

There are several open source ETL tools are **OpenRefine**,

Apatar, CloverETL, Pentaho and Talend.

In these above tools, we are going to use **OpenRefine 2.8 ETL tool**to different sample datasets forextracting, data cleaning, transforming & loading.



(iv). Perform various OLAP operations such slice, dice, roll up, drill down and pivot.

OLAP Operations:

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data.

Here is the list of OLAP operations

- Roll-up (Drill-up)
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up (Drill-up):

Roll-up performs aggregation on a data cube in any of the following ways

- By climbing up a concept hierarchy for a dimension
- By dimension reduction
- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.

- When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down:

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.
- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice:

The slice operation selects one particular dimension from a given cube and provides a new sub-cube.

Dice:

Dice selects two or more dimensions from a given cube and provides a new sub-cube.

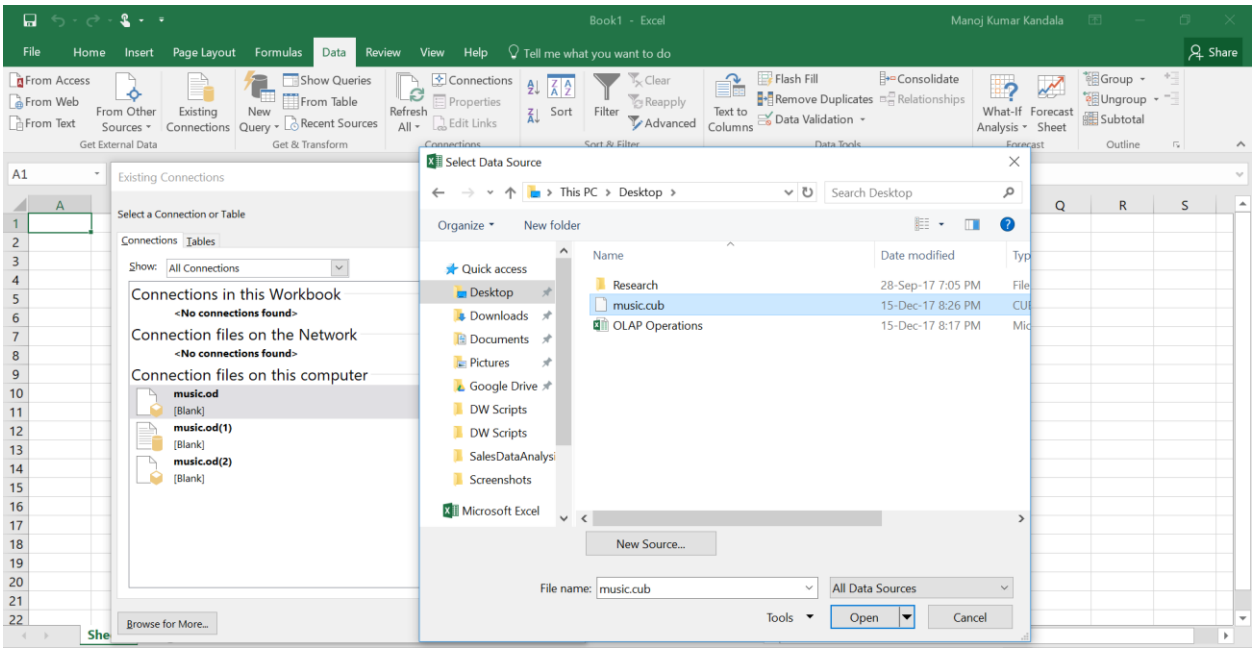
Pivot (rotate):

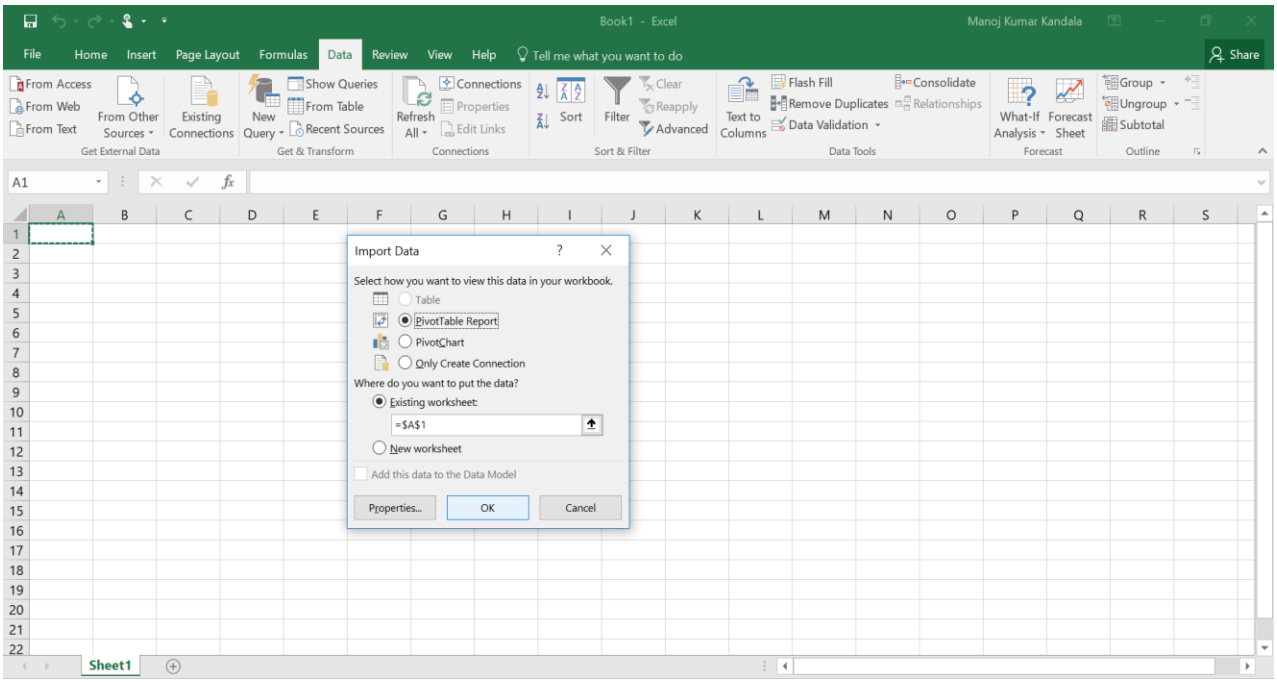
The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data.

Now, we are practically implementing all these OLAP Operations using **Microsoft Excel**.

Procedure for OLAP Operations:

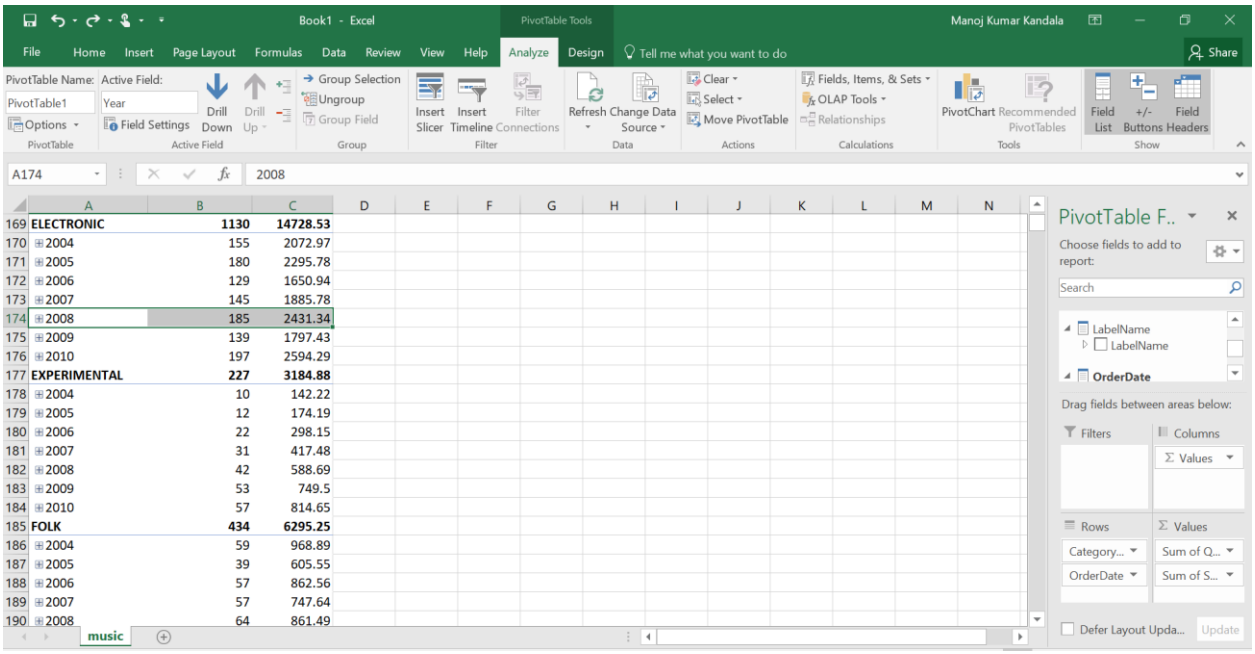
1.Open Microsoft Excel, go to**Data** tab in top & click on "**Existing Connections"**.
2. Existing Connections window will be opened, there **"Browse for more"**option should be clicked for importing **.cub extension file** for performing OLAP Operations. For sample, I took **music.cub** file.



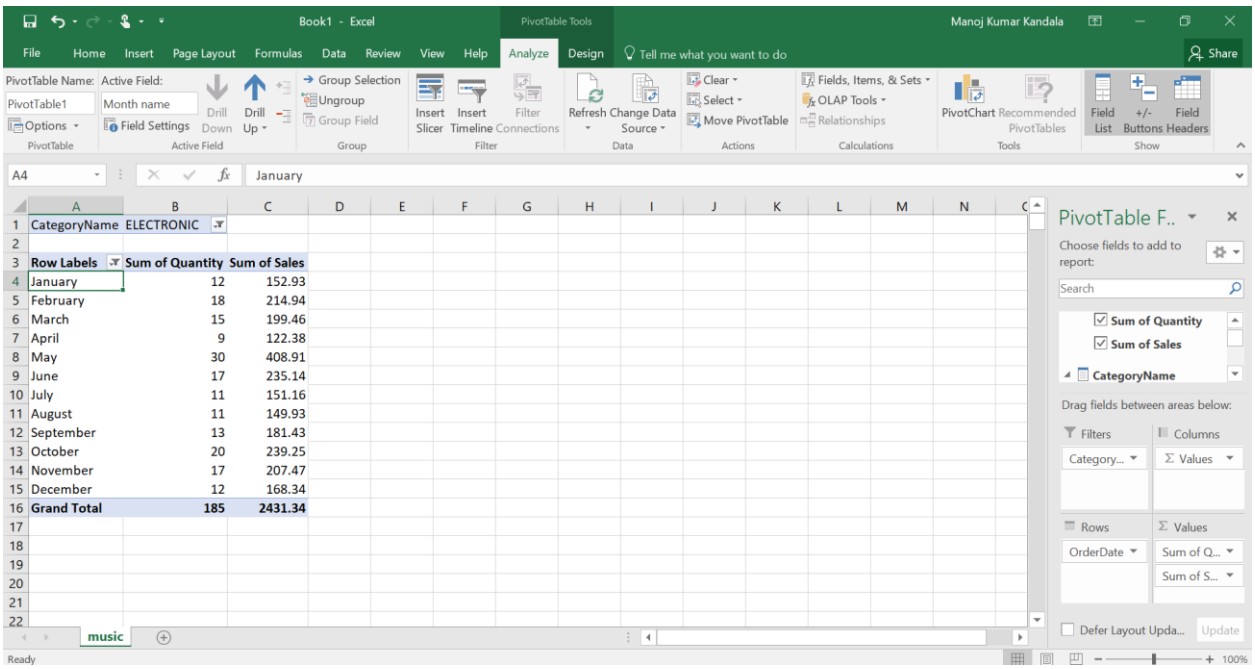


3.As shown in above window, select "**PivotTable Report"** and click **"OK".**

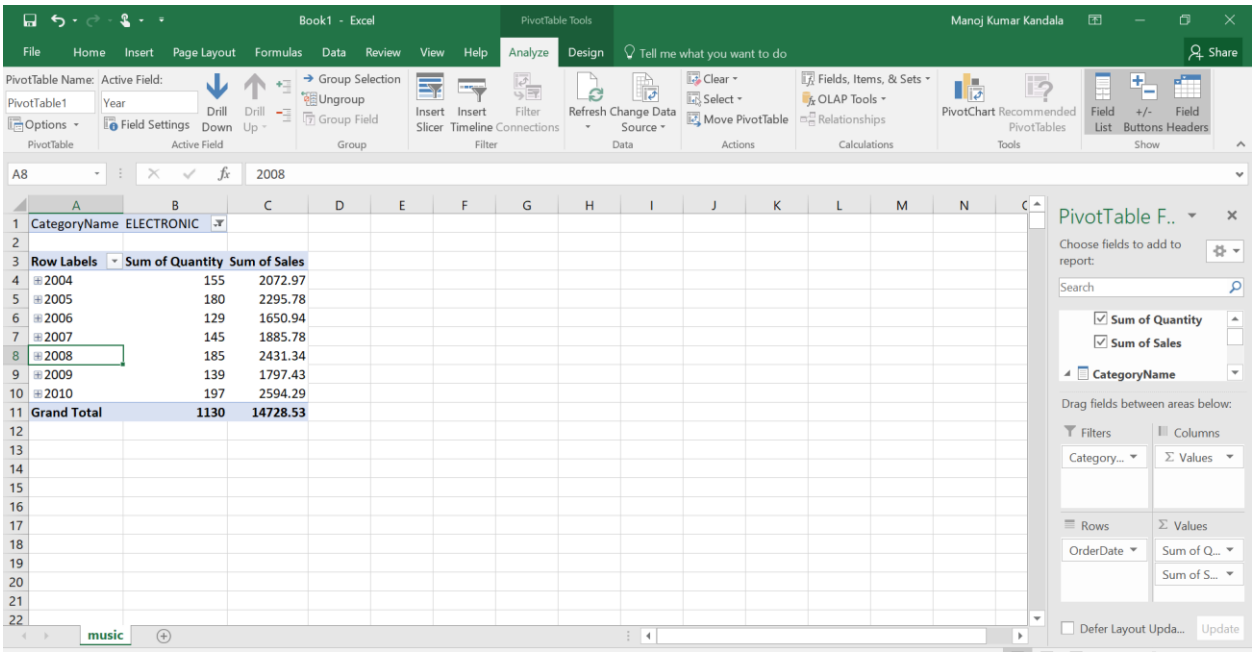
4.We got all the **music.cub** data for analyzing different OLAP Operations.Firstly, we performed **drill-down operation** as shown below.



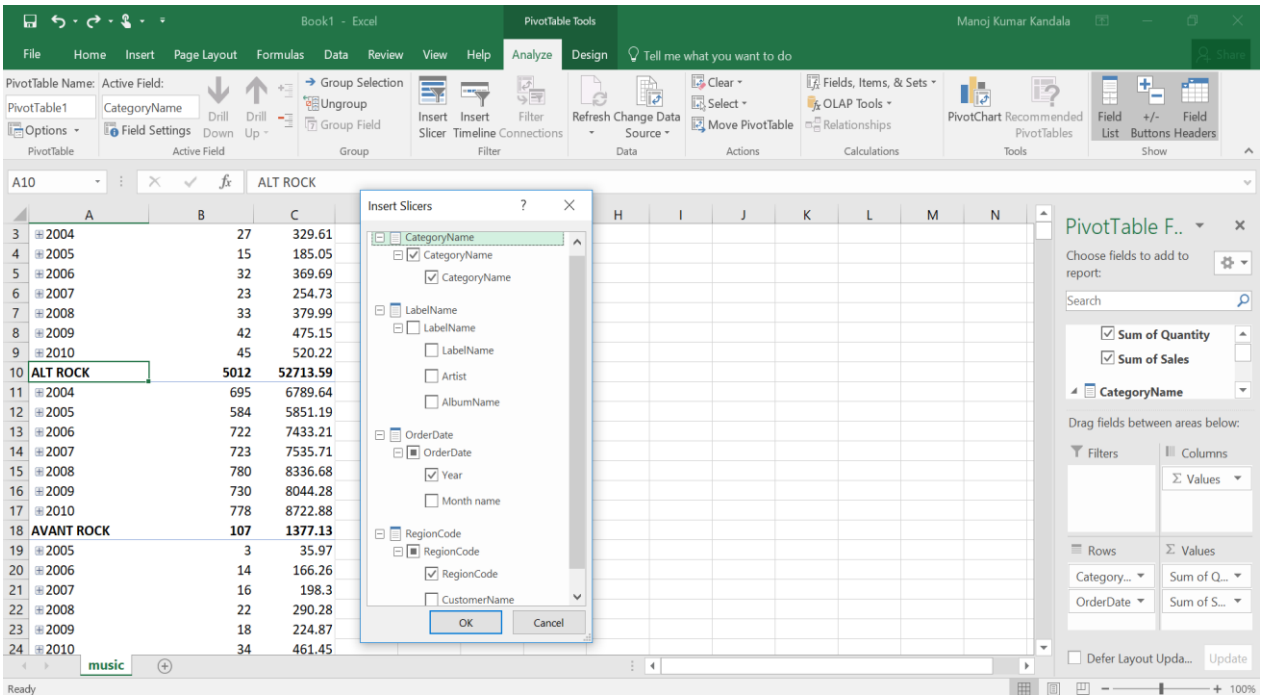
In the above window, we selected year **'2008'** in **'Electronic'** Category, then automatically the Drill-Down option is enabled on top navigation options. We will click on **'Drill-Down'** option, then the below window will be displayed.



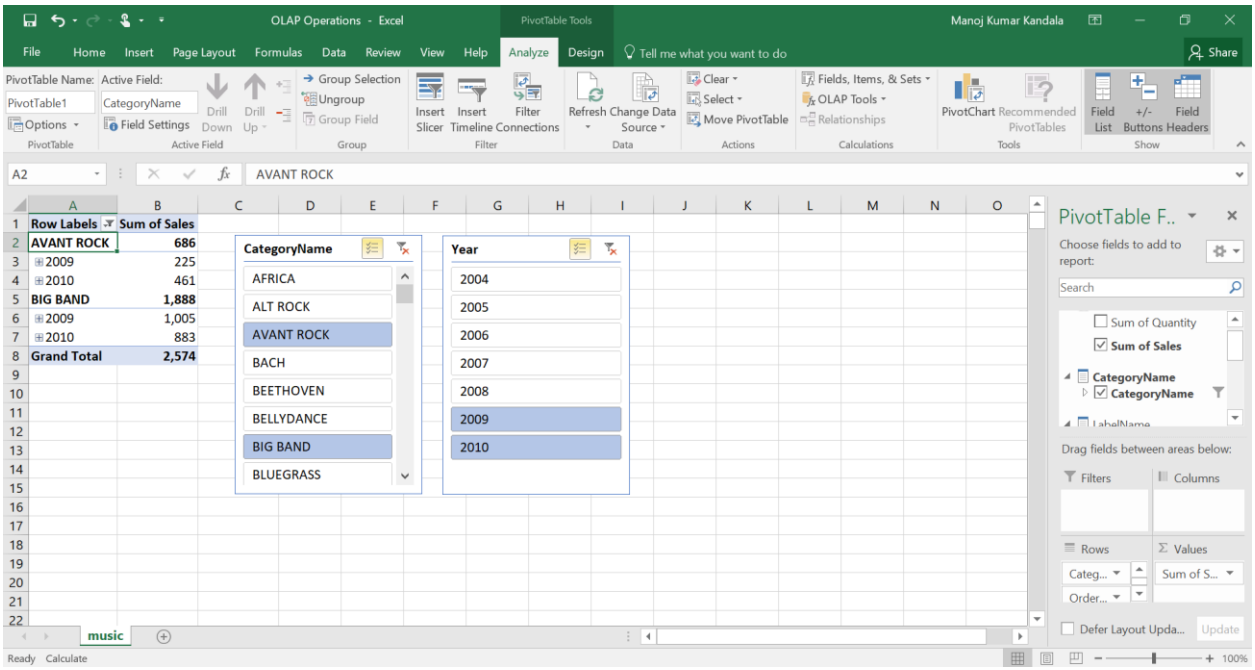
5. Now we are going to perform **roll-up (drill-up) operation**, in the above window I selected January month then automatically **Drill-up option** is enabled on top. We will click on **Drill-up** option, then the below window will be displayed.



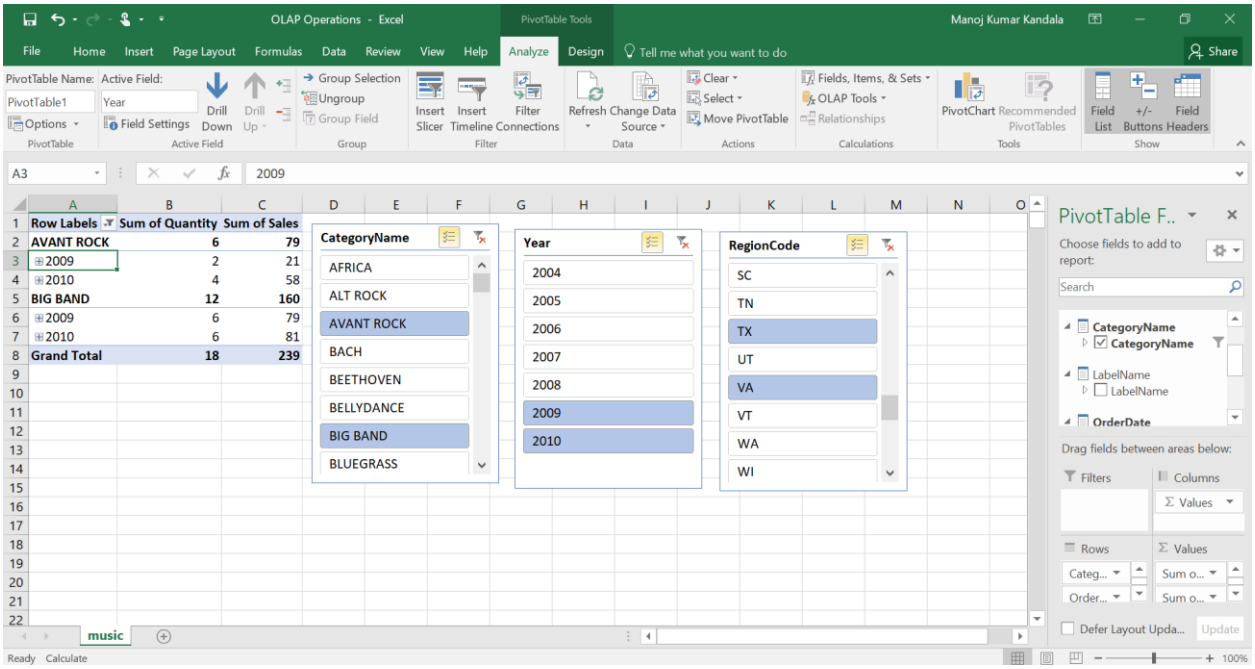
6. Next OLAP operation **Slicing** is performed by inserting slicer as shown in top navigation options.



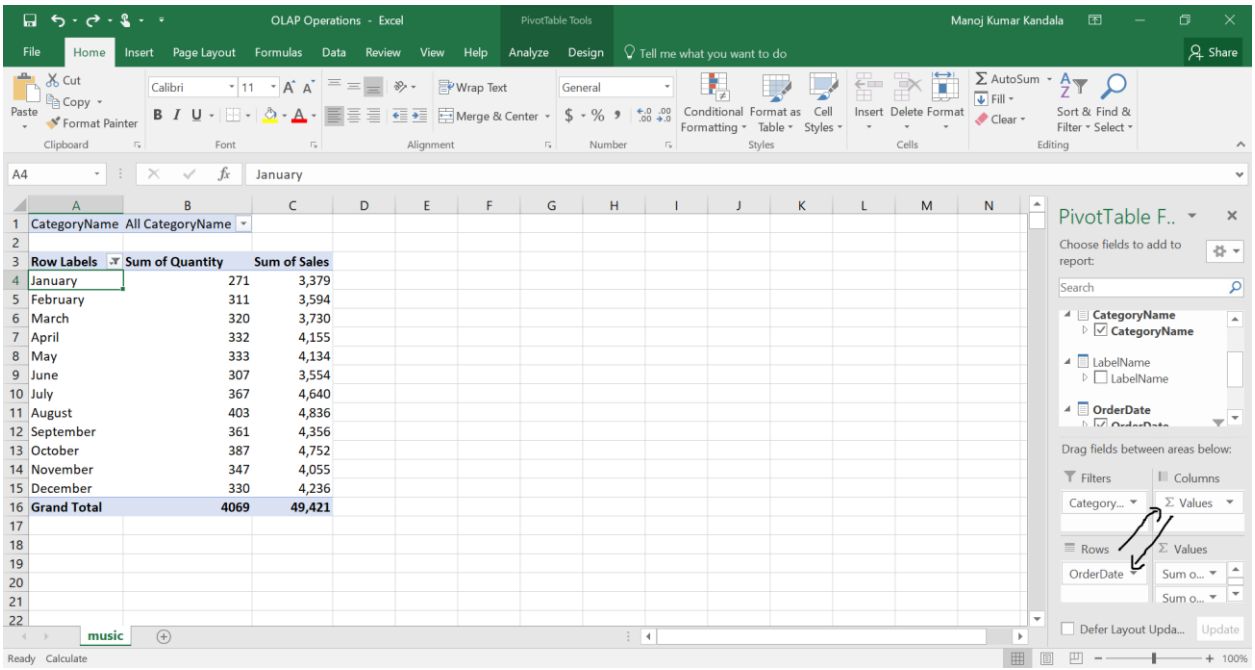
While inserting slicers for slicing operation, we select 2 Dimensions (for e.g. CategoryName & Year) only with one Measure (for e.g. Sum of sales).After inserting a slice& adding a filter (CategoryName: AVANT ROCK & BIG BAND; Year: 2009 & 2010), we will get table as shown below.



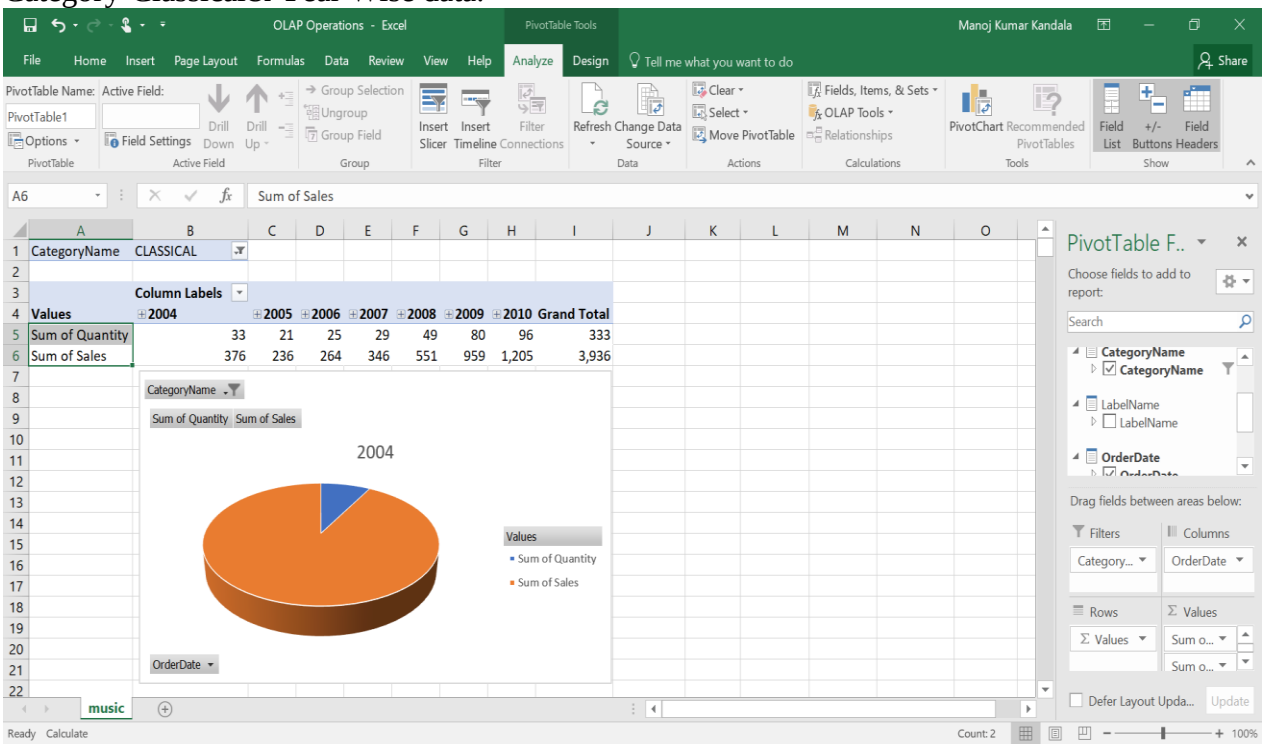
7. Dicing operation is similar to Slicing operation. Here we are selecting 3 dimensions (CategoryName, Year, RegionCode)& 2 Measures (Sum of Quantity, Sum of Sales) through **'insert slicer'** option. After that adding a filter for CategoryName, Year & RegionCode as shown below.



8. Finally, the **Pivot (rotate)** OLAP operation is performed by swapping rows (Order Date-Year) & columns (Values-Sum of Quantity & Sum of Sales) through right side bottom navigation bar as shown below.



After Swapping (rotating), we will get resultant as represented below with a pie-chart for Category-Classical& Year Wise data.



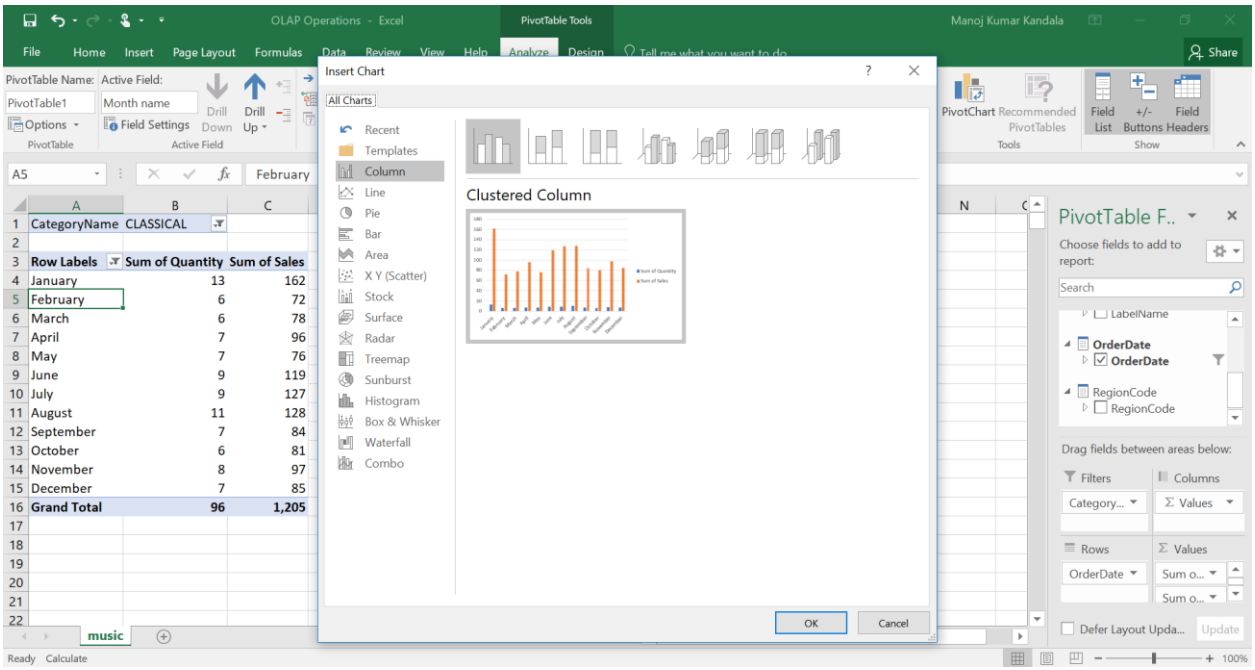
(v). Explore visualization features of the tool for analysis like identifying trends etc.

There are different visualization features for analyzing the data for trend analysisin data warehouses. Some of the popular visualizations are:

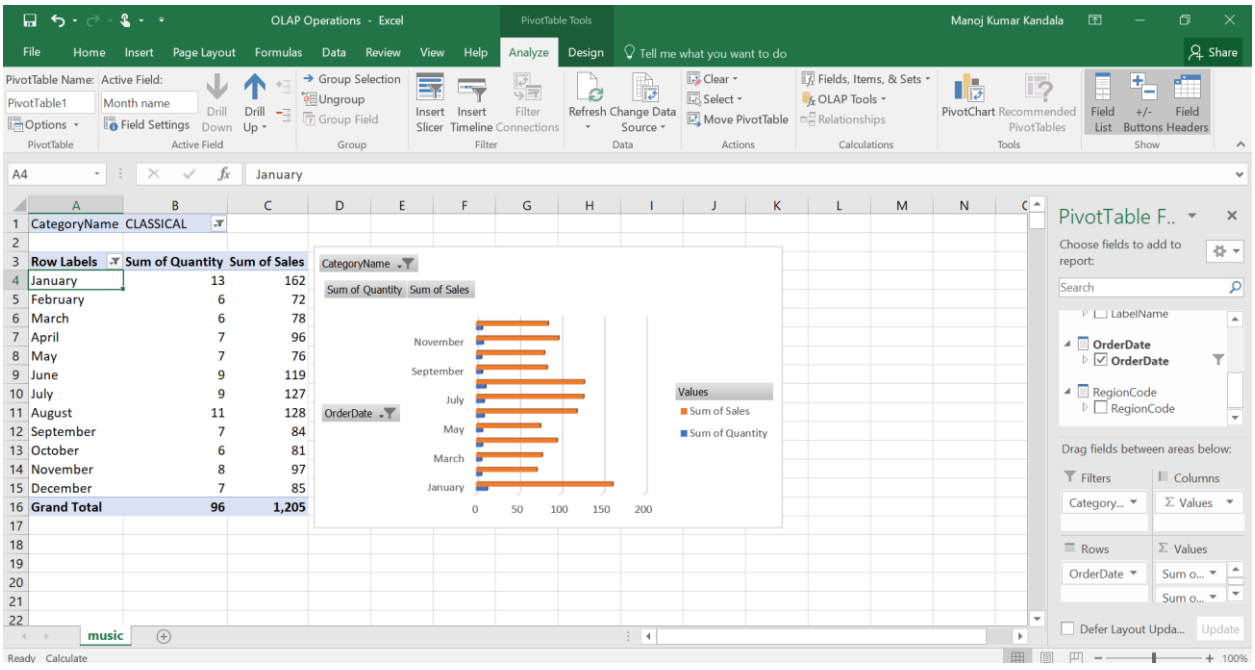
1. Column Charts

2. Line Charts
3. Pie Chart
4. Bar Graphs
4. Area Graphs
5. X & Y Scatter Graphs
6. Stock Graphs
7. Surface Charts
8. Radar Graphs
9. Treemap
10. Sunburst
11. Histogram
12. Box & Whisker
13. Waterfall
14. Combo Graphs
15. Geo Map
16. Heat Grid
17. Interactive Report
18. Stacked Column
19. Stacked Bar
20. Scatter Area

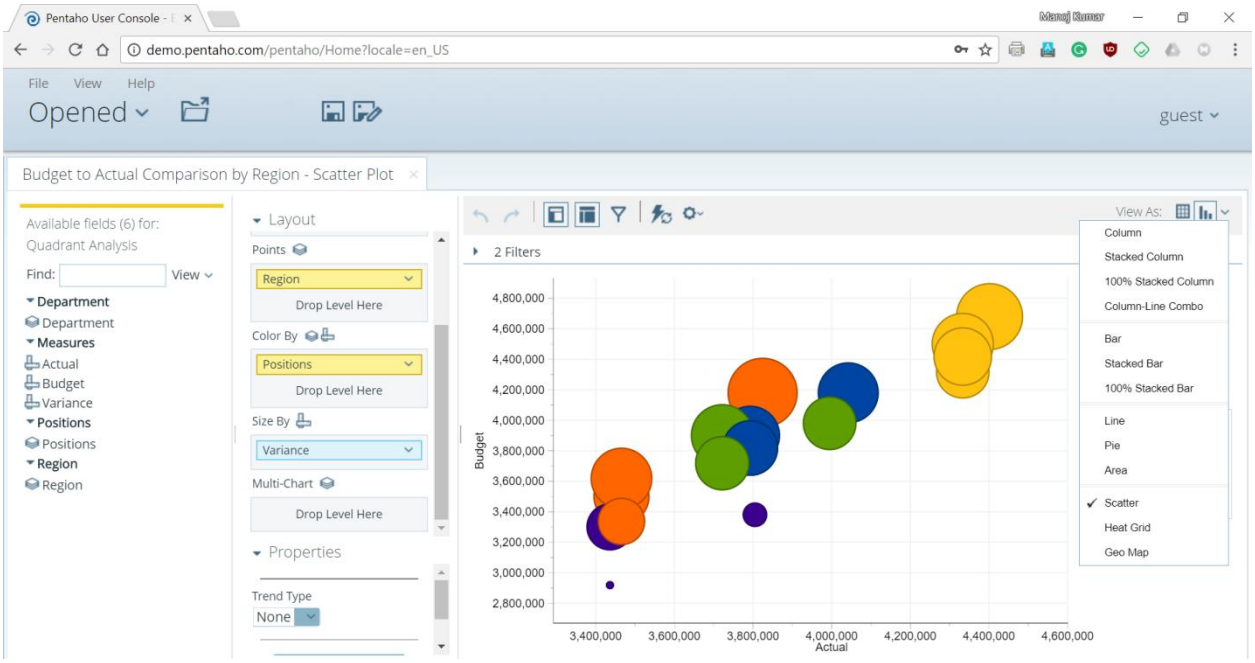
These type of visualizations can be used for analyzing data for trend analysis. Some of the tools for data visualization are Microsoft Excel, Tableau, Pentaho Business Analytics Online etc. Practically different visualization features are tested with different sample datasets.



In the below window, we used 3D-Column Charts of **Microsoft Excel** for analyzing data in data warehouse.



Below window, represents the data visualization through **Pentaho Business Analytics tool online** (http://www.pentaho.com/hosted-demo) for some sample dataset.



B. Explore WEKA Data Mining/Machine Learning Toolkit

(i). Downloading and/or installation of WEKA data mining toolkit

Procedure:

1. Go to the Weka website, http://www.cs.waikato.ac.nz/ml/weka/, and download the software. On the left-hand side, click on the link that says download.

2. Select the appropriate link corresponding to the version of the software based on your operating system and whether or not you already have Java VM running on your machine (if you don't know what Java VM is, then you probably don't).

3. The link will forward you to a site where you can download the software from a mirror site. Save the self-extracting executable to disk and then double click on it to install Weka. Answer yes or next to the questions during the installation.

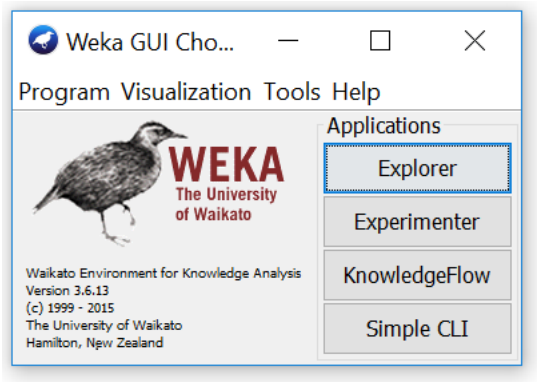
4. Click yes to accept the Java agreement if necessary. After you install the program Weka should appear on your start menu under Programs (if you are using Windows).

5. Running Weka from the start menu select Programs, then Weka.You will see the Weka GUI Chooser. Select Explorer. The Weka Explorer will then launch.

(ii). Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface.

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting pointfor launching Weka's main GUI applications and supporting tools. If one prefersa MDI ("multiple document interface") appearance, then this is provided by analternative launcher called "Main" (class weka.gui.Main).

The GUI Chooser consists of four buttons—one for each of the four majorWeka applications—and four menus.



The buttons can be used to start the following applications:

Explorer- An environment for exploring data with WEKA

a) Click on "explorer" button to bring up the explorer window.
b) Make sure the "preprocess" tab is highlighted.
c) Open a new file by clicking on "Open New file" and choosing a file with ".arff" extension from the "Data" directory.
d) Attributes appear in the window below.
e) Click on the attributes to see the visualization on the right.
f) Click "visualize all" to see them all

Experimenter- An environment for performing experiments and conducting statistical tests between learning schemes.

a) Experimenter is for comparing results.
b) Under the "set up" tab click "New".
c) Click on "Add New" under "Data" frame. Choose a couple of arff format files from "Data" directory one at a time.
d) Click on "Add New" under "Algorithm" frame. Choose several algorithms, one at a time by clicking "OK" in the window and "Add New".
e) Under the "Run" tab click "Start".

f) Wait for WEKA to finish.
g) Under "Analyses" tab click on "Experiment" to see results.

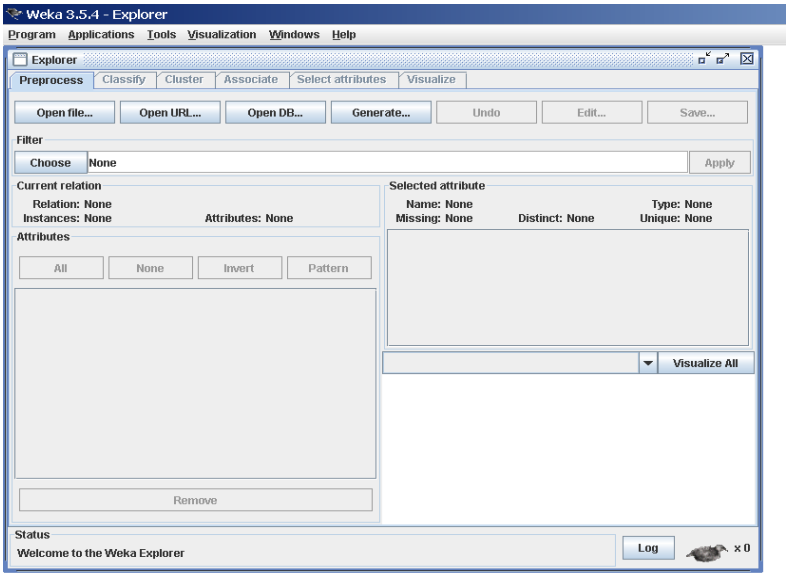
Knowledge Flow- This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantageis that it supports incremental learning.

SimpleCLI - Provides a simple command-line interface that allows directexecution of WEKA commands for operating systems that do not provide their own command line interface.

(iii). Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, classify panel, Cluster panel, Associate panel and Visualize panel)

When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.
The tabs are as follows:
1. Preprocess. Choose and modify the data being acted on.
2. Classify. Train and test learning schemes that classify or perform regression.
3. Cluster. Learn clusters for the data.
4. Associate. Learn association rules for the data.
5. Select attributes. Select the most relevant attributes in the data.
6. Visualize. View an interactive 2D plot of the data.
Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in.

1. Preprocessing



Loading Data:

The first four buttons at the top of the preprocess section enable you to loaddata into WEKA:

1. Open file.... Brings up a dialog box allowing you to browse for the datafile on the local file system.

2. Open URL.... Asks for a Uniform Resource Locator address for wherethe data is stored.

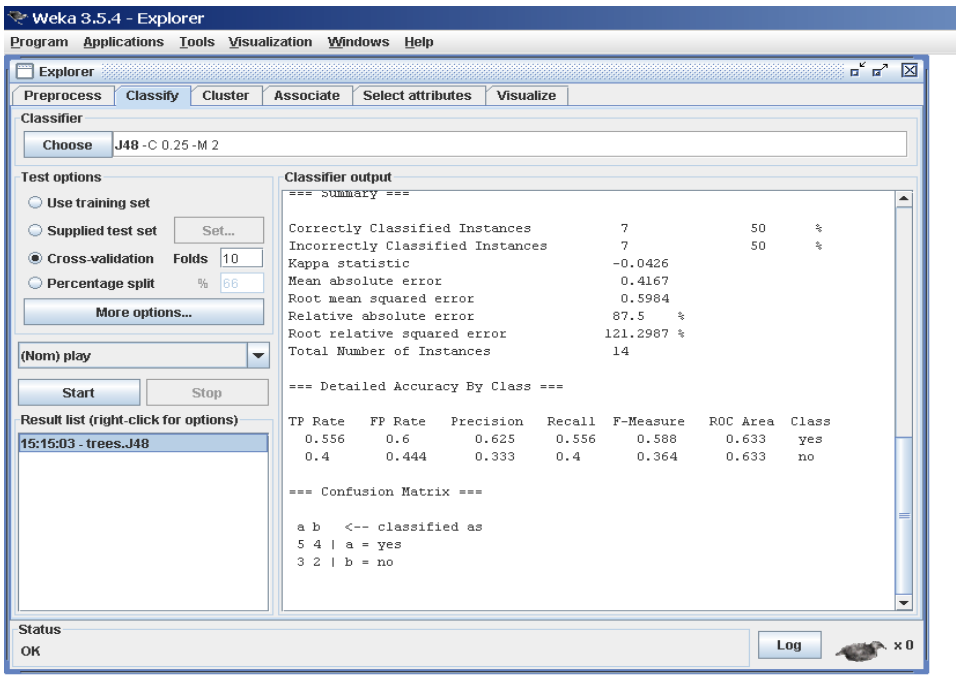
3. Open DB.... Reads data from a database. (Note that to make this workyou might have to edit the file in weka/experiment/DatabaseUtils.props.)

4. Generate.... Enables you to generate artificial data from a variety ofDataGenerators.

Using the Open file... button you can read files in a variety of formats:

WEKA's ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension,C4.5 files a .data and .names extension, and serialized Instances objects a .bsiextension.

2. Classification:



Selecting a Classifier

At the top of the classify section is the Classifier box. This box has a text fieldthat gives the name of the currently selected classifier, and its options. Clickingon the text box with the left mouse button brings up a GenericObjectEditordialog box, just the same as for filters, that you can use to configure the optionsof the current classifier. With a right click (or Alt+Shift+left click) you canonce again copy the setup string to the clipboard or display the properties in aGenericObjectEditor dialog box. The Choose button allows you to choose on4eof the classifiers that are available in WEKA.

Test Options

The result of applying the chosen classifier will be tested according to the optionsthat are set by clicking in the Test options box. There are four test modes:

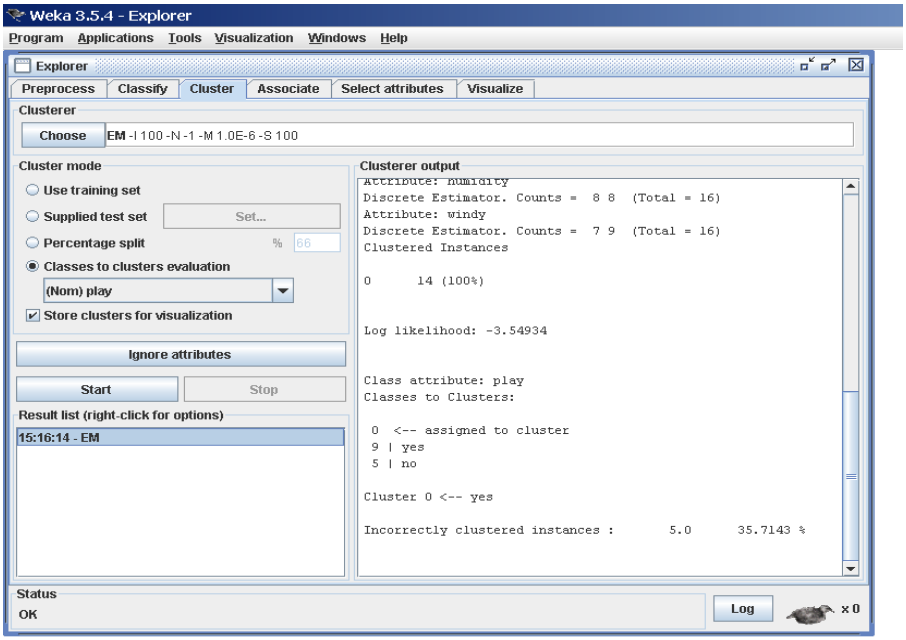
1. **Use training set:** The classifier is evaluated on how well it predicts theclass of the instances it was trained on.

2. **Supplied test set:** The classifier is evaluated on how well it predicts theclass of a set of instances loaded from a file. Clicking the Set... buttonbrings up a dialog allowing you to choose the file to test on.

3**. Cross-validation:** The classifier is evaluated by cross-validation, usingthe number of folds that are entered in the Folds text field.

4. **Percentage split:** The classifier is evaluated on how well it predicts acertain percentage of the data which is held out for testing. The amountof data held out depends on the value entered in the % field.

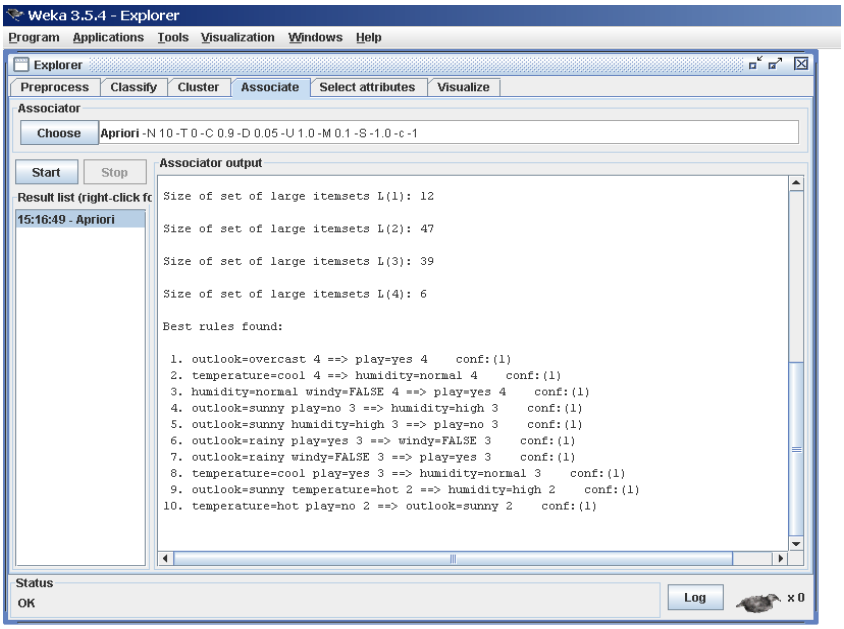
3. Clustering:



Cluster Modes:

The Cluster mode box is used to choose what to cluster and how to evaluatethe results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split.

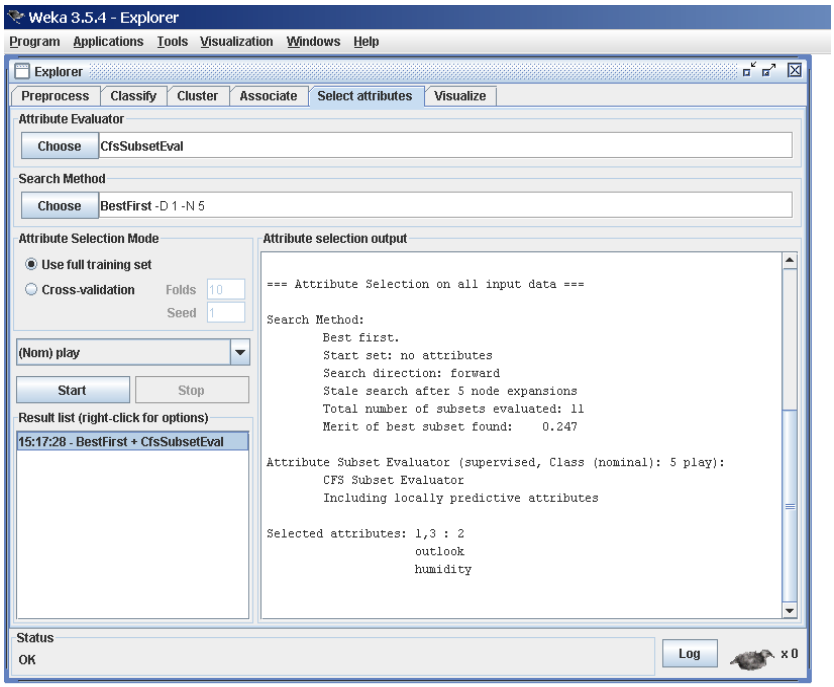
4. Associating:



Setting Up

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers in the other panels.

5. Selecting Attributes:

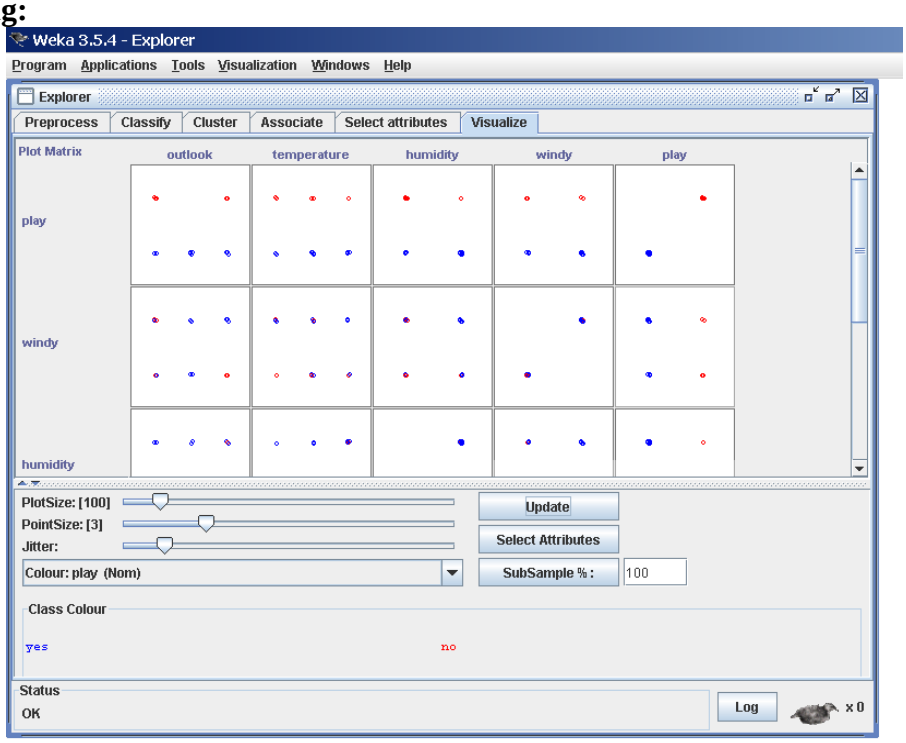


Searching and Evaluating

Attribute selection involves searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction. To do this, two objects must be set up: an attribute evaluator and a searchmethod. The evaluator determines what method is used to

assign a worth toeach subset of attributes. The search method determines what style of searchis performed.

6. Visualizing:



WEKA's visualization section allows you to visualize 2D plots of the current relation.

(iv). Study the arff file format

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.

Overview

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.

The **Header** of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
```

```
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

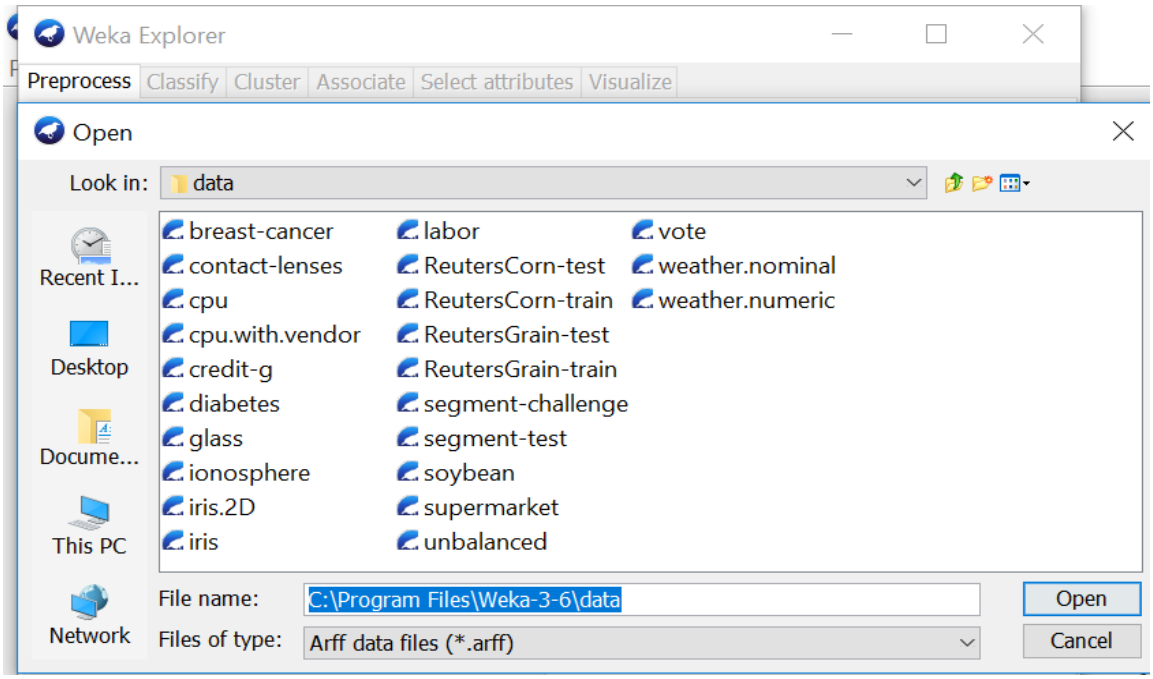
The **Data** of the ARFF file looks like the following:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Lines that begin with a % are comments.
The **@RELATION**, **@ATTRIBUTE** and **@DATA** declarations are case insensitive.

(v). Explore the available data sets in WEKA

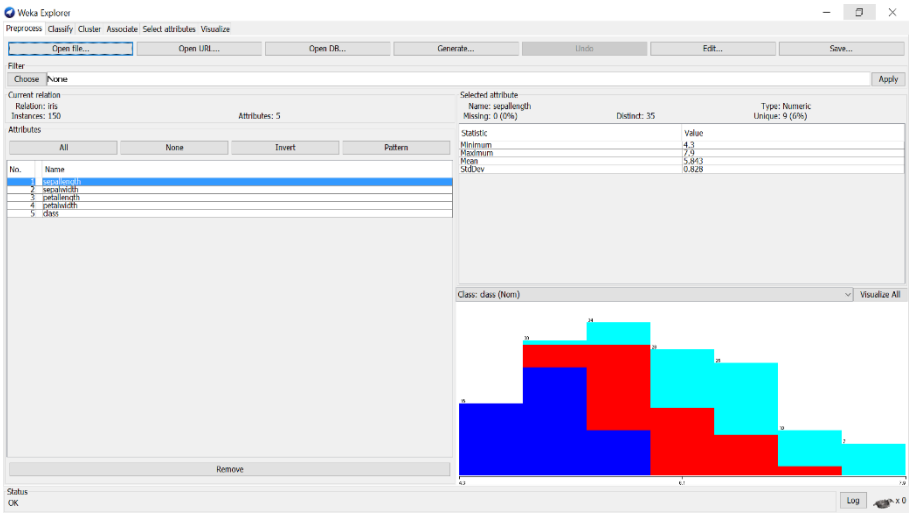
There are 23 different datasets are available in weka (C:\Program Files\Weka-3-6\) by default for testing purpose. All the datasets are available in. arff format. Those datasets are listed below.



(vi). Load a data set (ex. Weather dataset, Iris dataset, etc.)

Procedure:

1. Open the weka tool and select the explorer option.
2. New window will be opened which consists of different options (Preprocess, Association etc.)
3. In the preprocess, click the "open file" option.
4. Go to C:\Program Files\Weka-3-6\data for finding different existing. arff datasets.
5. Click on any dataset for loading the data then the data will be displayed as shown below.



(vii). Load each dataset and observe the following:

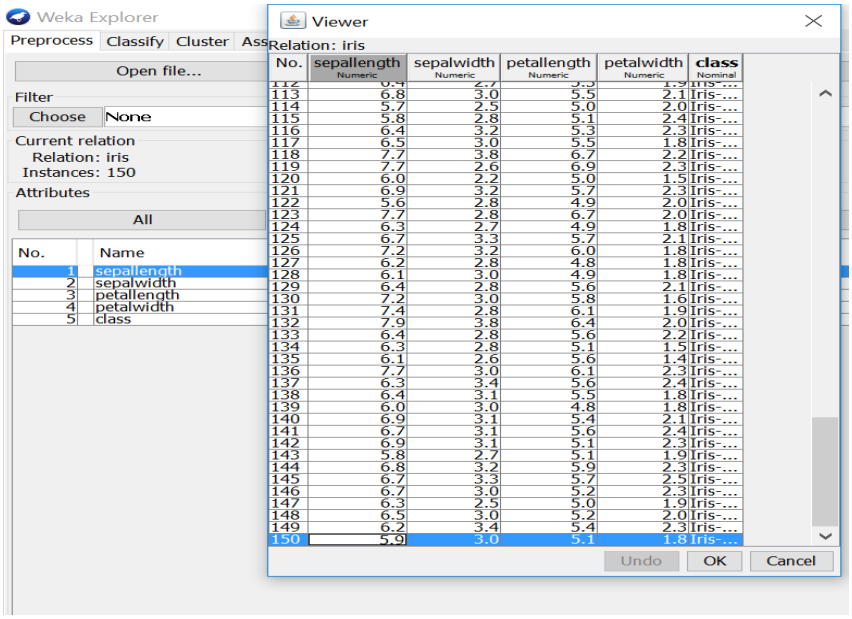
Here we have taken **IRIS.arff** dataset as sample for observing all the below things.

i. List the attribute names and they types

There are 5 attributes& its datatype present in the above loaded dataset (IRIS.arff)
sepallength – Numeric
sepalwidth – Numeric
petallength – Numeric
petallength – Numeric
Class – Nominal

ii. Number of records in each dataset

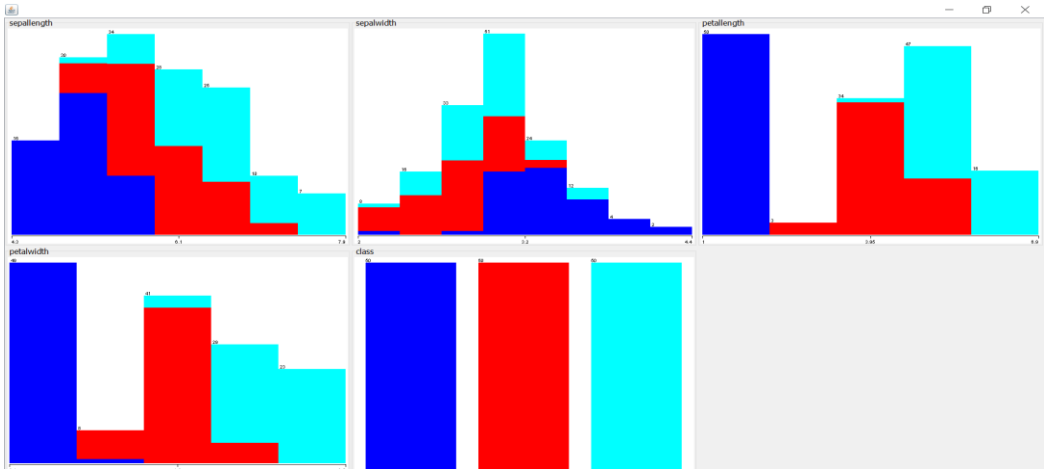
There are total <u>150</u> records (Instances) in dataset (IRIS.arff).



iii. Identify the class attribute (if any)

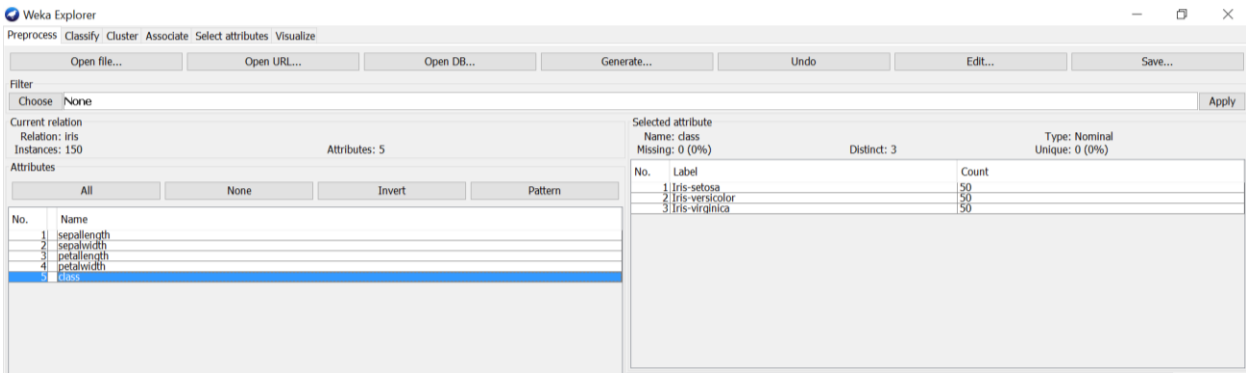
There is one class attribute which consists of 3 labels. They are:
1. Iris-setosa
2. Iris-versicolor
3. Iris-virginica

iv. Plot Histogram

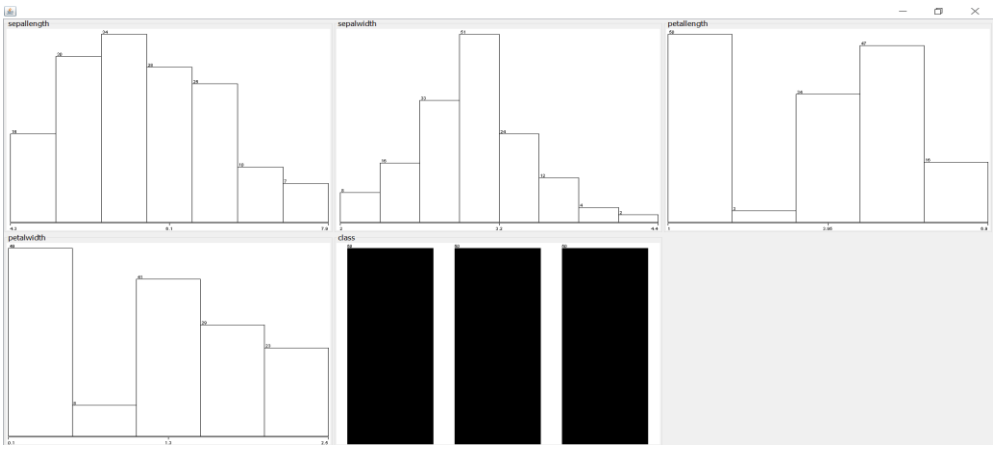


v. Determine the number of records for each class.

There is one class attribute (150 records) which consists of 3 labels. They are shown below
1. Iris-setosa - 50 records
2. Iris-versicolor – 50 records
3. Iris-virginica – 50 records



vi. Visualize the data in various dimensions

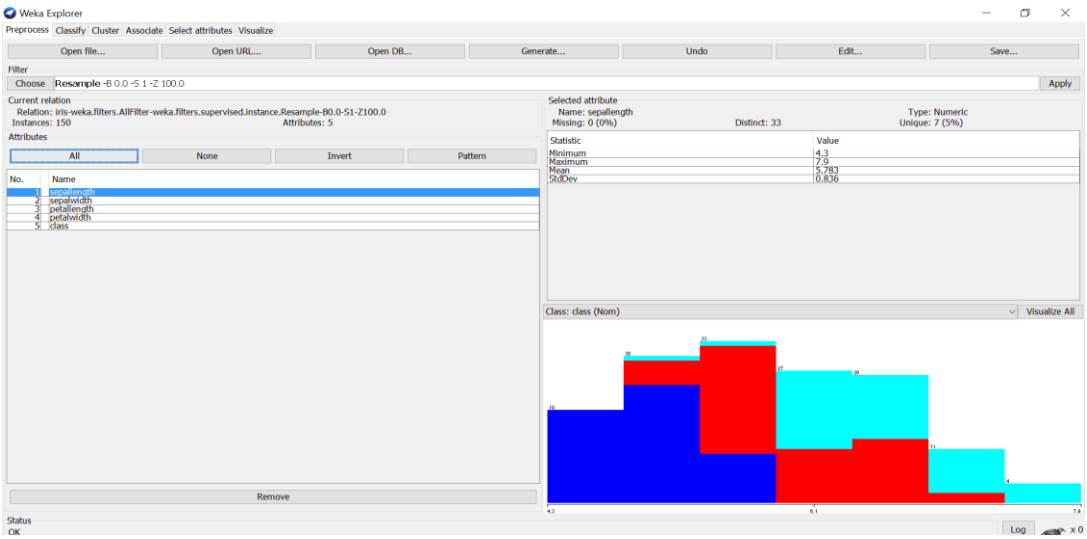


2. Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets

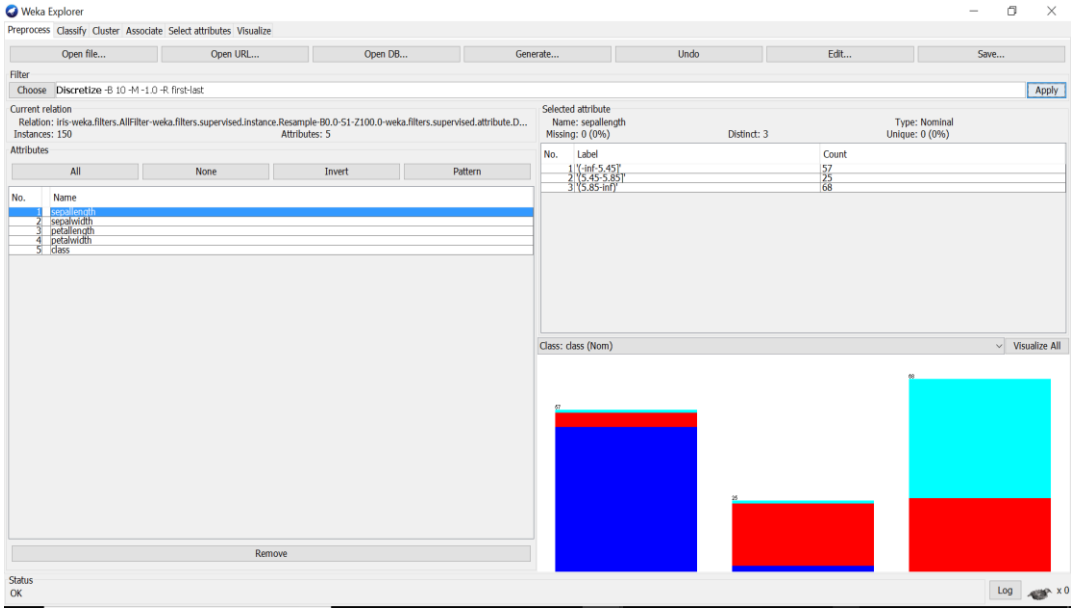
A. Explore various options available in Weka for preprocessing data and apply (like Discretization Filters, Resample filter, etc.) on each dataset

Procedure:

1. For preprocessing the data after selecting the dataset (IRIS.arff).
2. Select Filter option & apply the resample filter & see the below results.



3. Select another filter option & apply the discretization filter, see the below results



Likewise, we can apply different filters for preprocessing the data & see the results in different dimensions.

B. Load each dataset into Weka and run Aprori algorithm with different support and confidence values. Study the rules generated.

Procedure:

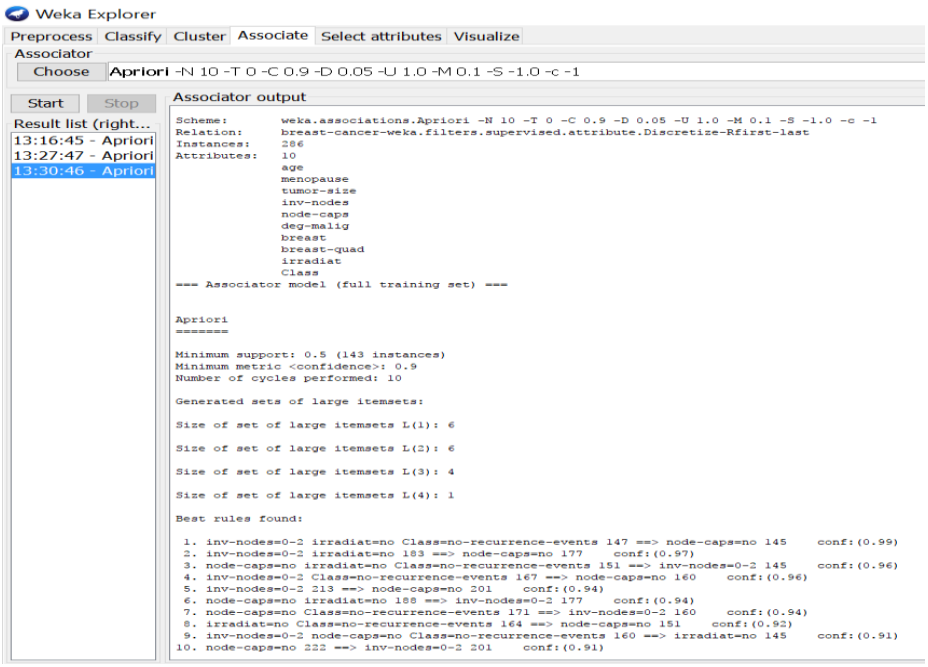
1. Load the dataset (Breast-Cancer.arff) into weka tool
2. Go to associate option & in left-hand navigation bar we can see different association algorithms.
3. In which we can select Aprori algorithm & click on select option.
4. Below we can see the rules generated with different support & confidence values for that selected dataset.

```
Relation:     breast-cancer
Instances:    286
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.5 (143 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 6

Size of set of large itemsets L(3): 4

Size of set of large itemsets L(4): 1

Best rules found:

 1. inv-nodes=0-2 irradiat=no Class=no-recurrence-events 147 ==> node-caps=no 145    conf:(0.99)
 2. inv-nodes=0-2 irradiat=no 183 ==> node-caps=no 177    conf:(0.97)
 3. node-caps=no irradiat=no Class=no-recurrence-events 151 ==> inv-nodes=0-2 145    conf:(0.96)
 4. inv-nodes=0-2 Class=no-recurrence-events 167 ==> node-caps=no 160    conf:(0.96)
 5. inv-nodes=0-2 213 ==> node-caps=no 201    conf:(0.94)
 6. node-caps=no irradiat=no 188 ==> inv-nodes=0-2 177    conf:(0.94)
 7. node-caps=no Class=no-recurrence-events 171 ==> inv-nodes=0-2 160    conf:(0.94)
 8. irradiat=no Class=no-recurrence-events 164 ==> node-caps=no 151    conf:(0.92)
 9. inv-nodes=0-2 node-caps=no Class=no-recurrence-events 160 ==> irradiat=no 145    conf:(0.91)
10. node-caps=no 222 ==> inv-nodes=0-2 201    conf:(0.91)
```

C. Apply different discretization filters on numerical attributes and run the Apriori association rule algorithm. Study the rules generated. Derive interesting insights and observe the effect of discretization in the rule generation process.

Procedure:

1. Load the dataset (Breast-Cancer.arff) into weka tool& select the discretize filter & apply it.
2. Go to associate option & in left-hand navigation bar we can see different association algorithms.
3. In which we can select Aprori algorithm & click on select option.
4. Below we can see the rules generated with different support & confidence values for that selected dataset.

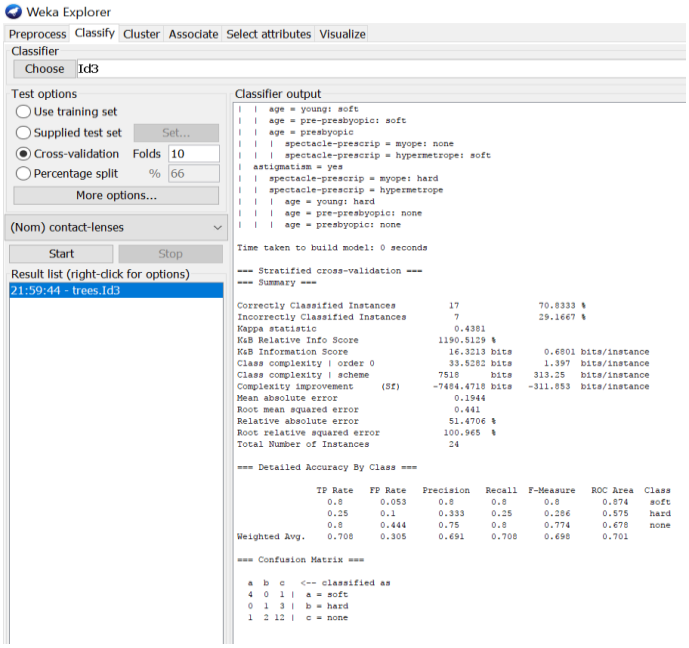


3. Demonstrate performing classification on data sets

A. Load each dataset into Weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.

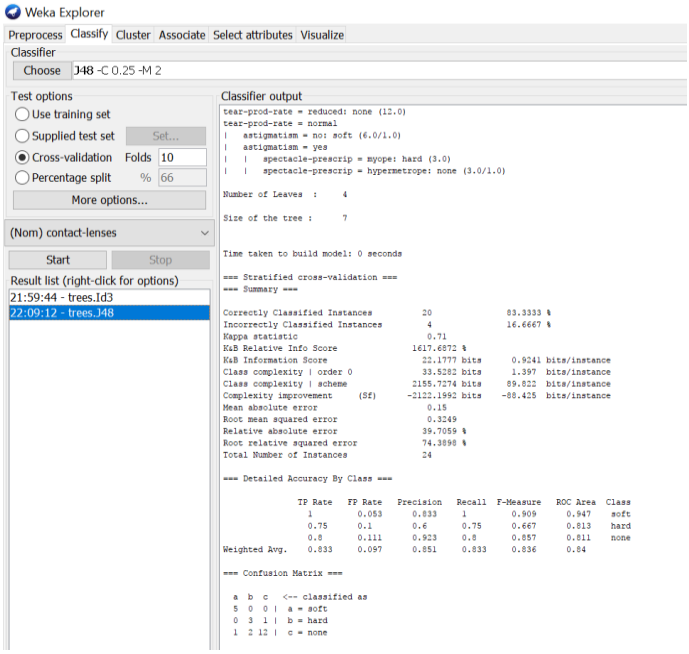
Procedure for Id3:

1. Load the dataset (Contact-lenses.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under tree section.
3. In which we selected Id3 algorithm, in more options select the output entropy evaluation measures& click on start option.
4. Then we will get classifier output, entropy values& Kappa Statistic as represented below.



5. In the above screenshot, we can run classifiers with different test options (Cross-validation, Use Training Set, Percentage Split, Supplied Test set).
The result of applying the chosen classifier will be tested according to the optionsthat are set by clicking in the Test options box. There are four test modes:
A. Use training set: The classifier is evaluated on how well it predicts theclass of the instances it was trained on.
B. Supplied test set: The classifier is evaluated on how well it predicts theclass of a set ofinstances loaded from a file. Clicking the Set... buttonbrings up a dialog allowing you to choose the file to test on.
C. Cross-validation: The classifier is evaluated by cross-validation, usingthe number of folds that are entered in the Folds text field.
D. Percentage split: The classifier is evaluated on how well it predicts acertain percentage of the data which is held out for testing. The amountof data held out depends on the value entered in the % field.

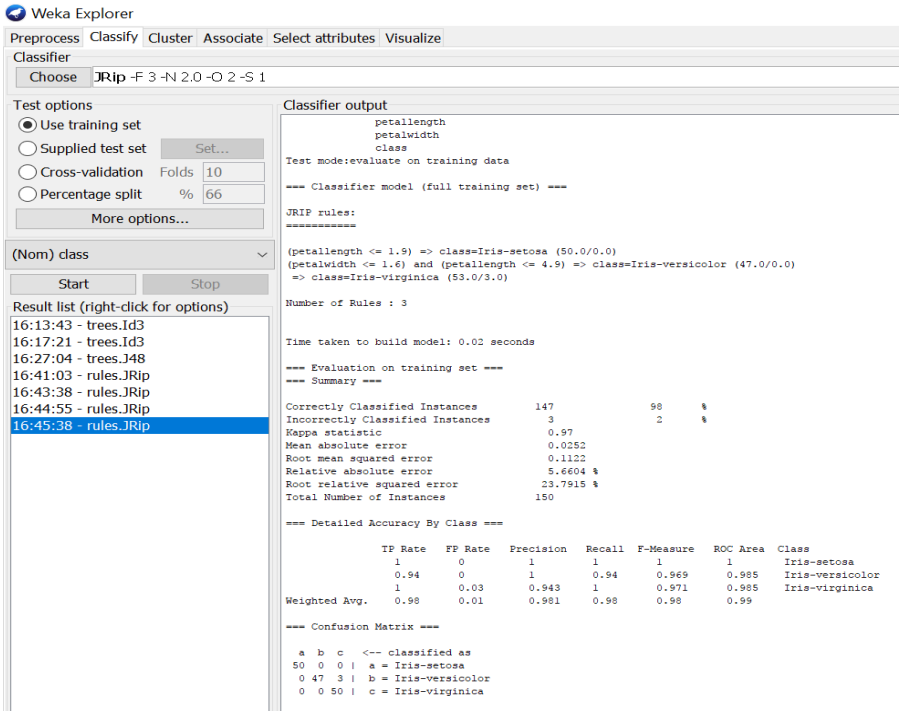
Procedure for J48:
1. Load the dataset (Contact-lenses.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under tree section.
3. In which we selected J48 algorithm,in more options select the output entropy evaluation measures& click on start option.
4. Then we will get classifier output, entropy values & Kappa Statistic as represented below.
5. In the below screenshot, we can run classifiers with different test options (Cross-validation, Use Training Set, Percentage Split, Supplied Test set).



B. Extract if-then rules from the decision tree generated by the classifier, Observethe confusion matrix and derive Accuracy, F-measure, TPrate, FPrate, Precision and Recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results.

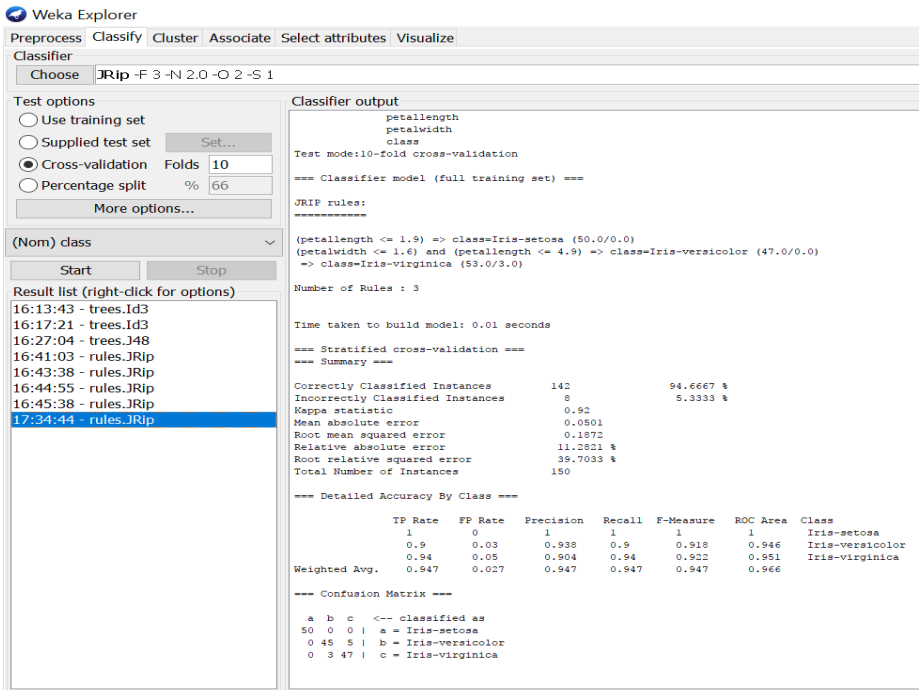
Procedure:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under rules section.
3. In which we selected JRip (If-then) algorithm & click on start option with “use training set” test option enabled.
4. Then we will get detailed accuracy by class consists ofF-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



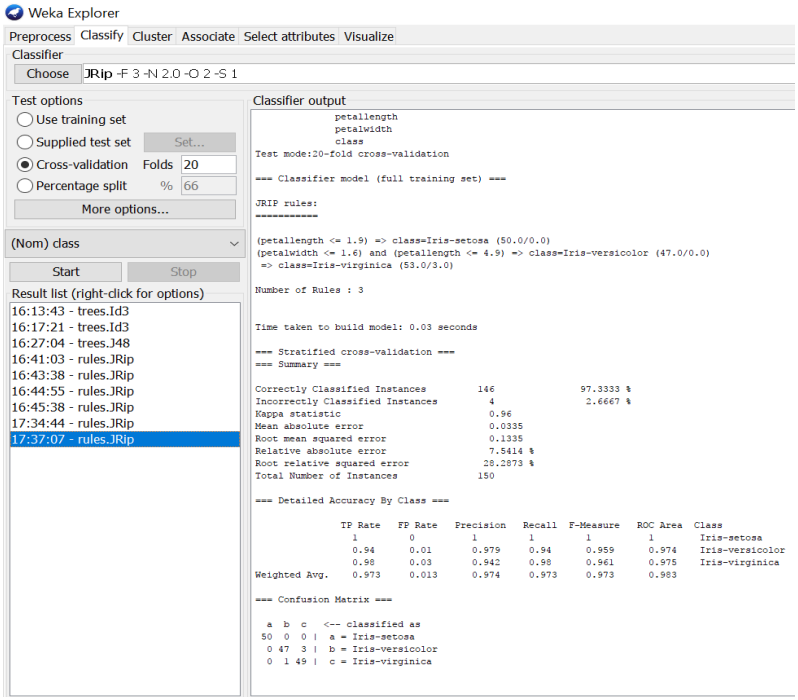
Using Cross-Validation Strategy with 10 folds:

Here, we enabled cross-validation test option with 10 folds & clicked start button as represented below.



Using Cross-Validation Strategy with 20 folds:

Here, we enabled cross-validation test option with 20 folds & clicked start button as represented below.

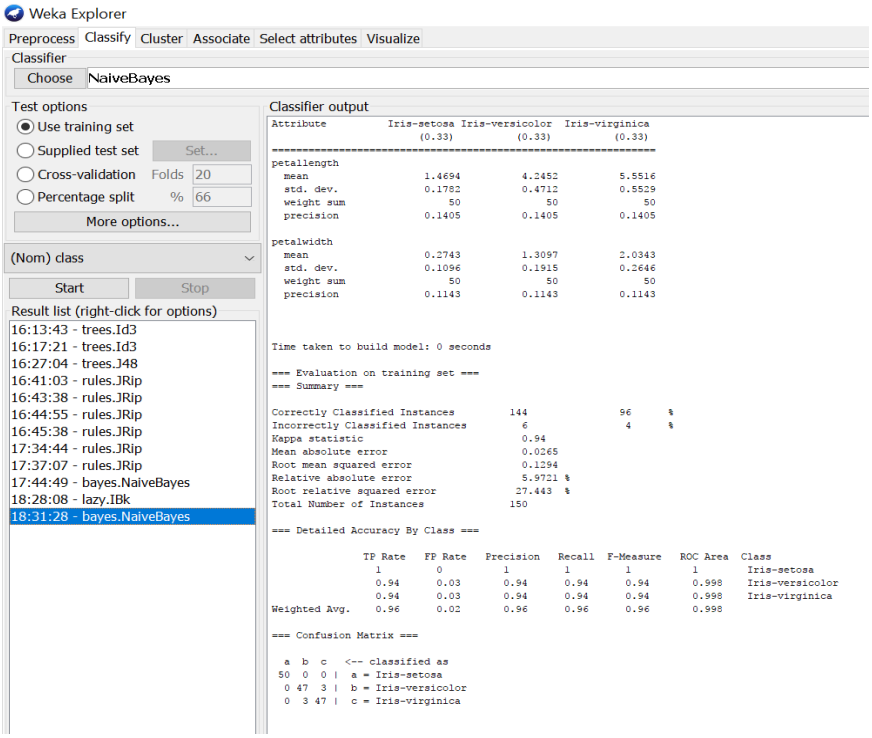


If we see the above results of cross validation with 10 folds & 20 folds. As per our observation the error rate is lesser with 20 folds got 97.3% correctness when compared to 10 folds got 94.6% correctness.

C. Load each dataset into Weka and perform Naive-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.

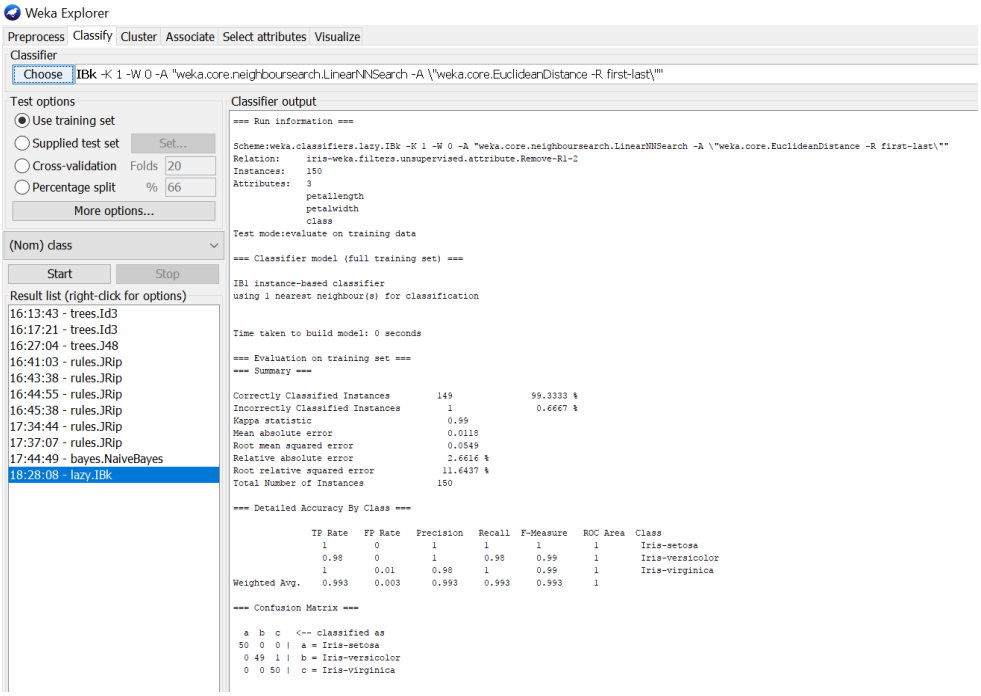
Procedure for Naïve-Bayes:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with “use training set” test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



Procedure for K-Nearest Neighbour (IBK):

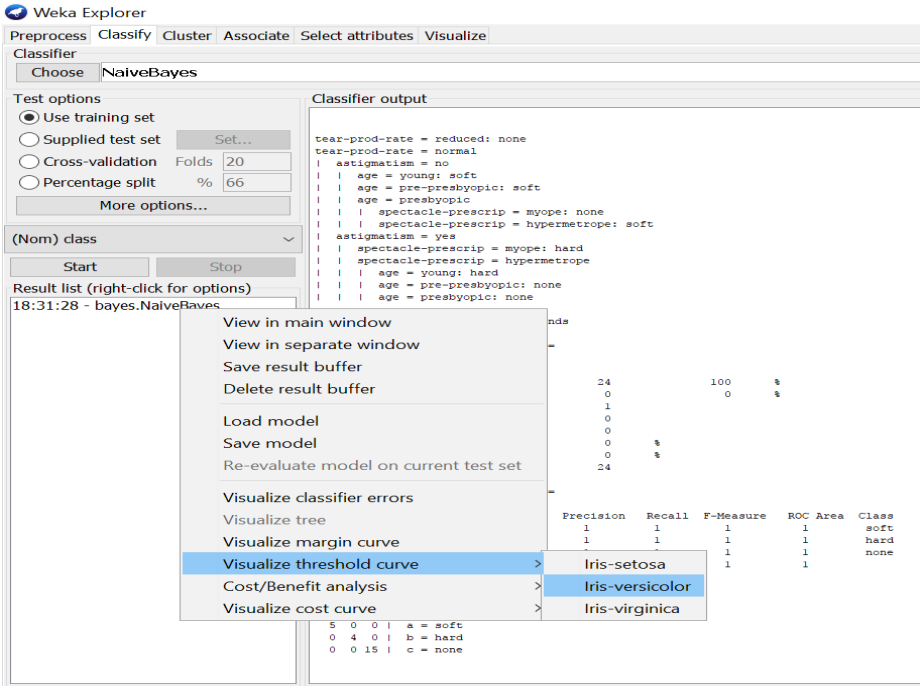
1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under lazy section.
3. In which we selected K-Nearest Neighbour (IBK) algorithm & click on start option with "use training set" test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.

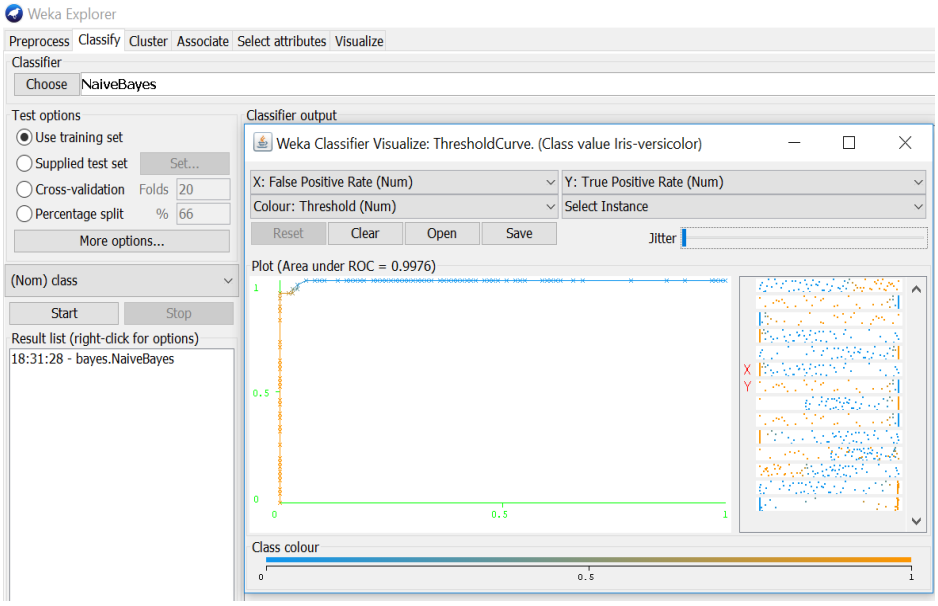


D. Plot RoC Curves

Procedure:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with "use training set" test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix.
5. For plotting RoC Curves, we need to right click on "bayes.NaiveBayes" for getting more options, In which we will select the "Visualize Threshold Curve" & go to any class (Iris-setosa, Iris-versicolor, Iris-Virgincia) as shown in below snapshot.
6. After selecting an class, RoC (Receiver Operating Characteristic) Curve plot will be displayed which has X-Axis –False Positive (FP) rate and Y-Axis – True Positive (TP) rate.

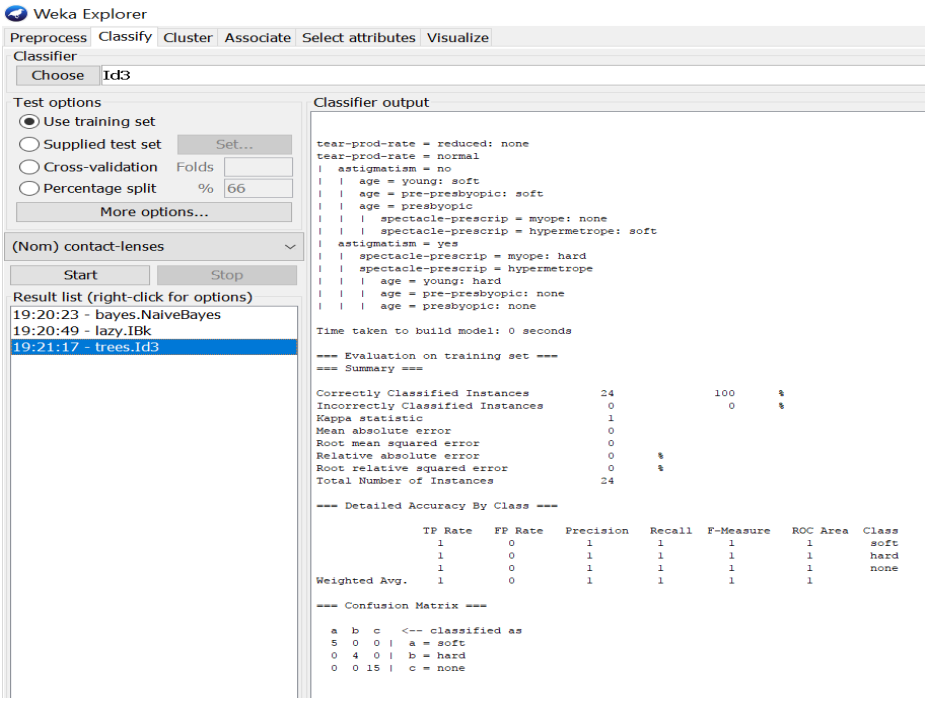

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
20
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
18:31:28 - bayes.NaiveBayes
Classifier output
tear-prod-rate = reduced: none
tear-prod-rate = normal
| astigmatism = no
| | age = young: soft
| | age = pre-presbyopic: soft
| | age = presbyopic
| | | spectacle-prescrip = myope: none
| | | spectacle-prescrip = hypermetrope: soft
| astigmatism = yes
| | spectacle-prescrip = myope: hard
| | spectacle-prescrip = hypermetrope
| | | age = young: hard
| | | age = pre-presbyopic: none
| | | age = presbyopic: none
Precision Recall F-Measure ROC Area Class
soft
hard
none
a = soft
b = hard
c = none
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Iris-setosa
Iris-versicolor
Iris-virginica


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
20
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
18:31:28 - bayes.NaiveBayes
Classifier output
Weka Classifier Visualize: ThresholdCurve. (Class value Iris-versicolor)
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.9976)
Class colour

E. Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

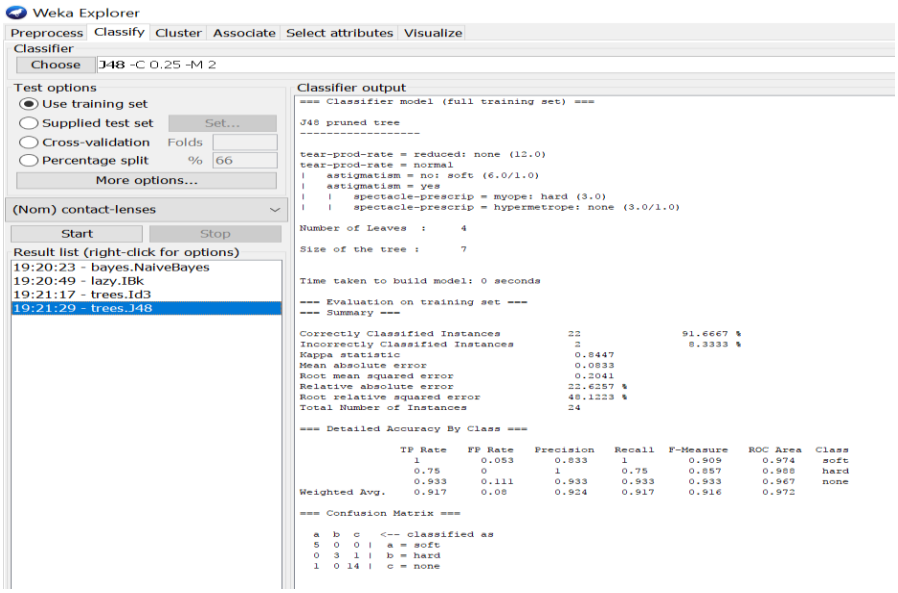
Procedure for ID3:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under trees section.
3. In which we selected ID3 algorithm & click on start option with "use training set" test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



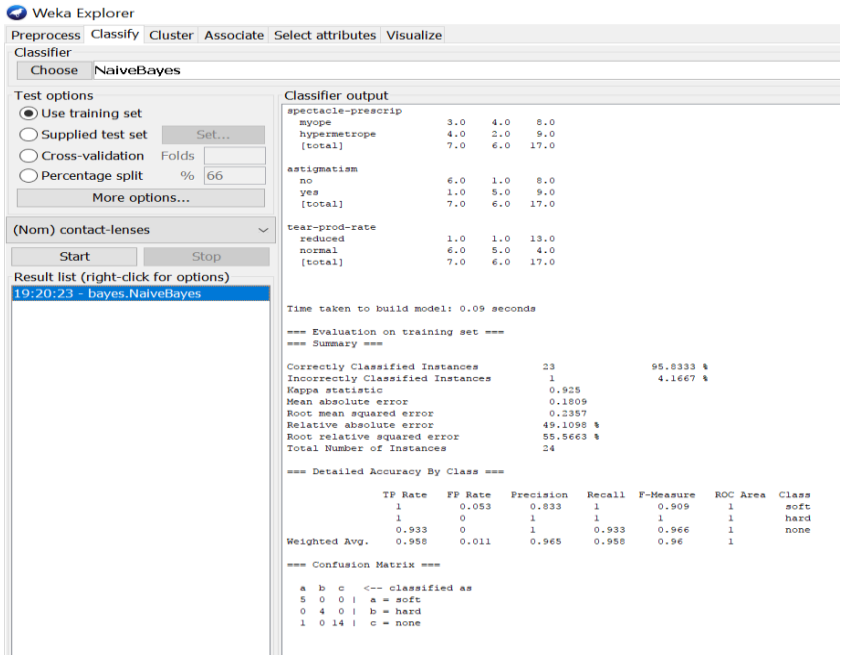
Procedure for J48:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under trees section.
3. In which we selected J48 algorithm & click on start option with "use training set" test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



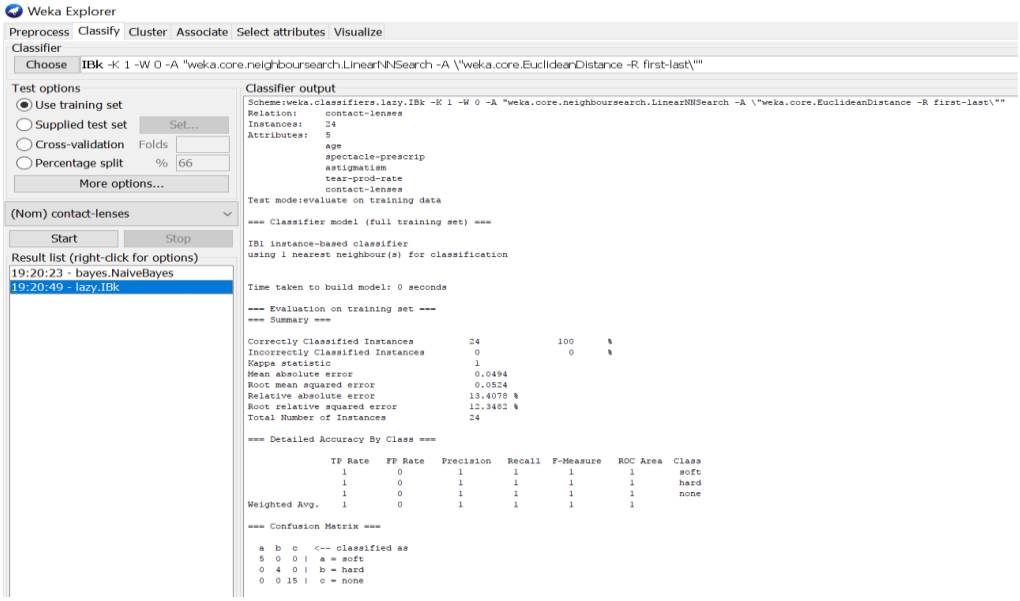
Procedure for Naïve-Bayes:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with "use training set" test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



Procedure for K-Nearest Neighbour (IBK):

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassifcation algorithms under lazy section.
3. In which we selected K-Nearest Neighbour (IBK) algorithm & click on start option with “use training set” test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



By observing all these algorithms (ID3, K-NN, J48 & Naïve Bayes) results, we will conclude that

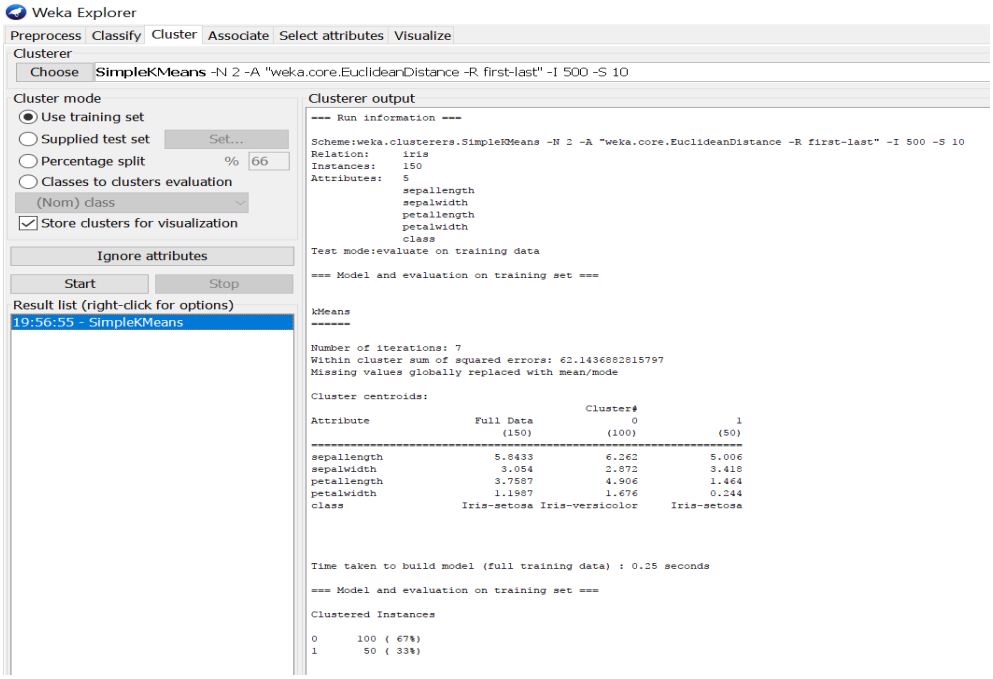
Hence,
ID3 Algorithm’saccuracy & performance is best.
J48 Algorithm’s accuracy & performance is poor.

4. Demonstrate performing clustering on data sets

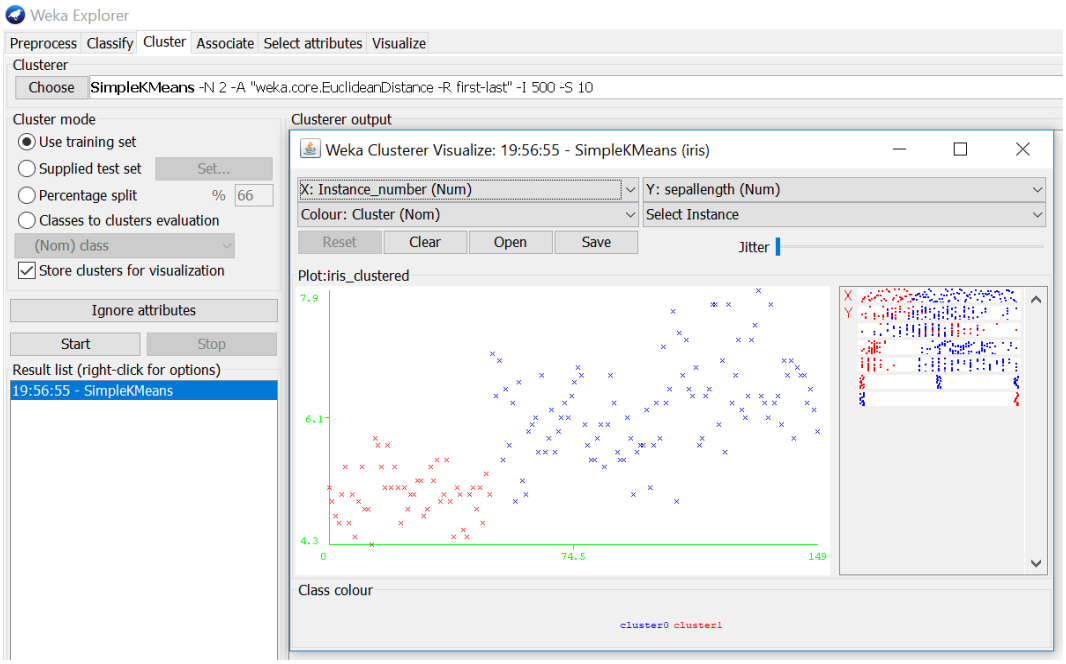
A. Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.

Procedure:

1. Load the dataset (Iris.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclustering algorithms under lazy section.
3. In which we selected Simple K-Means algorithm & click on start option with "use training set" test option enabled.
4. Then we will get the sum of squared errors, centroids, No. of iterations & clustered instances as represented below.

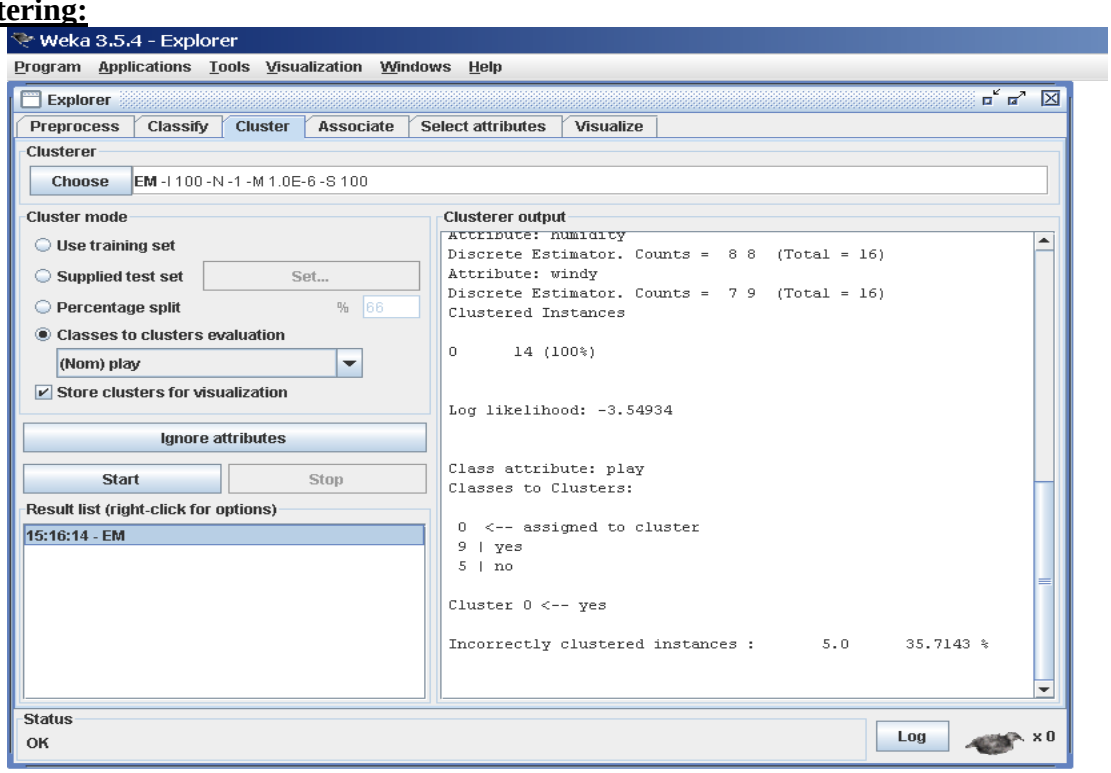


5. If we right click on simple k means, we will get more options in which "Visualize cluster assignments" should be selected for getting cluster visualization as shown below.



B. Explore other clustering techniques available in Weka.

Clustering:



Selecting a Clusterer:

By now you will be familiar with the process of selecting and configuring objects.Clicking on the clustering scheme listed in the Clusterer box at the top of thewindow brings up aGenericObjectEditor dialog with which to choose a newclustering scheme.

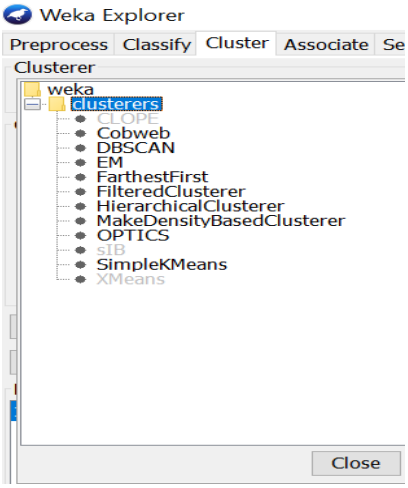
Cluster Modes:

The Cluster mode box is used to choose what to cluster and how to evaluatethe results. The first three options are the same as for classification: Use training set, supplied test set and Percentage split, now the data is assigned to clusters instead of trying to predict a specific class.The fourth mode, Classes to clusters evaluation, compares how well thechosen clusters match up with a pre-assigned class in the data. The drop-downbox below this option selects the class, just as in the Classify panel.An additional option in the Cluster mode box, the Store clusters forvisualization tick box, determines whether or not it will be possible to visualizethe clusters once training is complete. When dealing with datasets that are solarge that memory becomes aproblem it may be helpful to disable this option.

Ignoring Attributes:

Often, some attributes in the data should be ignored when clustering. TheIgnore attributes button brings up a small window that allows you to selectwhich attributes are ignored. Clicking on an attribute in the window highlightsit, holding down the SHIFT key selects a range of consecutive attributes, andholding down CTRL toggles individual attributes on and off. To cancel theselection, back out with the Cancel button. To activate it, click the Selectbutton. The next time clustering is invoked, the selected attributes are ignored.

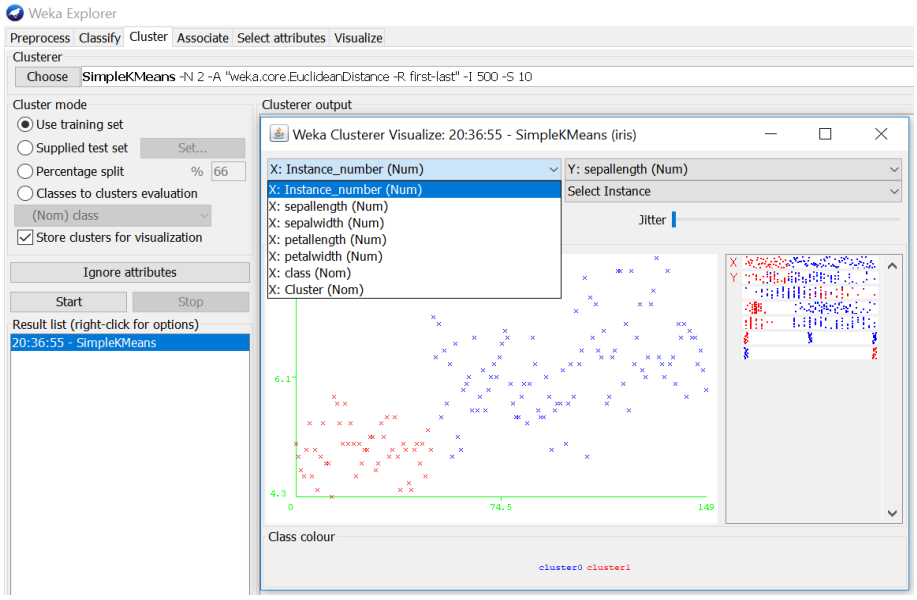
There are 12 clustering algorithms available in weka tool. They are shown below.



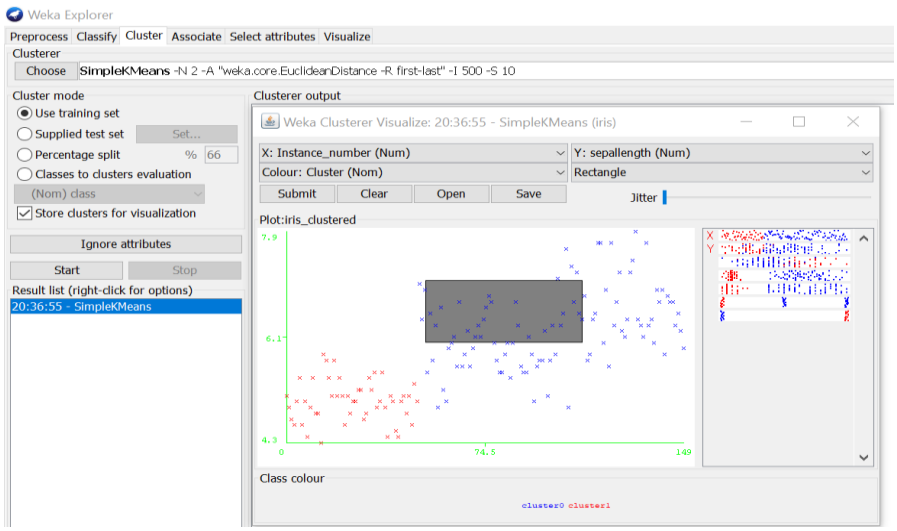
Through visualize cluster assignments, we can clearly see the clusters in graphical visualization.

C. Explore visualization features of Weka to visualize the clusters. Derive interesting insights and explain.

- If we right click on simple k means, we will get more options in which "Visualize cluster assignments" should be selected for getting cluster visualization as shown below.
- In that cluster visualization we are having different features to explore by changing the X-axis, Y-axis, Color, Jitter& Select instance (Rectangle, Polygon & Polyline) for getting different sets of cluster outputs.



- As shown in above screenshot, all the dataset (Iris.arff) tuples are represented in X-axis & in similar way it will represented for y-axis also. For each cluster, the color will be different. In the above figure, there are two clusters which are represented in blue & red colors.
- In the select instance we can select different shapes for choosing clustered area as shown in below screenshot, rectangle shape is selected.



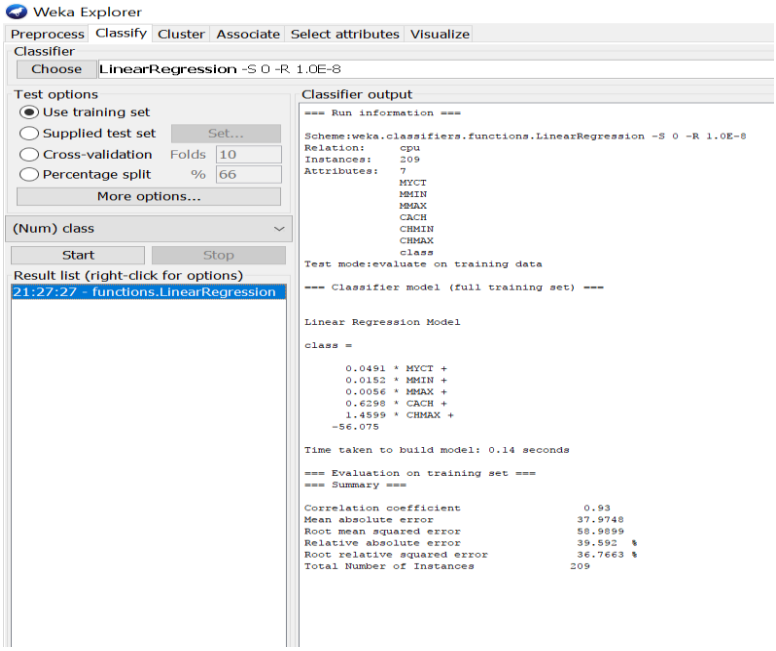
- By this visualization feature we can observe different clustering outputs for an dataset by changing those X-axis, Y-axis, Color & Jitter options.

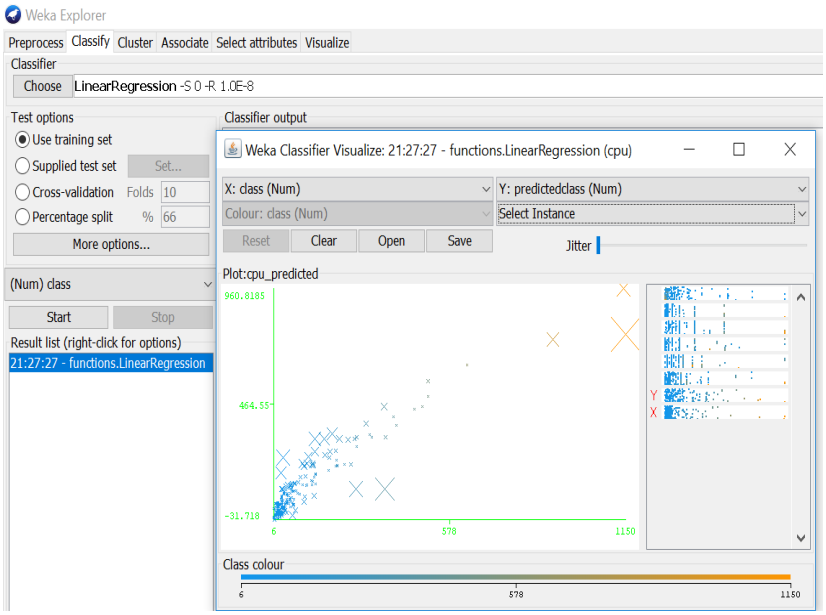
5. Demonstrate performing Regression on data sets

A. Load each dataset into Weka and build Linear Regression model. Study the clusters formed. Use Training set option. Interpret the regression model and derive patterns and conclusions from the regression results.

Procedure:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with use training set option.
4. Then we will get regression model & its result as shown below.
5. The patterns are visually mentioned below for regression model through visualize classifier errors option which is available in right click options.

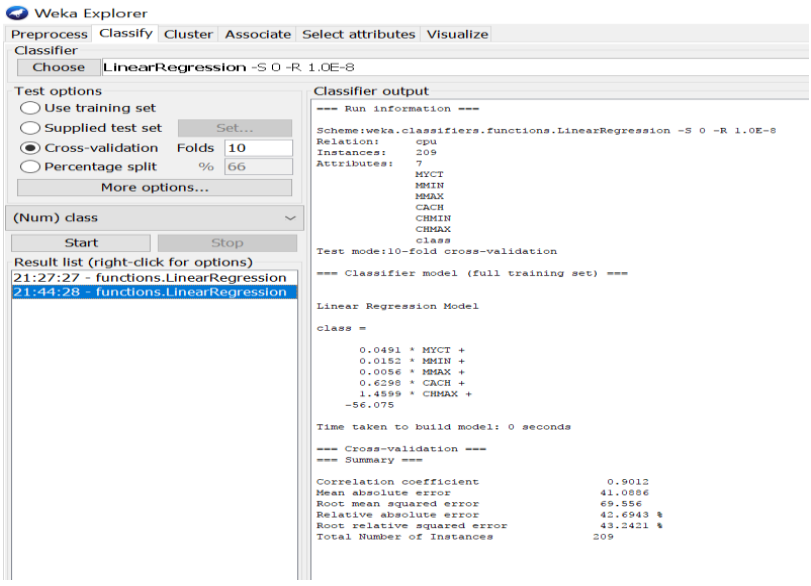




B. Use options cross-validation and percentage split and repeat running the Linear Regression Model. Observe the results and derive meaningful results.

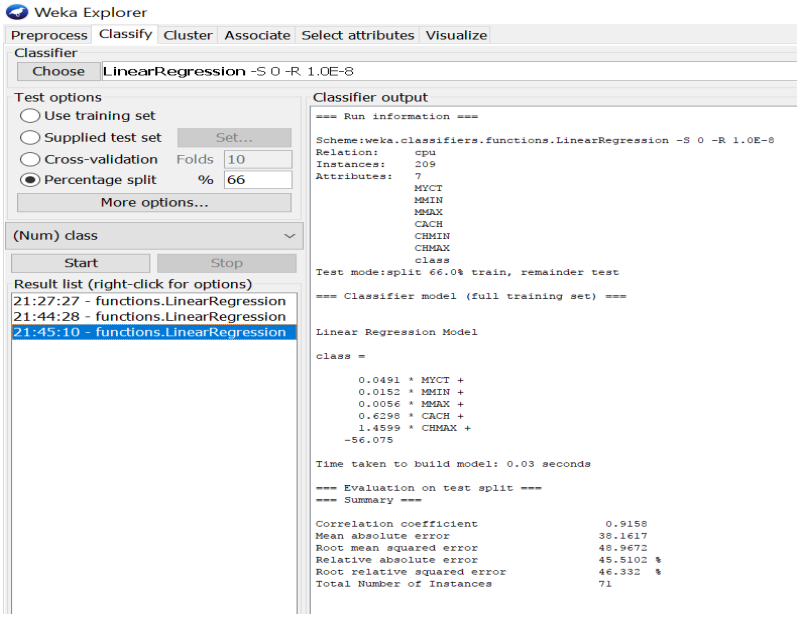
Procedure for cross-validation:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with cross validation option with 10 folds.
4. Then we will get regression model & its result as shown below.



Procedure for percentage split:

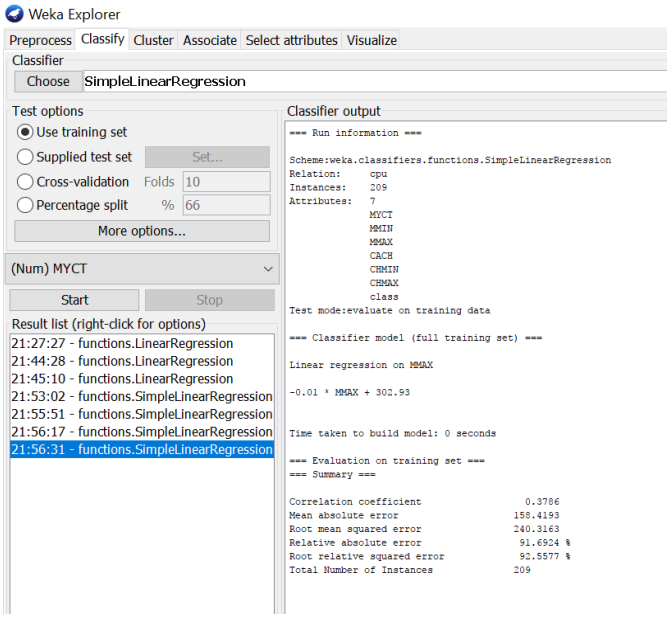
1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with percentage split option with 66% split.
4. Then we will get regression model & its result as shown below.



C. Explore Simple linear regression technique that only looks at one variable

Procedure:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Simple Linear Regression algorithm & click on start option with use training set option with one variable (MYCT).
4. Then we will get regression model & its result as shown below.



Data Mining

Credit Risk Assesment – The German Credit Data

Task 1:- list all the categorical (or nominal) attributes and the real-valued attributes separately.

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) You can see the list of categorical and nominal attributes can be dispayed.

Output:-

Title: German Credit data

Number of Instances: 1000

The categorical attributes are 13 that are as follows.

For algorithms that need numerical attributes, Strathclyde University produced the file "german.data-numeric". This file has been edited and several indicator variables added to make it suitable for algorithms which cannot cope with categorical variables.Severalattributes that are ordered categorical (such as attribute 17) havebeen coded as integer. This was the form used by StatLog.

6. Number of Attributes german: 20 (7 numerical, 13 categorical)
Number of Attributes german.numer: 24 (24 numerical)

7. Attribute description for german

Attribute 1: (qualitative)
Status of existing checking account
A11 : ... < 0 DM
A12 : 0 <= ... < 200 DM
A13 : ... >= 200 DM /
salary assignments for at least 1 year
A14 : no checking account

Attribute 2: (numerical)
Duration in month

Attribute 3: (qualitative)
Credit history
A30 : no credits taken/
all credits paid back duly
A31 : all credits at this bank paid back duly
A32 : existing credits paid back duly till now
A33 : delay in paying off in the past
A34 : critical account/
other credits existing (not at this bank)

Attribute 4: (qualitative)
Purpose
A40 : car (new)
A41 : car (used)
A42 : furniture/equipment
A43 : radio/television
A44 : domestic appliances
A45 : repairs
A46 : education
A47 : (vacation - does not exist?)
A48 : retraining
A49 : business

A410 : others

Attribute 5: (numerical)
Credit amount

Attibute 6: (qualitative)
Savings account/bonds
A61 : ... < 100 DM
A62 : 100 <= ... < 500 DM
A63 : 500 <= ... < 1000 DM
A64 : .. >= 1000 DM
A65 : unknown/ no savings account

Attribute 7: (qualitative)
Present employment since
A71 : unemployed
A72 : ... < 1 year
A73 : 1 <= ... < 4 years
A74 : 4 <= ... < 7 years
A75 : .. >= 7 years

Attribute 8: (numerical)
Installment rate in percentage of disposable income

Attribute 9: (qualitative)
Personal status and sex
A91 : male : divorced/separated
A92 : female : divorced/separated/married
A93 : male : single
A94 : male : married/widowed
A95 : female : single

Attribute 10: (qualitative)
Other debtors / guarantors
A101 : none
A102 : co-applicant
A103 : guarantor

Attribute 11: (numerical)
Present residence since

Attribute 12: (qualitative)
Property
A121 : real estate
A122 : if not A121 : building society savings agreement/
life insurance

A123 : if not A121/A122 : car or other, not in attribute 6
A124 : unknown / no property

Attribute 13: (numerical)
Age in years

Attribute 14: (qualitative)
Other installment plans
A141 : bank
A142 : stores
A143 : none

Attribute 15: (qualitative)
Housing
A151 : rent
A152 : own
A153 : for free

Attribute 16: (numerical)
Number of existing credits at this bank

Attribute 17: (qualitative)
Job
A171 : unemployed/ unskilled - non-resident
A172 : unskilled - resident
A173 : skilled employee / official
A174 : management/ self-employed/
highly qualified employee/ officer

Attribute 18: (numerical)
Number of people being liable to provide maintenance for

Attribute 19: (qualitative)
Telephone
A191 : none
A192 : yes, registered under the customers name

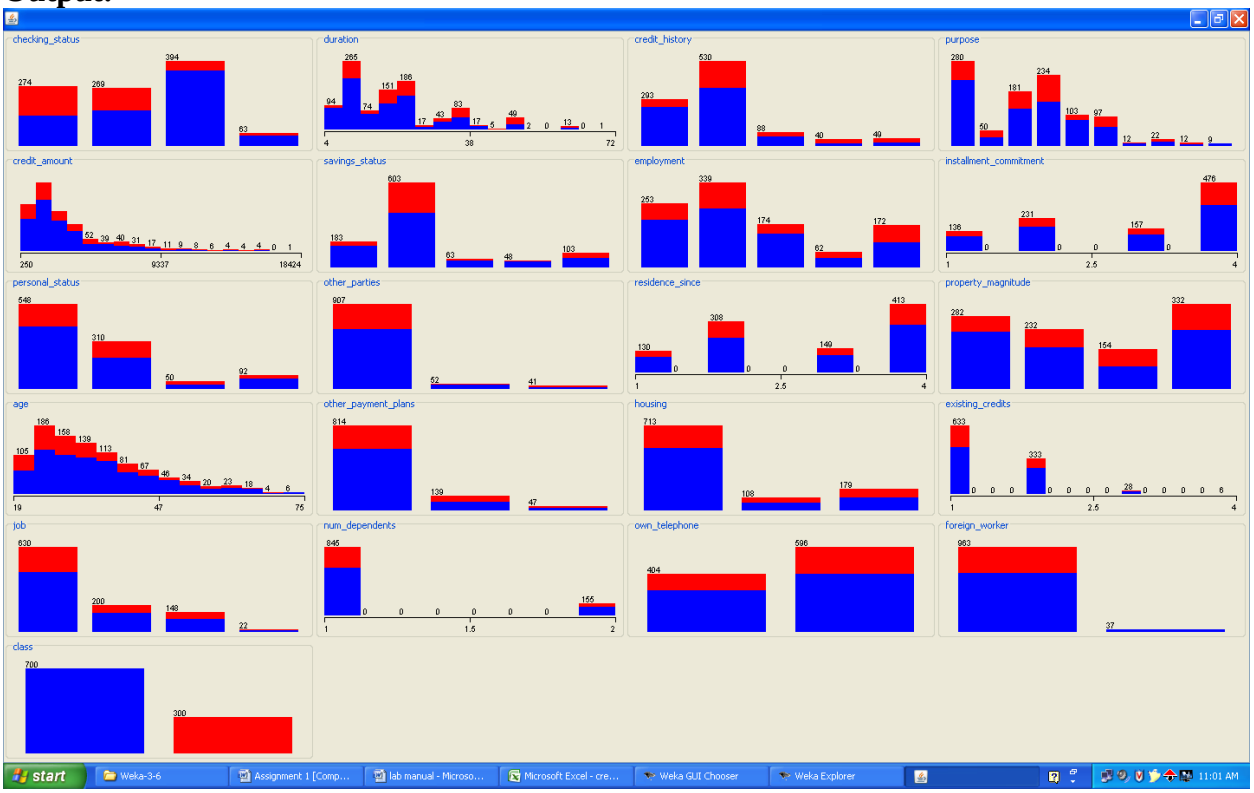
Attribute 20: (qualitative)
foreign worker
A201 : yes
A202 : no

Task 2:- what attributes do you think might be crucial in making the credit assessment ? come up with some simple rules in plain English using your selected attributes

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Choose the credit history has the crucial attributes are 293.

Output:-



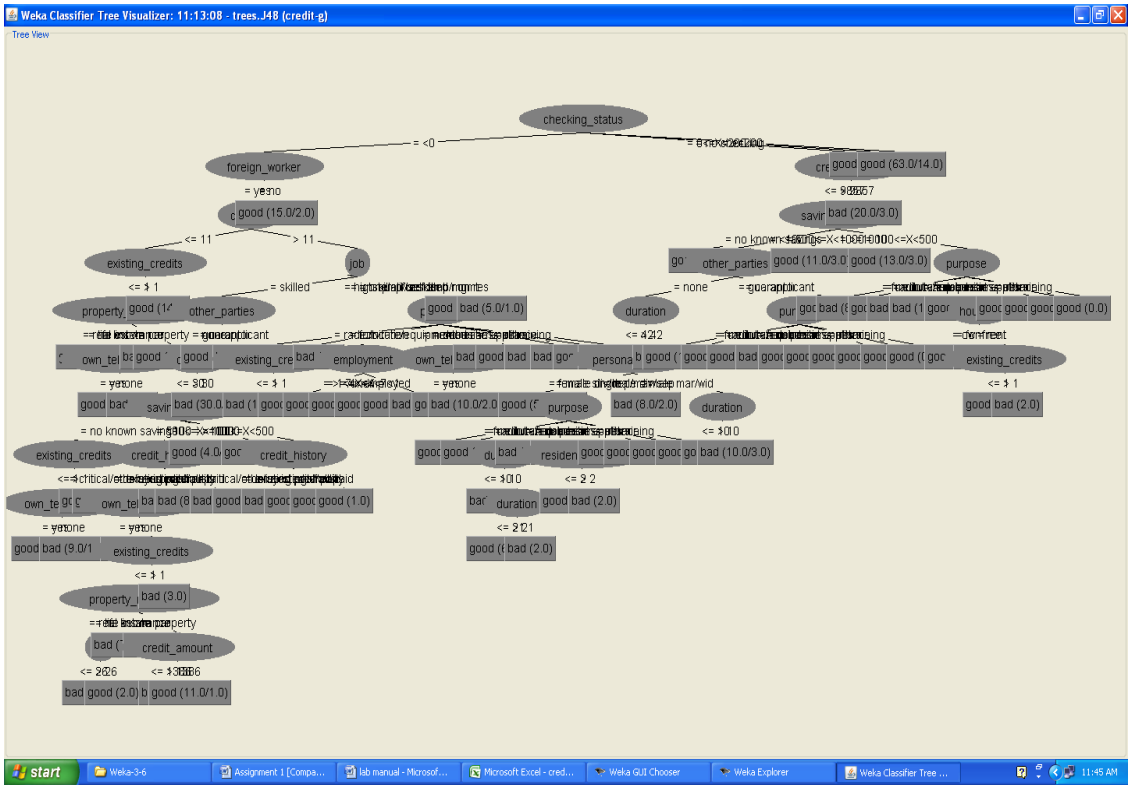
Task 3:- one type of model that you can create is a Decision tree – train a Decision tree using the complete dataset as the training data. Report the model obtained after training.

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Next click the "classify" menu on the top.
7) Click the "choose" and list of "trees and rules" can be displayed.
8) Select the "trees" under click the "J48". And click "start".

9) Result list can be displayed and right click on “J48” and choose the and select the “visualize tree”. See the decision tree.

Output:-



Task 4:-

Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly? (this is also called testing on the training set) why do you think you cannot get 100% training accuracy?

Procedure:-

1) Insert the data into the excel sheet and save the file is “.CSV”.
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are ”Explorer, Experimenter, Knowledge flow, simple CLI”.
4) Click the “Explorer” button.
5) A new window is open and click the “preprocess” tab and open a “german.arff ” file.
6) Next click the “classify” menu on the top.

7) Click the “choose” and list of “trees and rules” can be displayed.
8) Select the “trees” under click the “J48”. And choose the radio button is “using training dataset”. And click “start”. See the correctly classified instances.

Output:-

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     credit-g-weka.filters.unsupervised.attribute.Remove-R3
Instances:    1000
Attributes:   20
              checking_status
              duration
              purpose
              credit_amount
              savings_status
              employment
              installment_commitment
              personal_status
              other_parties
              residence_since
              property_magnitude
              age
              other_payment_plans
              housing
              existing_credits
              job
              num_dependents
              own_telephone
              foreign_worker
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
------------------

checking_status = <0
|   foreign_worker = yes
|   |   duration <= 11
|   |   |   existing_credits <= 1
|   |   |   |   property_magnitude = real estate: good (8.0/1.0)
|   |   |   |   property_magnitude = life insurance
|   |   |   |   |   own_telephone = yes: good (4.0)
|   |   |   |   |   own_telephone = none: bad (2.0)
```

```
|  |  |  |  property_magnitude = no known property: bad (3.0)
|  |  |  |  property_magnitude = car: good (2.0/1.0)
|  |  |  existing_credits > 1: good (14.0)
|  |  duration > 11
|  |  |  job = skilled
|  |  |  |  other_parties = none
|  |  |  |  |  savings_status = no known savings
|  |  |  |  |  |  existing_credits <= 1
|  |  |  |  |  |  |  own_telephone = yes: good (4.0/1.0)
|  |  |  |  |  |  |  own_telephone = none: bad (10.0/1.0)
|  |  |  |  |  |  existing_credits > 1: good (2.0)
|  |  |  |  |  savings_status = <100: bad (98.0/30.0)
|  |  |  |  |  savings_status = 500<=X<1000: good (5.0/2.0)
|  |  |  |  |  savings_status = >=1000: good (4.0)
|  |  |  |  |  savings_status = 100<=X<500
|  |  |  |  |  |  property_magnitude = real estate: good (1.0)
|  |  |  |  |  |  property_magnitude = life insurance: bad (3.0)
|  |  |  |  |  |  property_magnitude = no known property: good (0.0)
|  |  |  |  |  |  property_magnitude = car: good (2.0)
|  |  |  |  other_parties = guarantor
|  |  |  |  |  duration <= 45: good (10.0/1.0)
|  |  |  |  |  duration > 45: bad (2.0)
|  |  |  |  other_parties = co applicant: bad (7.0/1.0)
|  |  |  job = unskilled resident
|  |  |  |  purpose = radio/tv
|  |  |  |  |  existing_credits <= 1: bad (10.0/3.0)
|  |  |  |  |  existing_credits > 1: good (2.0)
|  |  |  |  purpose = education: bad (1.0)
|  |  |  |  purpose = furniture/equipment
|  |  |  |  |  employment = >=7: good (2.0)
|  |  |  |  |  employment = 1<=X<4: good (4.0)
|  |  |  |  |  employment = 4<=X<7: good (1.0)
|  |  |  |  |  employment = unemployed: good (0.0)
|  |  |  |  |  employment = <1: bad (3.0)
|  |  |  |  purpose = new car
|  |  |  |  |  own_telephone = yes: good (2.0)
|  |  |  |  |  own_telephone = none: bad (10.0/2.0)
|  |  |  |  purpose = used car: bad (1.0)
|  |  |  |  purpose = business: good (3.0)
|  |  |  |  purpose = domestic appliance: bad (1.0)
|  |  |  |  purpose = repairs: bad (1.0)
|  |  |  |  purpose = other: good (1.0)
|  |  |  |  purpose = retraining: good (1.0)
|  |  |  job = high qualif/self emp/mgmt: good (30.0/8.0)
|  |  |  job = unemp/unskilled non res: bad (5.0/1.0)
|  foreign_worker = no: good (15.0/2.0)
```

```
checking_status = 0<=X<200
|  credit_amount <= 9857
|  |  savings_status = no known savings: good (41.0/5.0)
|  |  savings_status = <100
|  |  |  other_parties = none
|  |  |  |  duration <= 42
|  |  |  |  |  personal_status = male single: good (52.0/15.0)
|  |  |  |  |  personal_status = female div/dep/mar
|  |  |  |  |  |  purpose = radio/tv: good (8.0/2.0)
|  |  |  |  |  |  purpose = education: good (4.0/2.0)
|  |  |  |  |  |  purpose = furniture/equipment
|  |  |  |  |  |  |  duration <= 10: bad (3.0)
|  |  |  |  |  |  |  duration > 10
|  |  |  |  |  |  |  |  duration <= 21: good (6.0/1.0)
|  |  |  |  |  |  |  |  duration > 21: bad (2.0)
|  |  |  |  |  |  purpose = new car: bad (5.0/1.0)
|  |  |  |  |  |  purpose = used car: bad (1.0)
|  |  |  |  |  |  purpose = business
|  |  |  |  |  |  |  residence_since <= 2: good (3.0)
|  |  |  |  |  |  |  residence_since > 2: bad (2.0)
|  |  |  |  |  |  purpose = domestic appliance: good (0.0)
|  |  |  |  |  |  purpose = repairs: good (1.0)
|  |  |  |  |  |  purpose = other: good (0.0)
|  |  |  |  |  |  purpose = retraining: good (0.0)
|  |  |  |  |  personal_status = male div/sep: bad (8.0/2.0)
|  |  |  |  |  personal_status = male mar/wid
|  |  |  |  |  |  duration <= 10: good (6.0)
|  |  |  |  |  |  duration > 10: bad (10.0/3.0)
|  |  |  |  duration > 42: bad (7.0)
|  |  |  other_parties = guarantor
|  |  |  |  purpose = radio/tv: good (18.0/1.0)
|  |  |  |  purpose = education: good (0.0)
|  |  |  |  purpose = furniture/equipment: good (0.0)
|  |  |  |  purpose = new car: bad (2.0)
|  |  |  |  purpose = used car: good (0.0)
|  |  |  |  purpose = business: good (0.0)
|  |  |  |  purpose = domestic appliance: good (0.0)
|  |  |  |  purpose = repairs: good (0.0)
|  |  |  |  purpose = other: good (0.0)
|  |  |  |  purpose = retraining: good (0.0)
|  |  |  other_parties = co applicant: good (2.0)
|  |  savings_status = 500<=X<1000: good (11.0/3.0)
|  |  savings_status = >=1000
|  |  |  duration <= 10: bad (2.0)
|  |  |  duration > 10: good (11.0/1.0)
|  |  savings_status = 100<=X<500
```

```
|   |   |   purpose = radio/tv: bad (8.0/2.0)
|   |   |   purpose = education: good (0.0)
|   |   |   purpose = furniture/equipment: bad (4.0/1.0)
|   |   |   purpose = new car: bad (15.0/5.0)
|   |   |   purpose = used car: good (3.0)
|   |   |   purpose = business
|   |   |   |   housing = own: good (6.0)
|   |   |   |   housing = for free: bad (1.0)
|   |   |   |   housing = rent
|   |   |   |   |   existing_credits <= 1: good (2.0)
|   |   |   |   |   existing_credits > 1: bad (2.0)
|   |   |   purpose = domestic appliance: good (0.0)
|   |   |   purpose = repairs: good (2.0)
|   |   |   purpose = other: good (1.0)
|   |   |   purpose = retraining: good (0.0)
|   credit_amount > 9857: bad (20.0/3.0)
checking_status = no checking: good (394.0/46.0)
checking_status = >=200: good (63.0/14.0)

Number of Leaves  :   87

Size of the tree :        119


Time taken to build model: 0.02 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         839               83.9    %
Incorrectly Classified Instances       161               16.1    %
Kappa statistic                          0.5971
Mean absolute error                      0.2508
Root mean squared error                  0.3541
Relative absolute error                 59.6831 %
Root relative squared error             77.2695 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.921    0.353      0.859     0.921     0.889      0.848    good
                 0.647    0.079      0.779     0.647     0.707      0.848    bad
Weighted Avg.    0.839     0.271      0.835     0.839     0.834      0.848

=== Confusion Matrix ===
```

```
  a   b   <-- classified as
645  55 |   a = good
106 194 |   b = bad
```

Task 5:-

Is testing on the training set as you did above a good idea? Why or why not?

Description:-

- Performance on the training set is definitely not a good indicator of performance on an independent test data.
- For classification problem it is naturally to measures a classifiers performance in terms of the error rate. The classifier predicts the class of each instance if it is correct that is counted as a success if not it is an error.
- The error rate on the training data is called the " resubstitution error", because it is calculated by resubstituting the training instances into a classifier that was constructed for them.
- None of the data may be used to determine an estimate of the future error rate. In such situations uses the training data, validation data, and test data.
- The training data is used by one or more learning methods to come up with classifiers.
- Of course, what are interested in is the likely future performance on new data, not the past performance on old data. We already know the classification of each instance in the training set, which after all is why we can use it for training.
- Data are randomly partitioned into two independent sets, a training set and test set.
- Typically two-third of the data are allocated to the training set, and the remaining one-third allocated to the test set.

Task 6:-

One approach for solving the problem encountered in the previous questions is using cross-validation? Describe what is cross-validation briefly. Train a decision tree again using cross-validation and report your results. Does your accuracy increase or decrease? Why?

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Next click the "classify" menu on the top.
7) Click the "choose" and list of "trees and rules" can be displayed.
8) Select the "trees" under click the "J48". And check the radio button is cross-validation. And click "start". See the correctly classified instances.

In cross-validation you decided on a fixed number of folds or partitions of the data. Two-third for training and one-third for testing and repeat procedure three times so that in the end, every instance has been used exactly once for testing is called stratified cross-fold validation.In training dataset compared with cross-validation the accuracy is decreases.

Output:-

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: credit-g-weka.filters.unsupervised.attribute.Remove-R3
Instances: 1000
Attributes: 20
checking_status
duration
purpose
credit_amount
savings_status
employment
installment_commitment
personal_status
other_parties
residence_since
property_magnitude
age
other_payment_plans
housing
existing_credits
job
num_dependents
own_telephone
foreign_worker
class
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

```
checking_status = <0
|  foreign_worker = yes
|  |  duration <= 11
|  |  |  existing_credits <= 1
|  |  |  |  property_magnitude = real estate: good (8.0/1.0)
|  |  |  |  property_magnitude = life insurance
|  |  |  |  |  own_telephone = yes: good (4.0)
|  |  |  |  |  own_telephone = none: bad (2.0)
|  |  |  |  property_magnitude = no known property: bad (3.0)
|  |  |  |  property_magnitude = car: good (2.0/1.0)
|  |  |  existing_credits > 1: good (14.0)
|  |  duration > 11
|  |  |  job = skilled
|  |  |  |  other_parties = none
|  |  |  |  |  savings_status = no known savings
|  |  |  |  |  |  existing_credits <= 1
|  |  |  |  |  |  |  own_telephone = yes: good (4.0/1.0)
|  |  |  |  |  |  |  own_telephone = none: bad (10.0/1.0)
|  |  |  |  |  |  existing_credits > 1: good (2.0)
|  |  |  |  |  savings_status = <100: bad (98.0/30.0)
|  |  |  |  |  savings_status = 500<=X<1000: good (5.0/2.0)
|  |  |  |  |  savings_status = >=1000: good (4.0)
|  |  |  |  |  savings_status = 100<=X<500
|  |  |  |  |  |  property_magnitude = real estate: good (1.0)
|  |  |  |  |  |  property_magnitude = life insurance: bad (3.0)
|  |  |  |  |  |  property_magnitude = no known property: good (0.0)
|  |  |  |  |  |  property_magnitude = car: good (2.0)
|  |  |  |  other_parties = guarantor
|  |  |  |  |  duration <= 45: good (10.0/1.0)
|  |  |  |  |  duration > 45: bad (2.0)
|  |  |  |  other_parties = co applicant: bad (7.0/1.0)
|  |  |  job = unskilled resident
|  |  |  |  purpose = radio/tv
|  |  |  |  |  existing_credits <= 1: bad (10.0/3.0)
|  |  |  |  |  existing_credits > 1: good (2.0)
|  |  |  |  purpose = education: bad (1.0)
|  |  |  |  purpose = furniture/equipment
|  |  |  |  |  employment = >=7: good (2.0)
|  |  |  |  |  employment = 1<=X<4: good (4.0)
|  |  |  |  |  employment = 4<=X<7: good (1.0)
```

```
|  |  |  |  |  employment = unemployed: good (0.0)
|  |  |  |  |  employment = <1: bad (3.0)
|  |  |  |  purpose = new car
|  |  |  |  |  own_telephone = yes: good (2.0)
|  |  |  |  |  own_telephone = none: bad (10.0/2.0)
|  |  |  |  purpose = used car: bad (1.0)
|  |  |  |  purpose = business: good (3.0)
|  |  |  |  purpose = domestic appliance: bad (1.0)
|  |  |  |  purpose = repairs: bad (1.0)
|  |  |  |  purpose = other: good (1.0)
|  |  |  |  purpose = retraining: good (1.0)
|  |  |  job = high qualif/self emp/mgmt: good (30.0/8.0)
|  |  |  job = unemp/unskilled non res: bad (5.0/1.0)
|  foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
|  credit_amount <= 9857
|  |  savings_status = no known savings: good (41.0/5.0)
|  |  savings_status = <100
|  |  |  other_parties = none
|  |  |  |  duration <= 42
|  |  |  |  |  personal_status = male single: good (52.0/15.0)
|  |  |  |  |  personal_status = female div/dep/mar
|  |  |  |  |  |  purpose = radio/tv: good (8.0/2.0)
|  |  |  |  |  |  purpose = education: good (4.0/2.0)
|  |  |  |  |  |  purpose = furniture/equipment
|  |  |  |  |  |  |  duration <= 10: bad (3.0)
|  |  |  |  |  |  |  duration > 10
|  |  |  |  |  |  |  |  duration <= 21: good (6.0/1.0)
|  |  |  |  |  |  |  |  duration > 21: bad (2.0)
|  |  |  |  |  |  purpose = new car: bad (5.0/1.0)
|  |  |  |  |  |  purpose = used car: bad (1.0)
|  |  |  |  |  |  purpose = business
|  |  |  |  |  |  |  residence_since <= 2: good (3.0)
|  |  |  |  |  |  |  residence_since > 2: bad (2.0)
|  |  |  |  |  |  purpose = domestic appliance: good (0.0)
|  |  |  |  |  |  purpose = repairs: good (1.0)
|  |  |  |  |  |  purpose = other: good (0.0)
|  |  |  |  |  |  purpose = retraining: good (0.0)
|  |  |  |  |  personal_status = male div/sep: bad (8.0/2.0)
|  |  |  |  |  personal_status = male mar/wid
|  |  |  |  |  |  duration <= 10: good (6.0)
|  |  |  |  |  |  duration > 10: bad (10.0/3.0)
|  |  |  |  duration > 42: bad (7.0)
|  |  |  other_parties = guarantor
|  |  |  |  purpose = radio/tv: good (18.0/1.0)
|  |  |  |  purpose = education: good (0.0)
```

```
|   |   |   |   purpose = furniture/equipment: good (0.0)
|   |   |   |   purpose = new car: bad (2.0)
|   |   |   |   purpose = used car: good (0.0)
|   |   |   |   purpose = business: good (0.0)
|   |   |   |   purpose = domestic appliance: good (0.0)
|   |   |   |   purpose = repairs: good (0.0)
|   |   |   |   purpose = other: good (0.0)
|   |   |   |   purpose = retraining: good (0.0)
|   |   |   other_parties = co applicant: good (2.0)
|   |   savings_status = 500<=X<1000: good (11.0/3.0)
|   |   savings_status = >=1000
|   |   |   duration <= 10: bad (2.0)
|   |   |   duration > 10: good (11.0/1.0)
|   |   savings_status = 100<=X<500
|   |   |   purpose = radio/tv: bad (8.0/2.0)
|   |   |   purpose = education: good (0.0)
|   |   |   purpose = furniture/equipment: bad (4.0/1.0)
|   |   |   purpose = new car: bad (15.0/5.0)
|   |   |   purpose = used car: good (3.0)
|   |   |   purpose = business
|   |   |   |   housing = own: good (6.0)
|   |   |   |   housing = for free: bad (1.0)
|   |   |   |   housing = rent
|   |   |   |   |   existing_credits <= 1: good (2.0)
|   |   |   |   |   existing_credits > 1: bad (2.0)
|   |   |   purpose = domestic appliance: good (0.0)
|   |   |   purpose = repairs: good (2.0)
|   |   |   purpose = other: good (1.0)
|   |   |   purpose = retraining: good (0.0)
|   credit_amount > 9857: bad (20.0/3.0)
checking_status = no checking: good (394.0/46.0)
checking_status = >=200: good (63.0/14.0)

Number of Leaves  :   87
Size of the tree :    119


Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         703               70.3    %
Incorrectly Classified Instances       297               29.7    %
Kappa statistic                          0.2508
Mean absolute error                      0.3518
```

Root mean squared error 0.4836
Relative absolute error 83.722 %
Root relative squared error 105.5281 %
Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.83	0.593	0.765	0.83	0.796	0.634	good
	0.407	0.17	0.506	0.407	0.451	0.634	bad
Weighted Avg.	0.703	0.466	0.688	0.703	0.693	0.634	

=== Confusion Matrix ===

a	b	<-- classified as
581	119	a = good
178	122	b = bad

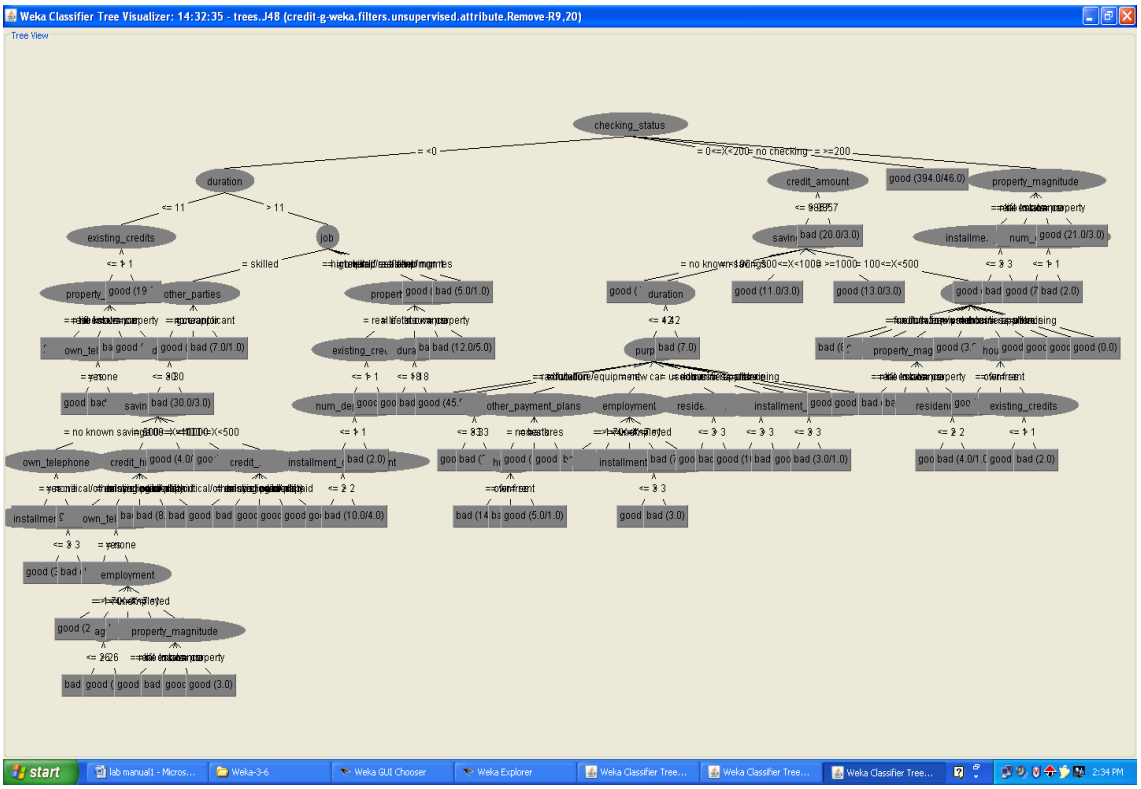
The accuracy is decreases compared with the use training dataset.

Task 7:-

Check to see if the data shows a bias against "foreign workers" (attribute 20) or "personal-status" (attribute 9(. One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the preprocess tab in weka in GUI Explorer. Did removing these attributes

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button. A new window is open and click the "preprocess" tab and open a "german.arff " file.
5) Next we can remove the foreign workers and personal-status attributes.
6) Next click the "classify" menu on the top.
7) Click the "choose" and list of "trees and rules" can be displayed.
8) Select the "trees" under click the "J48". And next right click on j48tree and click the "visualize tree"



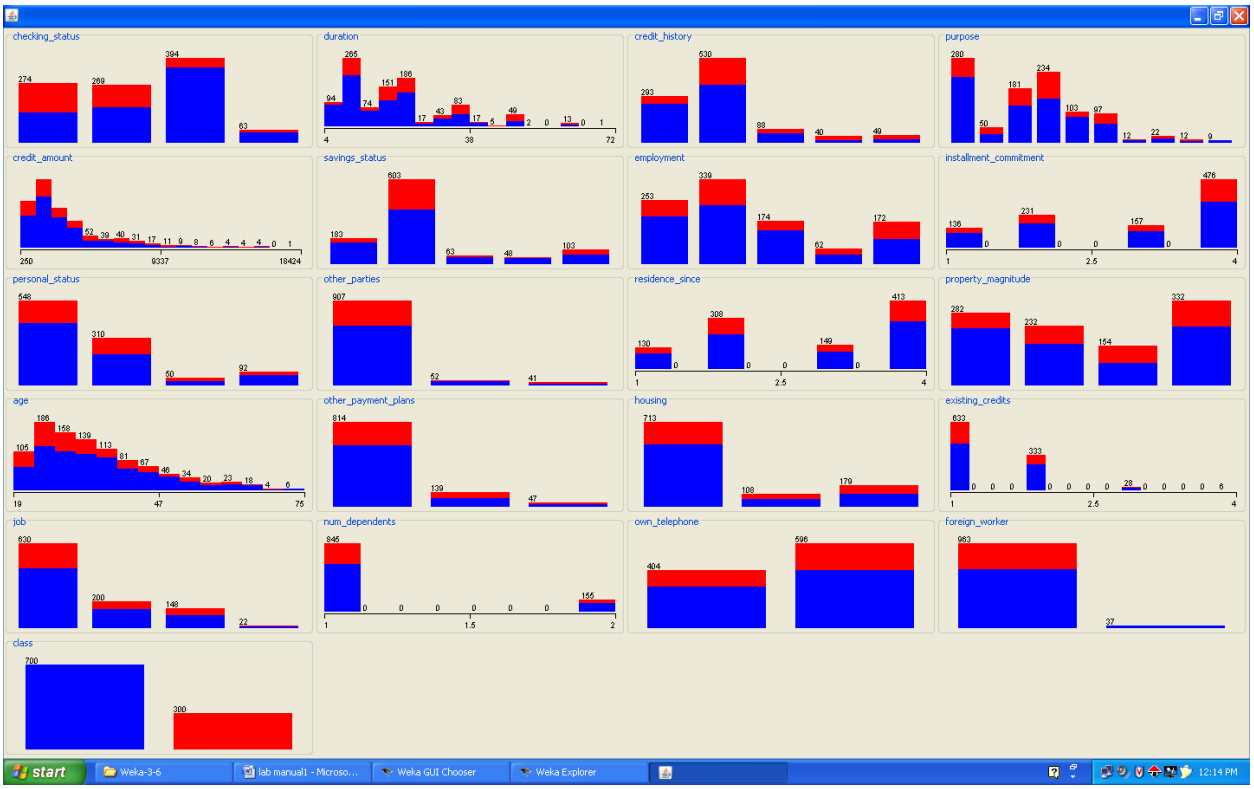
Task 8:-

Another question might be do you really need to input so many attributes to get good results? May be only a few would do. For example you could try just having attributes 2,3,5,7,10,17 (and 21 the class attribute naturally)). Try out some combinations (you had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes inintially before you start selecting the ones you want)

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Choose "visualize all" button and see the result.

Output:-



Task 9:-

Sometimes the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who had bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in weka. Train your decision tree against and report the decision tree and cross validation results. Are they significantly different from results obtained in problem 6 (using equal cost)

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Next click the "classify" menu on the top.
7) Click the "choose" and list of "trees and rules" can be displayed.
8) Select the "trees" under click the "J48". And choose the radio button is "using training dataset". And click "start". See the correctly classified instances.

9) Choose on result list and right click that one and click visualize “cost curve” is good can select.

Output:-

Using cross-validation dataset results

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     credit-g-weka.filters.unsupervised.attribute.Remove-R9,20
Instances:    1000
Attributes:   19
              checking_status
              duration
              credit_history
              purpose
              credit_amount
              savings_status
              employment
              installment_commitment
              other_parties
              residence_since
              property_magnitude
              age
              other_payment_plans
              housing
              existing_credits
              job
              num_dependents
              own_telephone
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

checking_status = <0
|   duration <= 11
|   |   existing_credits <= 1
|   |   |   property_magnitude = real estate: good (9.0/1.0)
|   |   |   property_magnitude = life insurance
```

```
|  |  |  |  own_telephone = yes: good (4.0)
|  |  |  |  own_telephone = none: bad (2.0)
|  |  |  property_magnitude = no known property: bad (3.0)
|  |  |  property_magnitude = car: good (2.0/1.0)
|  |  existing_credits > 1: good (19.0)
|  duration > 11
|  |  job = skilled
|  |  |  other_parties = none
|  |  |  |  duration <= 30
|  |  |  |  |  savings_status = no known savings
|  |  |  |  |  |  own_telephone = yes: good (6.0/1.0)
|  |  |  |  |  |  own_telephone = none
|  |  |  |  |  |  |  installment_commitment <= 3: good (3.0/1.0)
|  |  |  |  |  |  |  installment_commitment > 3: bad (7.0)
|  |  |  |  |  savings_status = <100
|  |  |  |  |  |  credit_history = critical/other existing credit: good (14.0/4.0)
|  |  |  |  |  |  credit_history = existing paid
|  |  |  |  |  |  |  own_telephone = yes: bad (5.0)
|  |  |  |  |  |  |  own_telephone = none
|  |  |  |  |  |  |  |  employment = >=7: good (2.0)
|  |  |  |  |  |  |  |  employment = 1<=X<4
|  |  |  |  |  |  |  |  |  age <= 26: bad (7.0/1.0)
|  |  |  |  |  |  |  |  |  age > 26: good (7.0/1.0)
|  |  |  |  |  |  |  |  employment = 4<=X<7: bad (5.0)
|  |  |  |  |  |  |  |  employment = unemployed: good (3.0/1.0)
|  |  |  |  |  |  |  |  employment = <1
|  |  |  |  |  |  |  |  |  property_magnitude = real estate: good (2.0)
|  |  |  |  |  |  |  |  |  property_magnitude = life insurance: bad (4.0)
|  |  |  |  |  |  |  |  |  property_magnitude = no known property: good (1.0)
|  |  |  |  |  |  |  |  |  property_magnitude = car: good (3.0)
|  |  |  |  |  |  credit_history = delayed previously: bad (4.0)
|  |  |  |  |  |  credit_history = no credits/all paid: bad (8.0/1.0)
|  |  |  |  |  |  credit_history = all paid: bad (6.0)
|  |  |  |  |  savings_status = 500<=X<1000: good (4.0/1.0)
|  |  |  |  |  savings_status = >=1000: good (4.0)
|  |  |  |  |  savings_status = 100<=X<500
|  |  |  |  |  |  credit_history = critical/other existing credit: good (2.0)
|  |  |  |  |  |  credit_history = existing paid: bad (3.0)
|  |  |  |  |  |  credit_history = delayed previously: good (0.0)
|  |  |  |  |  |  credit_history = no credits/all paid: good (0.0)
|  |  |  |  |  |  credit_history = all paid: good (1.0)
|  |  |  |  duration > 30: bad (30.0/3.0)
|  |  |  other_parties = guarantor: good (14.0/4.0)
|  |  |  other_parties = co applicant: bad (7.0/1.0)
|  |  job = unskilled resident
|  |  |  property_magnitude = real estate
```

```
|   |   |   |   existing_credits <= 1
|   |   |   |   |   num_dependents <= 1
|   |   |   |   |   |   installment_commitment <= 2: good (3.0)
|   |   |   |   |   |   installment_commitment > 2: bad (10.0/4.0)
|   |   |   |   |   num_dependents > 1: bad (2.0)
|   |   |   |   existing_credits > 1: good (3.0)
|   |   |   property_magnitude = life insurance
|   |   |   |   duration <= 18: good (9.0)
|   |   |   |   duration > 18: bad (3.0/1.0)
|   |   |   property_magnitude = no known property: bad (5.0)
|   |   |   property_magnitude = car: bad (12.0/5.0)
|   |   job = high qualif/self emp/mgmt: good (31.0/9.0)
|   |   job = unemp/unskilled non res: bad (5.0/1.0)
checking_status = 0<=X<200
|   credit_amount <= 9857
|   |   savings_status = no known savings: good (41.0/5.0)
|   |   savings_status = <100
|   |   |   duration <= 42
|   |   |   |   purpose = radio/tv: good (45.0/8.0)
|   |   |   |   purpose = education
|   |   |   |   |   age <= 33: good (2.0)
|   |   |   |   |   age > 33: bad (3.0/1.0)
|   |   |   |   purpose = furniture/equipment
|   |   |   |   |   other_payment_plans = none
|   |   |   |   |   |   housing = own: bad (14.0/5.0)
|   |   |   |   |   |   housing = for free: bad (0.0)
|   |   |   |   |   |   housing = rent: good (5.0/1.0)
|   |   |   |   |   other_payment_plans = bank: good (2.0/1.0)
|   |   |   |   |   other_payment_plans = stores: good (2.0)
|   |   |   |   purpose = new car
|   |   |   |   |   employment = >=7: bad (5.0)
|   |   |   |   |   employment = 1<=X<4: good (5.0/2.0)
|   |   |   |   |   employment = 4<=X<7: good (5.0/1.0)
|   |   |   |   |   employment = unemployed
|   |   |   |   |   |   installment_commitment <= 3: good (2.0)
|   |   |   |   |   |   installment_commitment > 3: bad (3.0)
|   |   |   |   |   employment = <1: bad (7.0/2.0)
|   |   |   |   purpose = used car
|   |   |   |   |   residence_since <= 3: good (6.0)
|   |   |   |   |   residence_since > 3: bad (3.0/1.0)
|   |   |   |   purpose = business
|   |   |   |   |   residence_since <= 3: good (10.0/2.0)
|   |   |   |   |   residence_since > 3: bad (5.0)
|   |   |   |   purpose = domestic appliance: good (1.0)
|   |   |   |   purpose = repairs
|   |   |   |   |   installment_commitment <= 3: good (3.0)
```

```
|  |  |  |  |  installment_commitment > 3: bad (3.0/1.0)
|  |  |  |  purpose = other: good (1.0)
|  |  |  |  purpose = retraining: good (1.0)
|  |  |  duration > 42: bad (7.0)
|  |  savings_status = 500<=X<1000: good (11.0/3.0)
|  |  savings_status = >=1000: good (13.0/3.0)
|  |  savings_status = 100<=X<500
|  |  |  purpose = radio/tv: bad (8.0/2.0)
|  |  |  purpose = education: good (0.0)
|  |  |  purpose = furniture/equipment: bad (4.0/1.0)
|  |  |  purpose = new car
|  |  |  |  property_magnitude = real estate: bad (0.0)
|  |  |  |  property_magnitude = life insurance: bad (6.0)
|  |  |  |  property_magnitude = no known property: good (2.0/1.0)
|  |  |  |  property_magnitude = car
|  |  |  |  |  residence_since <= 2: good (3.0)
|  |  |  |  |  residence_since > 2: bad (4.0/1.0)
|  |  |  purpose = used car: good (3.0)
|  |  |  purpose = business
|  |  |  |  housing = own: good (6.0)
|  |  |  |  housing = for free: bad (1.0)
|  |  |  |  housing = rent
|  |  |  |  |  existing_credits <= 1: good (2.0)
|  |  |  |  |  existing_credits > 1: bad (2.0)
|  |  |  purpose = domestic appliance: good (0.0)
|  |  |  purpose = repairs: good (2.0)
|  |  |  purpose = other: good (1.0)
|  |  |  purpose = retraining: good (0.0)
|  credit_amount > 9857: bad (20.0/3.0)
checking_status = no checking: good (394.0/46.0)
checking_status = >=200
|  property_magnitude = real estate
|  |  installment_commitment <= 3: good (15.0/3.0)
|  |  installment_commitment > 3: bad (6.0/1.0)
|  property_magnitude = life insurance: good (12.0)
|  property_magnitude = no known property
|  |  num_dependents <= 1: good (7.0/1.0)
|  |  num_dependents > 1: bad (2.0)
|  property_magnitude = car: good (21.0/3.0)
Number of Leaves  :  95
Size of the tree :  137
Time taken to build model: 0.03 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances         716               71.6    %
Incorrectly Classified Instances       284               28.4    %
```

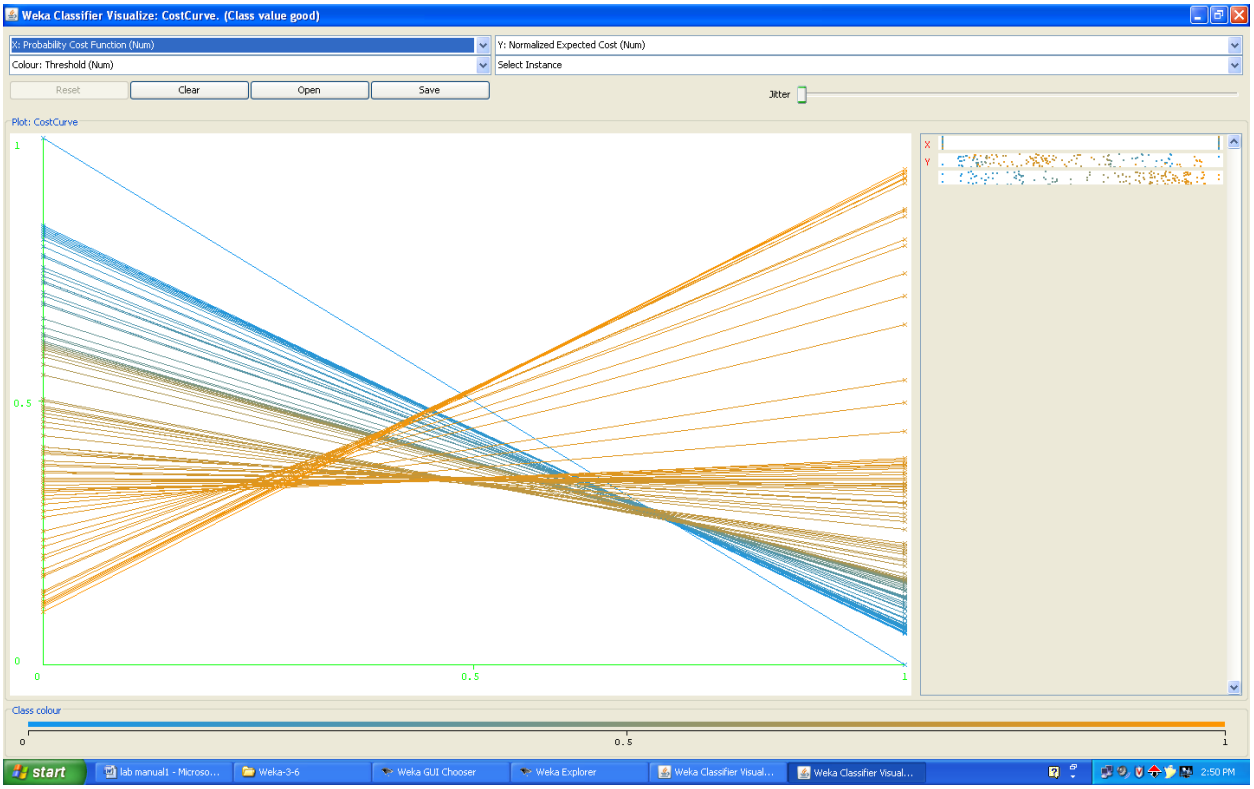
Kappa statistic 0.2843
Mean absolute error 0.3328
Root mean squared error 0.477
Relative absolute error 79.2118 %
Root relative squared error 104.0916 %
Total Number of Instances 1000
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.839	0.57	0.774	0.839	0.805	0.66	good
	0.43	0.161	0.533	0.43	0.476	0.66	bad
Weighted Avg.	0.716	0.447	0.702	0.716	0.706	0.66	

=== Confusion Matrix ===

a	b	<-- classified as
587	113	a = good
171	129	b = bad

Cost curve:-



Using training dataset results

=== Run information ===

```
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     credit-g-weka.filters.unsupervised.attribute.Remove-R9,20
Instances:    1000
Attributes:   19
              checking_status
              duration
              credit_history
              purpose
              credit_amount
              savings_status
              employment
              installment_commitment
              other_parties
              residence_since
              property_magnitude
              age
              other_payment_plans
              housing
              existing_credits
              job
              num_dependents
              own_telephone
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
------------------

checking_status = <0
|   duration <= 11
|   |   existing_credits <= 1
|   |   |   property_magnitude = real estate: good (9.0/1.0)
|   |   |   property_magnitude = life insurance
|   |   |   |   own_telephone = yes: good (4.0)
|   |   |   |   own_telephone = none: bad (2.0)
|   |   |   property_magnitude = no known property: bad (3.0)
|   |   |   property_magnitude = car: good (2.0/1.0)
|   |   existing_credits > 1: good (19.0)
|   duration > 11
|   |   job = skilled
|   |   |   other_parties = none
|   |   |   |   duration <= 30
|   |   |   |   |   savings_status = no known savings
|   |   |   |   |   |   own_telephone = yes: good (6.0/1.0)
```

```
|  |  |  |  |  |  own_telephone = none
|  |  |  |  |  |  |  installment_commitment <= 3: good (3.0/1.0)
|  |  |  |  |  |  |  installment_commitment > 3: bad (7.0)
|  |  |  |  |  savings_status = <100
|  |  |  |  |  |  credit_history = critical/other existing credit: good (14.0/4.0)
|  |  |  |  |  |  credit_history = existing paid
|  |  |  |  |  |  |  own_telephone = yes: bad (5.0)
|  |  |  |  |  |  |  own_telephone = none
|  |  |  |  |  |  |  |  employment = >=7: good (2.0)
|  |  |  |  |  |  |  |  employment = 1<=X<4
|  |  |  |  |  |  |  |  |  age <= 26: bad (7.0/1.0)
|  |  |  |  |  |  |  |  |  age > 26: good (7.0/1.0)
|  |  |  |  |  |  |  |  employment = 4<=X<7: bad (5.0)
|  |  |  |  |  |  |  |  employment = unemployed: good (3.0/1.0)
|  |  |  |  |  |  |  |  employment = <1
|  |  |  |  |  |  |  |  |  property_magnitude = real estate: good (2.0)
|  |  |  |  |  |  |  |  |  property_magnitude = life insurance: bad (4.0)
|  |  |  |  |  |  |  |  |  property_magnitude = no known property: good (1.0)
|  |  |  |  |  |  |  |  |  property_magnitude = car: good (3.0)
|  |  |  |  |  |  credit_history = delayed previously: bad (4.0)
|  |  |  |  |  |  credit_history = no credits/all paid: bad (8.0/1.0)
|  |  |  |  |  |  credit_history = all paid: bad (6.0)
|  |  |  |  |  savings_status = 500<=X<1000: good (4.0/1.0)
|  |  |  |  |  savings_status = >=1000: good (4.0)
|  |  |  |  |  savings_status = 100<=X<500
|  |  |  |  |  |  credit_history = critical/other existing credit: good (2.0)
|  |  |  |  |  |  credit_history = existing paid: bad (3.0)
|  |  |  |  |  |  credit_history = delayed previously: good (0.0)
|  |  |  |  |  |  credit_history = no credits/all paid: good (0.0)
|  |  |  |  |  |  credit_history = all paid: good (1.0)
|  |  |  |  duration > 30: bad (30.0/3.0)
|  |  |  other_parties = guarantor: good (14.0/4.0)
|  |  |  other_parties = co applicant: bad (7.0/1.0)
|  |  job = unskilled resident
|  |  |  property_magnitude = real estate
|  |  |  |  existing_credits <= 1
|  |  |  |  |  num_dependents <= 1
|  |  |  |  |  |  installment_commitment <= 2: good (3.0)
|  |  |  |  |  |  installment_commitment > 2: bad (10.0/4.0)
|  |  |  |  |  num_dependents > 1: bad (2.0)
|  |  |  |  existing_credits > 1: good (3.0)
|  |  |  property_magnitude = life insurance
|  |  |  |  duration <= 18: good (9.0)
|  |  |  |  duration > 18: bad (3.0/1.0)
|  |  |  property_magnitude = no known property: bad (5.0)
|  |  |  property_magnitude = car: bad (12.0/5.0)
```

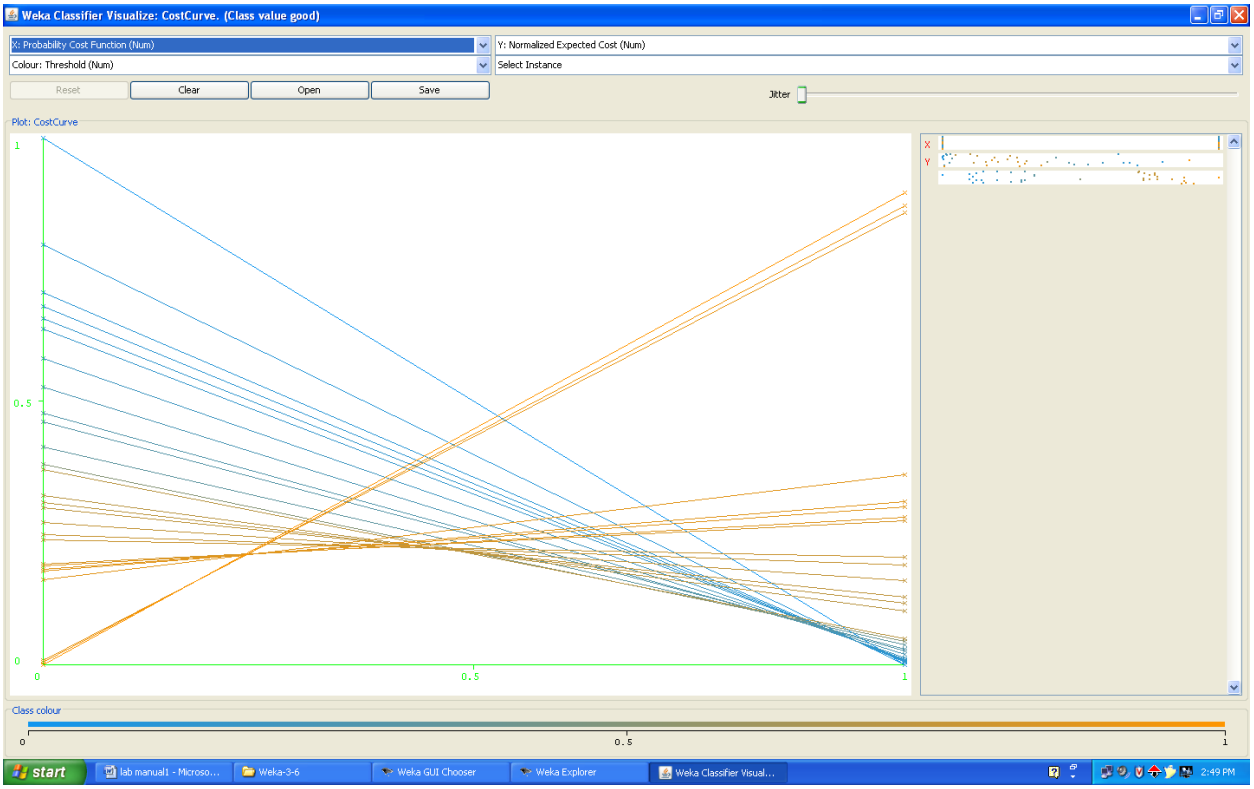
```
|   |   job = high qualif/self emp/mgmt: good (31.0/9.0)
|   |   job = unemp/unskilled non res: bad (5.0/1.0)
checking_status = 0<=X<200
|   credit_amount <= 9857
|   |   savings_status = no known savings: good (41.0/5.0)
|   |   savings_status = <100
|   |   |   duration <= 42
|   |   |   |   purpose = radio/tv: good (45.0/8.0)
|   |   |   |   purpose = education
|   |   |   |   |   age <= 33: good (2.0)
|   |   |   |   |   age > 33: bad (3.0/1.0)
|   |   |   |   purpose = furniture/equipment
|   |   |   |   |   other_payment_plans = none
|   |   |   |   |   |   housing = own: bad (14.0/5.0)
|   |   |   |   |   |   housing = for free: bad (0.0)
|   |   |   |   |   |   housing = rent: good (5.0/1.0)
|   |   |   |   |   other_payment_plans = bank: good (2.0/1.0)
|   |   |   |   |   other_payment_plans = stores: good (2.0)
|   |   |   |   purpose = new car
|   |   |   |   |   employment = >=7: bad (5.0)
|   |   |   |   |   employment = 1<=X<4: good (5.0/2.0)
|   |   |   |   |   employment = 4<=X<7: good (5.0/1.0)
|   |   |   |   |   employment = unemployed
|   |   |   |   |   |   installment_commitment <= 3: good (2.0)
|   |   |   |   |   |   installment_commitment > 3: bad (3.0)
|   |   |   |   |   employment = <1: bad (7.0/2.0)
|   |   |   |   purpose = used car
|   |   |   |   |   residence_since <= 3: good (6.0)
|   |   |   |   |   residence_since > 3: bad (3.0/1.0)
|   |   |   |   purpose = business
|   |   |   |   |   residence_since <= 3: good (10.0/2.0)
|   |   |   |   |   residence_since > 3: bad (5.0)
|   |   |   |   purpose = domestic appliance: good (1.0)
|   |   |   |   purpose = repairs
|   |   |   |   |   installment_commitment <= 3: good (3.0)
|   |   |   |   |   installment_commitment > 3: bad (3.0/1.0)
|   |   |   |   purpose = other: good (1.0)
|   |   |   |   purpose = retraining: good (1.0)
|   |   |   duration > 42: bad (7.0)
|   |   savings_status = 500<=X<1000: good (11.0/3.0)
|   |   savings_status = >=1000: good (13.0/3.0)
|   |   savings_status = 100<=X<500
|   |   |   purpose = radio/tv: bad (8.0/2.0)
|   |   |   purpose = education: good (0.0)
|   |   |   purpose = furniture/equipment: bad (4.0/1.0)
|   |   |   purpose = new car
```

```
|  |  |  |  property_magnitude = real estate: bad (0.0)
|  |  |  |  property_magnitude = life insurance: bad (6.0)
|  |  |  |  property_magnitude = no known property: good (2.0/1.0)
|  |  |  |  property_magnitude = car
|  |  |  |  |  residence_since <= 2: good (3.0)
|  |  |  |  |  residence_since > 2: bad (4.0/1.0)
|  |  |  purpose = used car: good (3.0)
|  |  |  purpose = business
|  |  |  |  housing = own: good (6.0)
|  |  |  |  housing = for free: bad (1.0)
|  |  |  |  housing = rent
|  |  |  |  |  existing_credits <= 1: good (2.0)
|  |  |  |  |  existing_credits > 1: bad (2.0)
|  |  |  purpose = domestic appliance: good (0.0)
|  |  |  purpose = repairs: good (2.0)
|  |  |  purpose = other: good (1.0)
|  |  |  purpose = retraining: good (0.0)
|  credit_amount > 9857: bad (20.0/3.0)
checking_status = no checking: good (394.0/46.0)
checking_status = >=200
|  property_magnitude = real estate
|  |  installment_commitment <= 3: good (15.0/3.0)
|  |  installment_commitment > 3: bad (6.0/1.0)
|  property_magnitude = life insurance: good (12.0)
|  property_magnitude = no known property
|  |  num_dependents <= 1: good (7.0/1.0)
|  |  num_dependents > 1: bad (2.0)
|  property_magnitude = car: good (21.0/3.0)
Number of Leaves  :   95
Size of the tree :    137
Time taken to build model: 0.02 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances         861               86.1    %
Incorrectly Classified Instances       139               13.9    %
Kappa statistic                          0.6458
Mean absolute error                      0.2194
Root mean squared error                  0.3312
Relative absolute error                 52.208  %
Root relative squared error             72.2688 %
Total Number of Instances             1000
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.95      0.347      0.865     0.95      0.905      0.869    good
                 0.653     0.05       0.848     0.653     0.738      0.869    bad
Weighted Avg.    0.861     0.258      0.86      0.861     0.855      0.869
```

=== Confusion Matrix ===

a	b	<-- classified as
665	35	a = good
104	196	b = bad

Cost curve:-



Task 10:-

Do you think it is a good idea to prefer simple decision tree instead of having long complex decision trees? How does the complexity of a decision tree relate to the bias of the model?

Description:-

- The computational complexity of the decision tree induction is O(n) stands for a quantity that grows at most linearly with n, O(n2) grows at most quadratic ally with n.
- Suppose the training data contains n instances and m attributes. We need to make some assumption about the size of the tree, and we will assume that its depth is on the order of log n, that is O(log n).
- The computational cost of building the tree in the first place is O(mn log n).

- Because there are log n different depths in the tree, the amount of work for this on attribute is O(n log n) . at each node all attributes are considered so the total amount of work is O(mn log n).
- The initial sort takes O(n log n) operations for each of up to m attributes thus the preceding complexity figure is unchanged.
- The complexity of sub-tree replacement is O(n).
- Finally sub-tree lifting has a basic complexity equal to sub-tree replacement. But there is an added cost because instances need to be reclassified during the lifting operation. During the whole process each instance may have to be reclassified at every node between its leaf and the root, I.e, as many as O(log n) times. That makes the total number of reclassification O(n log n)
- And reclassification is not a single operation one that occurs near the root will take O(log n) operations, and one of the average depth will take half of this. Thus the total complexity of sub-tree is as follows.
- O(n(log n)2)\taking into account all these operations, the full complexity of decision tree induction is
- O(mn log n)+O(n(log n)2).

Task 11:-

You can make your Decision tree simpler by pruning the nodes. One approach is to use Reduce error pruning-Explain this idea briefly. Try reduce error pruning for training your Decision tree using cross-validation (you can do this in weka) and report the Decision tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increases?

Description:-

We need to estimate the error at internal nodes as well as at leaf nodes. If we had such an estimate, it would be clear whether to replace, or raise, a particular sub-tree simply by comparing the estimated error of the sub-tree with that of its proposed replacement. Before estimating the error for sub-tree proposed for raising.

One Way of coming up with an error estimate is the standard verification technique is hold back some of the data originally given and use it as an independent test set to estimate the error at each node. This is called reduced-error pruning.

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.
6) Next click the "classify" menu on the top.
7) Click the "choose" and list of "trees and rules" can be displayed.

8) Select the "trees" under click the "J48". And choose the radio button is "using training dataset". And click "start". See the correctly classified instances.

Output:-

=== Evaluation on training set ===

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.956	0.38	0.854	0.956	0.902	0.857	good
	0.62	0.044	0.857	0.62	0.72	0.857	bad
Weighted Avg.	0.855	0.279	0.855	0.855	0.847	0.857	

=== Stratified cross-validation ===

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.843	0.61	0.763	0.843	0.801	0.641	good
	0.39	0.157	0.515	0.39	0.444	0.641	bad
Weighted Avg.	0.707	0.474	0.689	0.707	0.694	0.641	

Task 12:-

How can you convert a Decision tree into "if-then-else-rules". Make up your own small Decision tree consisting of 2-3 level s and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules- one such classifier in weka is rules. PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one. Can you predict what attribute that might be in this dataset? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a OneR classifier. Rank the performance of j48,PART,and OneR.

Procedure:-

1) Insert the data into the excel sheet and save the file is ".CSV".
2) Click on weka executable jar file.
3) A window can open that window contains the for buttons that are "Explorer, Experimenter, Knowledge flow, simple CLI".
4) Click the "Explorer" button.
5) A new window is open and click the "preprocess" tab and open a "german.arff " file.

6) Next click the “classify” menu on the top.
7) Click the “choose” and list of “trees and rules” can be displayed.
8) Select the “rules” under click the “PART and OneR”. And choose the radio button is “using training dataset”. And click “start”. See the correctly classified instances.

Output:-

PART Rule

=== Run information ===

Scheme: weka.classifiers.rules.PART -M 2 -C 0.25 -Q 1

Relation: credit-g

Instances: 1000

Attributes: 21

checking_status

duration

credit_history

purpose

credit_amount

savings_status

employment

installment_commitment

personal_status

other_parties

residence_since

property_magnitude

age

other_payment_plans

housing

existing_credits

job

num_dependents

own_telephone

foreign_worker

class

Test mode: evaluate on training data

=== Classifier model (full training set) ===

PART decision list

---

checking_status = no checking AND

other_payment_plans = none AND

credit_history = critical/other existing credit: good (134.0/3.0)

checking_status = no checking AND

existing_credits <= 1 AND

other_payment_plans = none AND

purpose = radio/tv: good (49.0/2.0)

checking_status = no checking AND

foreign_worker = yes AND

employment = 4<=X<7: good (35.0/2.0)

foreign_worker = no AND

personal_status = male single: good (21.0)

checking_status = no checking AND

purpose = used car AND

other_payment_plans = none: good (23.0)

duration <= 15 AND

other_parties = guarantor: good (22.0/1.0)

duration <= 11 AND

credit_history = critical/other existing credit: good (29.0/3.0)

checking_status = >=200 AND

num_dependents <= 1 AND

property_magnitude = car: good (20.0/3.0)

checking_status = no checking AND

property_magnitude = real estate AND

other_payment_plans = none AND

age > 23: good (25.0)

savings_status = >=1000 AND

property_magnitude = real estate: good (10.0)

savings_status = 500<=X<1000 AND

employment = >=7: good (13.0/1.0)

credit_history = no credits/all paid AND

housing = rent: bad (9.0)

savings_status = no known savings AND

checking_status = 0<=X<200 AND

existing_credits > 1: good (9.0)

checking_status = >=200 AND

num_dependents <= 1 AND

property_magnitude = life insurance: good (9.0)

installment_commitment <= 2 AND

other_parties = co applicant AND

existing_credits > 1: bad (5.0)

installment_commitment <= 2 AND

credit_history = delayed previously AND

existing_credits > 1 AND

residence_since > 1: good (14.0/3.0)

installment_commitment <= 2 AND

credit_history = delayed previously AND

existing_credits <= 1: good (9.0)

duration > 30 AND

savings_status = 100<=X<500: bad (13.0/3.0)

credit_history = all paid AND

other_parties = none AND

other_payment_plans = bank: bad (16.0/5.0)

duration > 30 AND

savings_status = no known savings AND

num_dependents > 1: good (5.0)

duration > 30 AND

credit_history = delayed previously: bad (9.0)

duration > 42 AND

savings_status = <100 AND

residence_since > 1: bad (28.0/3.0)

purpose = used car AND

credit_amount <= 8133 AND

existing_credits > 1: good (11.0)

purpose = used car AND

credit_amount > 8133: bad (8.0/1.0)

purpose = used car AND

employment = 1<=X<4: good (7.0)

purpose = used car: good (16.0/3.0)

purpose = furniture/equipment AND

other_payment_plans = stores: good (8.0)

credit_history = all paid AND

other_parties = none AND

other_payment_plans = none: bad (10.0)

purpose = business AND

residence_since <= 1: good (9.0)

other_payment_plans = stores AND

purpose = radio/tv AND

personal_status = male single: bad (6.0/1.0)

purpose = radio/tv AND

employment = >=7 AND

num_dependents <= 1: good (20.0/1.0)

installment_commitment <= 3 AND

purpose = furniture/equipment AND

other_parties = none AND

own_telephone = yes: good (19.0/3.0)

checking_status = no checking AND

savings_status = no known savings AND

personal_status = male single: good (11.0/1.0)

checking_status = 0<=X<200 AND

employment = 4<=X<7 AND

personal_status = male single AND

residence_since > 2: good (9.0)

purpose = other: good (5.0/1.0)

installment_commitment <= 2 AND

foreign_worker = yes AND

credit_history = existing paid AND

residence_since > 1 AND

other_parties = none AND

other_payment_plans = none AND

housing = rent AND

installment_commitment <= 1: good (9.0)

housing = rent AND

other_payment_plans = none AND

purpose = new car: bad (13.0/2.0)

other_payment_plans = stores AND

property_magnitude = life insurance: bad (4.0/1.0)

other_payment_plans = bank AND

other_parties = none AND

housing = rent: bad (7.0/1.0)

installment_commitment > 3 AND

existing_credits <= 1 AND

savings_status = <100 AND

credit_history = existing paid AND

purpose = new car: bad (17.0/5.0)

checking_status = >=200 AND

job = skilled AND

credit_amount > 1283: good (5.0)

checking_status = >=200: bad (9.0/1.0)

duration <= 15 AND

property_magnitude = real estate AND

num_dependents <= 1 AND

other_payment_plans = none AND

foreign_worker = yes AND

existing_credits <= 1: good (22.0/5.0)

num_dependents > 1 AND

own_telephone = yes: good (14.0/2.0)

purpose = repairs AND

property_magnitude = car: good (5.0/1.0)

purpose = education AND

checking_status = <0: bad (9.0/1.0)

purpose = repairs: bad (7.0/1.0)

purpose = education AND

checking_status = 0<=X<200: good (5.0)

purpose = education AND

personal_status = male single: bad (3.0)

purpose = business AND

age > 27: bad (14.0/3.0)

purpose = new car AND

other_payment_plans = bank: bad (13.0/2.0)

purpose = radio/tv AND

foreign_worker = yes AND

other_parties = guarantor AND

existing_credits <= 1: good (5.0/1.0)

purpose = radio/tv AND

other_parties = none AND

foreign_worker = yes AND

other_payment_plans = bank AND

num_dependents <= 1: good (6.0/1.0)

purpose = radio/tv AND

foreign_worker = yes AND

other_parties = none AND

other_payment_plans = none AND

checking_status = <0 AND

credit_history = existing paid AND

savings_status = <100 AND

job = skilled: bad (9.0/1.0)

purpose = radio/tv AND

other_parties = none AND

foreign_worker = yes AND

other_payment_plans = none AND

credit_history = existing paid AND

savings_status = <100 AND

credit_amount > 2051: good (6.0)

purpose = business: good (5.0)

housing = for free AND

other_payment_plans = none AND

residence_since > 2: good (12.0/3.0)

purpose = new car AND

other_parties = none AND

housing = own AND

num_dependents <= 1 AND

savings_status = <100 AND

checking_status = <0 AND

property_magnitude = car: bad (6.0/1.0)

purpose = radio/tv AND

foreign_worker = yes AND

other_parties = none AND

other_payment_plans = none AND

existing_credits > 1 AND

credit_amount <= 2225: good (4.0)

purpose = radio/tv AND

foreign_worker = yes AND

other_parties = none AND

job = unskilled resident: bad (6.0)

purpose = furniture/equipment AND

other_parties = co applicant AND

duration > 15: bad (4.0)

employment = unemployed AND

residence_since <= 2: bad (9.0)

purpose = new car AND

other_parties = none AND

property_magnitude = car AND

installment_commitment <= 2: good (5.0)

purpose = furniture/equipment AND

other_parties = none AND

savings_status = <100 AND

num_dependents <= 1 AND

own_telephone = yes: bad (5.0/1.0)

purpose = furniture/equipment AND

own_telephone = none AND

other_parties = none AND

duration > 27: bad (9.0/1.0)

purpose = furniture/equipment AND

other_parties = none AND

personal_status = male single AND

residence_since > 3: good (6.0)

purpose = furniture/equipment AND

other_parties = none AND

personal_status = female div/dep/mar AND

job = skilled AND

employment = 1<=X<4: bad (6.0/2.0)

purpose = furniture/equipment AND

savings_status = <100 AND

housing = own AND

other_parties = none AND

personal_status = male single: bad (5.0/1.0)

purpose = furniture/equipment: good (25.0/5.0)

other_parties = none AND

purpose = radio/tv AND

foreign_worker = yes AND

personal_status = male mar/wid: bad (5.0)

other_parties = none AND

purpose = radio/tv AND

other_payment_plans = none AND

own_telephone = none: good (8.0/1.0)

purpose = new car AND

other_parties = none AND

num_dependents <= 1 AND

credit_history = existing paid AND

own_telephone = yes AND

age > 29: good (5.0/1.0)

credit_history = delayed previously AND

age <= 45: good (4.0)

other_payment_plans = none AND

other_parties = none AND

housing = own AND

personal_status = female div/dep/mar AND

installment_commitment > 1: bad (10.0)

other_parties = none AND

housing = own AND

purpose = radio/tv AND

existing_credits <= 1: good (4.0)

housing = own AND

other_parties = none AND

purpose = new car AND

installment_commitment > 3 AND

savings_status = <100: bad (7.0/2.0)

checking_status = <0: bad (10.0/2.0)

: good (12.0/3.0)

Number of Rules : 78

Time taken to build model: 0.13 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	909	90.9 %
Incorrectly Classified Instances	91	9.1 %
Kappa statistic	0.7815	
Mean absolute error	0.1485	
Root mean squared error	0.2725	
Relative absolute error	35.3384 %	
Root relative squared error	59.4574 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.941	0.167	0.929	0.941	0.935	0.949	good
	0.833	0.059	0.859	0.833	0.846	0.949	bad
Weighted Avg.	0.909	0.134	0.908	0.909	0.909	0.949	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 659  41 |   a = good
  50 250 |   b = bad
```

OneR Rule

=== Run information ===

Scheme: weka.classifiers.rules.OneR -B 6
Relation: credit-g
Instances: 1000
Attributes: 21

- checking_status
- duration
- credit_history
- purpose
- credit_amount
- savings_status
- employment
- installment_commitment
- personal_status
- other_parties
- residence_since
- property_magnitude
- age
- other_payment_plans

housing

existing_credits

job

num_dependents

own_telephone

foreign_worker

class

Test mode: evaluate on training data

=== Classifier model (full training set) ===

credit_amount:

< 883.0 -> good

< 922.0 -> bad

< 938.0 -> good

< 979.5 -> bad

< 1206.5 -> good

< 1223.5 -> bad

< 1267.5 -> good

< 1286.0 -> bad

< 1821.5 -> good

< 1865.5 -> bad

< 3913.5 -> good

< 3969.0 -> bad

< 4049.5 -> good

< 4329.5 -> bad

< 4726.0 -> good

< 5024.0 -> bad

< 6322.5 -> good

< 6564.0 -> bad

< 6750.0 -> good
< 6917.5 -> bad
< 7760.5 -> good
< 8109.5 -> bad
< 9340.5 -> good
< 10331.5 -> bad
< 11191.0 -> good
>= 11191.0 -> bad

(742/1000 instances correct)

Time taken to build model: 0.02 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	742	74.2 %
Incorrectly Classified Instances	258	25.8 %
Kappa statistic	0.2896	
Mean absolute error	0.258	
Root mean squared error	0.5079	
Relative absolute error	61.4052 %	
Root relative squared error	110.8409 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.917	0.667	0.762	0.917	0.833	0.625	good

	0.333	0.083	0.633	0.333	0.437	0.625 bad
Weighted Avg.	0.742	0.492	0.724	0.742	0.714	0.625

=== Confusion Matrix ===

a	b	<-- classified as
642	58	a = good
200	100	b = bad

Rank of the j48,PART,OneR is

Correctly classified instances in PART is:- 90.9%

Correctly classified instances in j48 is:- 83.3%

Correctly classified instances in OneR is:- 74.2%