DATA MINING LAB

1. LAB OBJECTIVE

- BI Portal Lab: The objective of the lab exercise is to integrate pro-built reports into a portal application.
- Metadata & ETL Lab: The objective of the lab exercises is to implement metadata import agents to pull metadata from leading business intelligence tools and populate a metadata repository. To understand ETL process.
- Data Mining: The Objective of the lab exercise is to implement various Algorithms using DM Tools (E.g. WEKA, Yale)

Data Mining LAB:

2. LAB Outcome

Upon successful completion of this Lab the student will be able to:

- Pre-Process the user huge input data of various databases.
- Mine the association rules from the transactional data bases using automated data mining tool called WEKA.
- Mine the Classification models using automated data mining tool called WEKA.
- Mine the clusters using automated data mining tool called WEKA.
- Mine the outliers using automated data mining tool called WEKA.
- Work with data mining tool such as WEKA.

3. Introduction About Data Mining LAB

There are 60 systems (Compaq Presario) installed in this Lab. Their configurations are as follows:

Processor	:	Intel(R) Pentium(R) Dual CPU 2.0GH$_z$
RAM	:	2 GB
Hard Disk	:	160 GB
Mouse	:	Optical Mouse

Software:

1. All systems are configured in DUAL BOOT mode i.e., Students can boot from Windows XP or Linux as per their lab requirement.

3.1 WEKA INTRODUCTION

Weka is created by researchers at the university WIKATO in New Zealand. University of Waikato, Hamilton, New Zealand Alex Seewald (original Command-line primer) David Scuse (original Experimenter tutorial)

- It is java based application.
- It is collection often source, Machine Learning Algorithm.
- The routines (functions) are implemented as classes and logically arranged in packages.
- It comes with an extensive GUI Interface.
- Weka routines can be used standalone via the command line interface.

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting point for launching Weka‘s main GUI applications and supporting tools.

If one prefers a MDI (–multiple document interface‖) appearance, then this is provided by an alternative launcher called –Main‖ (class weka.gui.Main).

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

Fig 1:Weka GUI Chooser

The buttons can be used to start the following applications:

- **Explorer An environment** for exploring data with WEKA (the rest of this documentation deals with this application in more detail).
- **Experimenter** An environment for performing experiments and conducting statistical tests between learning schemes.
- **Knowledge Flow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.
- **SimpleCLI Provides** a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface

I. Explorer

The Graphical user interface

Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are grayed 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:

- **Preprocess.** Choose and modify the data being acted on.
- **Classify.** Train & test learning schemes that classify or perform regression
- **Cluster.** Learn clusters for the data.
- **Associate.** Learn association rules for the data.
- **Select attributes.** Select the most relevant attributes in the data.
- **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. The Explorer can be easily extended with custom tabs. The Wiki article –Adding tabs in the Explorer

II. Experimenter

Introduction

The Weka Experiment Environment enables the user to create, run, modify, and analyze experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyze the results to determine if one of the schemes is (statistically) better than the other schemes.

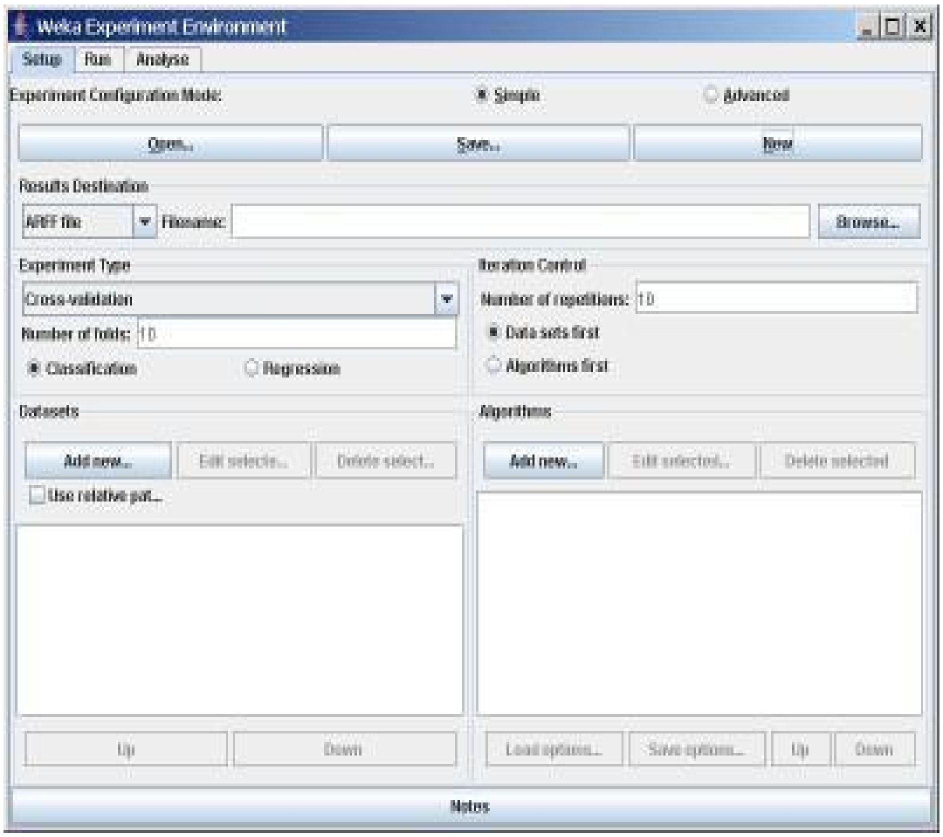

Fig 2:Weka Experiment Environment

The Experiment Environment can be run from the command line using the Simple CLI. For example, the following commands could be typed into the CLI to run the one scheme on the Iris dataset using a basic train and test process. (Note that the commands would be typed on one line into the CLI.) While commands can be typed directly into the CLI, this technique is not particularly convenient and the experiments are not easy to modify. The Experimenter comes in two flavors, either with a simple interface that provides most of the functionality one needs for experiments, or with an interface with full access to the Experimenter‘s capabilities.

You can choose between those two with the Experiment Configuration Mode radio buttons:

- Simple
- Advanced

Both setups allow you to setup standard experiments that are run locally on a single machine, or remote experiments, which are distributed between several hosts. The distribution of experiments cuts down the time the experiments will take until completion, but on the other hand the setup takes more time. The next section covers the standard experiments (both, simple and advanced), followed by the remote experiments and finally the analyzing of the results.

III. Knowledge Flow

Introduction

The Knowledge Flow provides an alternative to the Explorer as a graphical front end to WEKA‘s core algorithms. The Knowledge Flow presents a data-flow inspired interface to WEKA. The user can select WEKA components from a palette place them on a layout canvas and connect them together in order to form a knowledge flow for processing and analyzing data. At present, all of WEKA‘s classifiers, filters, clusterers, associates, loaders and savers are available in the Knowledge Flow along with some extra tools.

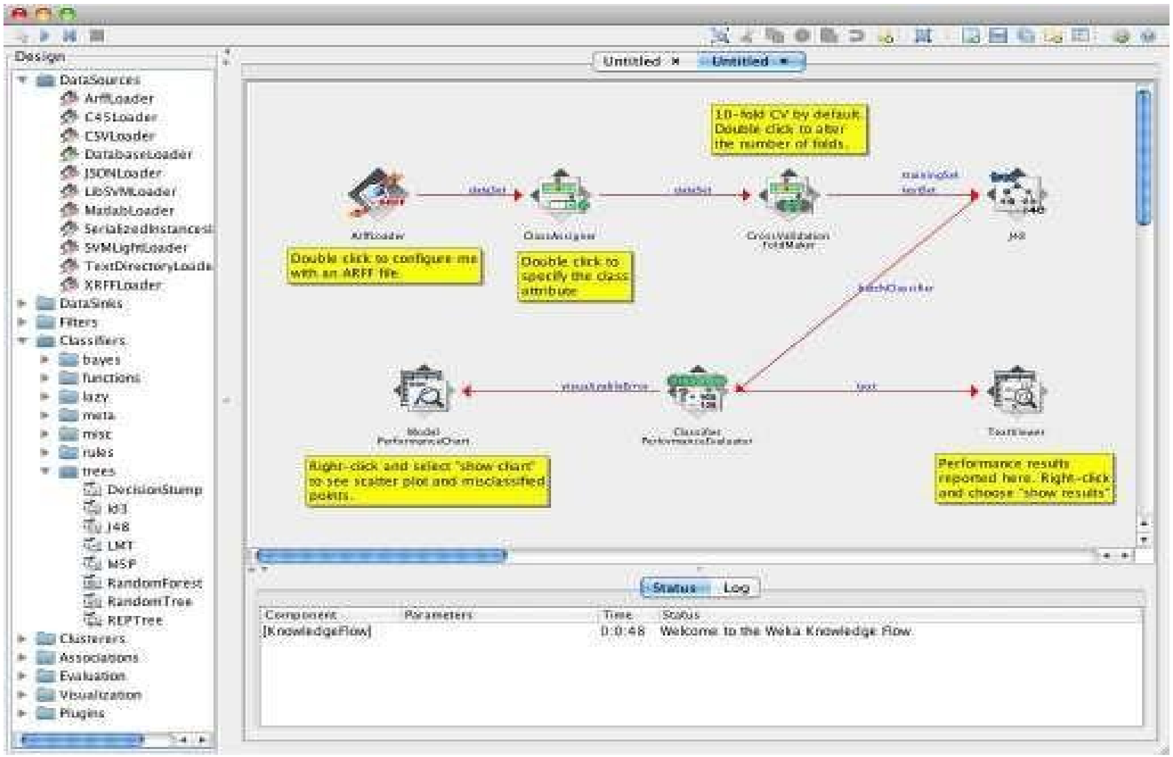

Fig 3: Knowledge Flow

The Knowledge Flow can handle data either incrementally or in batches (the Explorer handles batch data only). Of course learning from data incrementally requires a classifier that can be updated on an instance by instance basis. Currently in WEKA there are ten classifiers that can handle data incrementally.

The Knowledge Flow offers the following features:

- **intuitive** data flow style layout
- **process** data in batches or incrementally
- **process multiple batches** or streams in parallel (each separate flow executes in its own thread)
- **process multiple streams sequentially** via a user-specified order of execution
- **chain filters** together
- **view models** produced by classifiers for each fold in a cross validation

- **visualize performance** of incremental classifiers during processing (Scrolling plots of classification accuracy, RMS error, predictions etc.)

- **Plug-in** –perspectives‖ that add major new functionality (e.g. 3D data visualization, time series forecasting environment etc.)

IV. Simple CLI

The Simple CLI provides full access to all Weka classes, i.e., classifiers, filters, clusterers, etc., but without the hassle of the CLASSPATH (it facilitates the one, with which Weka was started). It offers a simple Weka shell with separated command line and output.

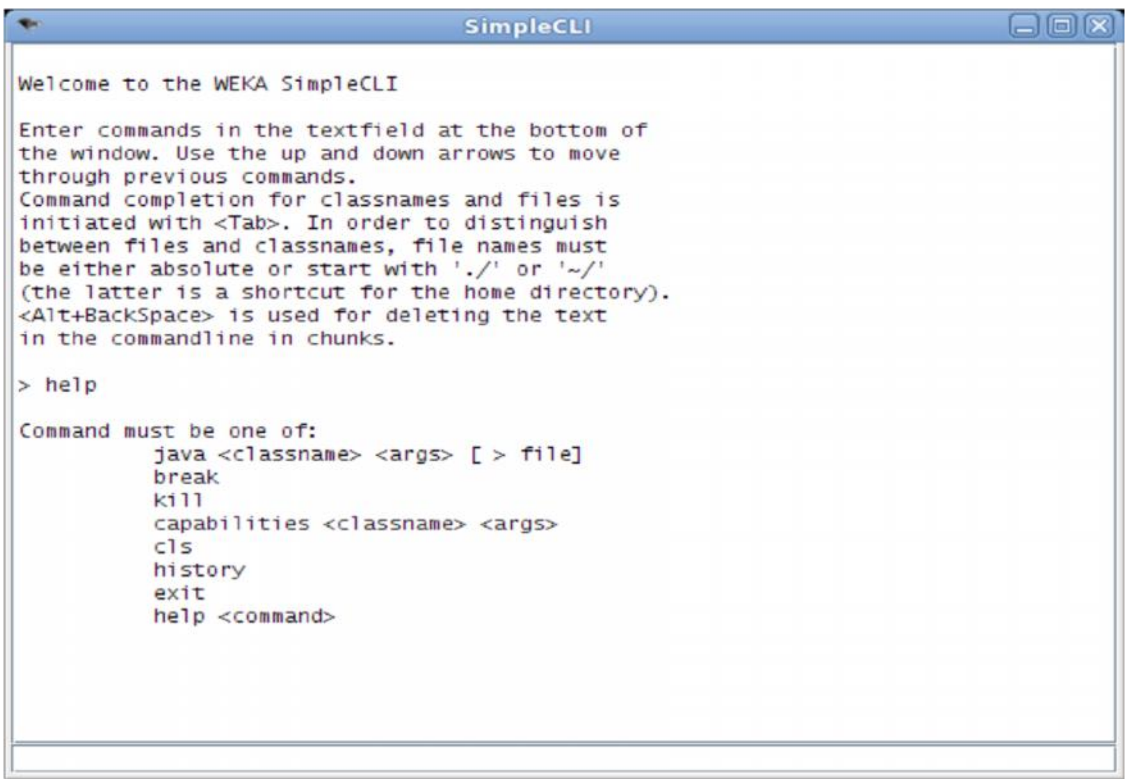

Fig 4: Simple CLI

Commands :

The following commands are available in the Simple CLI:

- java <class name> [<args>] invokes a java class with the given arguments (if any)
- break: stops the current thread, e.g., a running classifier, in a friendly manner kill stops the current thread in an unfriendly fashion
- cls :clears the output area
- capabilities <class name> [<args>] lists the capabilities of the specified class, e.g., for a classifier with its option: capabilities weka.classifiers.meta.Bagging -W weka.classifiers.trees.Id3
- exit :exits the Simple CLI
- Help: [<command>] provides an overview of the available commands if without a command name as argument, otherwise more help on the specified command.

Invocation In order to invoke a Weka class, one has only to prefix the class with‖java‖. This command tells the Simple CLI to load a class and execute it with any given parameters. E.g., the J48 classifier can be invoked on the iris dataset with the following command:

java weka.classifiers.trees.J48 -t c:/temp/iris.arff

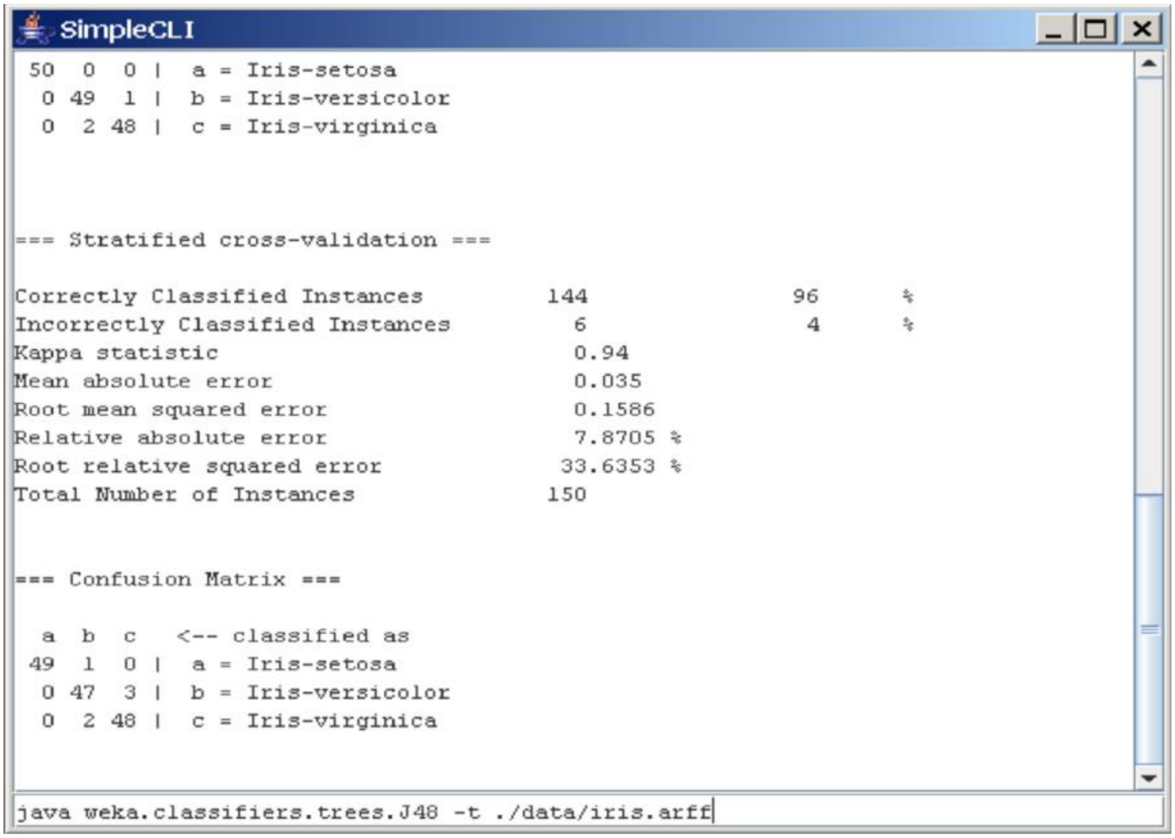

Fig 5: Simple CLI

3.2Command redirection

Starting with this version of Weka one can perform a basic redirection.

Java weka.classifiers.trees.J48 -t test.arff > j48.txt

Note: the > must be preceded and followed by a space, otherwise it is not recognized as redirection, but part of another parameter.

4. A STANDARD OPERATING PROCEDURE – SOP:

a) Explanation on today‘s experiment by the concerned faculty using OHP/PPT covering the following aspects:
25min

1) Name of the experiment/Aim
2) Software/Hardware required
3) Commands with suitable Options
4) Test Data
    1) Valid data sets
    2) Limiting value sets
    3) Invalid data sets

b) Writing of shell programs by the students 25 min.

c) Compiling and execution of the program

Writing of the experiment in the Observation Book:

The students will write the today‘s experiment in the Observation book as per the following format:

a) Name of the experiment/Aim
b) Software/Hardware required
c) Commands with suitable Options
d) Shell Programs/System call using C-Programs
e) Test Data
    a. Valid data sets
    b. Limiting value sets
    c. Invalid data sets
f) Results for different data sets
g) Viva-Voce Questions and Answers
h) Errors observed (if any) during compilation/execution
i) Signature of the Faculty

4.B GUIDELINES TO STUDENTS IN LAB

<u>Students are advised to maintain discipline and follow the guidelines given below:</u>

- Keep all your bags in the racks and carry the observation book and record book.
- Mobile phones/pen drives/ CDs are not allowed in the labs.
- Maintain proper dress code along with ID Card
- Occupy the computers allotted to you and maintain the discipline.
- Student must submit the record with the last week experiment details and observation book with the brief of the present experiment.
- Read the write up of the experiment given in the manual.
- Students must use the equipment with care. Any damage is caused student is punishable
- After completion of every experiment, the observation notes to be shown to the lab in -charge and after correction the record must be updated and submit to the lab in charge for correction.
- **Lab marks are given on Continuous Evaluation Basis as per JNTU guidelines**
- If any student is absent for any lab, they need to be complete the same experiment in the free time before attending next lab session.

<u>Steps to perform experiments in the lab by the student</u>

Step1: Students have to write the Date, aim, Software and Hardware requirements for the scheduled experiment in the observation book.

Step2: Students have to listen and understand the experiment explained by the faculty and note down the important points in the observation book.

Step3: Students need to write procedure/algorithm in the observation book.

Step4: Analyze and Develop/implement the logic of the program by the student in respective platform

Step5: After approval of logic of the experiment by the faculty then the experiment has to be executed on the system.

Step6: After successful execution, the results have to be recorded in the observation book and shown to the lab in charge faculty..

Step7: Students need to attend the Viva-Voce on that experiment and write the same in the observation book.

Step8: Update the completed experiment in the record and submit to the concerned faculty in-charge.

Instructions to maintain the record

- Before staring of the first lab session students must buy the record book and bring the same to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost, inform on the same day to the faculty in charge and submit the new record within 2 days for correction.
- If record is not submitted in time or record is not written properly, the record evaluation marks (5M) will be reduced accordingly.

Awarding the marks for day to day evaluation:

Total marks for day to day evaluation: 15 Marks (as per JNTUH).

Breakup for 15 Marks:

Record	5 Marks
Exp setup/program written and execution	5 Marks
Result and Viva-Voce	5 Marks

Allocation of Marks for Lab Internal Examinations:

Total marks for lab internal Examination: 25 Marks (as per JNTUH).

Break up for 25 Marks:

Average of day to day evaluation marks: 15 Marks

Lab Internal Mid examination: 10 Marks

Allocation of Marks for Lab External Examinations:

Total marks for External lab Examinations: 50 Marks as per JNTUH.

5. List of Lab Experiments as per JNTU

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 can not 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 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 a number of ways.
2. Books: Find some training manuals for loan officers or perhaps a suitable textbook on finance. Translate this knowledge from text from 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 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 data set. Consisting of **1000** actual cases collected in Germany.

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!) There are 20 attributes used in judging a loan applicant (ie., 7 Numerical attributes and 13 Categorical or Nominal attributes). The goal is to classify the applicant into one of two categories. Good or Bad.

6. JNTU SYLLABUS

DATA MINING LAB

B.Tech. IV Year I Sem.	**L**	**T**	**P**	**C**
Course Code: CS703PC	**0**	**0**	**3**	**2**

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

Experiment 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.

Experiment 3:
One type of model that you can create is a Decision tree . train a Decision tree using the complete data set as the training data. Report the model obtained after training.

Experiment 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 can not get 100% training accuracy?

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

Experiment 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 accuracy increase/decrease? Why?

Experiment 7:

Check to see if the data shows a bias against "foreign workers" or "personal-status". One way to do this is to remove these attributes from the data set and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. Did removing these attributes have any significantly effect? Discuss.

Experiment 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. 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.)

Experiment 9:

Sometimes, The cost of rejecting an applicant who actually has good credit might be higher than accepting an applicant who has bad credit. Instead of counting the misclassification equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. By using a cost matrix in weak. Train your decision tree and report the Decision Tree and cross validation results. Are they significantly different from results obtained in problem 6.

Experiment 10:

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

Experiment 11:

You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning. Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross validation and report the Decision Trees you obtain? Also Report your accuracy using the pruned model does your Accuracy increase?

Experiment 12:

How can you convert a Decision Tree into "if-then-else rules". Make up your own small Decision Tree consisting 2-3 levels and convert 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 data set? One R 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, one R.

***

7. LIST OF ADDITIONAL EXPERIMENTS FOR THE SEMESTER

DATA WAREHOUSING AND DATAMINING LAB

S. No	Name of the experiment
1	1. Perform cluster analysis on German credit data set using partition clustering algorithm
2	Perform cluster analysis on German credit data set using hierarchal clustering algorithm

8. Content of Lab Experiments

Experiment 1

Aim: List all the categorical (or nominal) attributes and the real valued attributes separately.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Algorithm / Procedure:

1. For each attribute of Germen data set identify type of data and define data type, either numeric or string.
    a. If attribute is string type, find the values of attribute.
    b. If the value is discrete, define attribute as nominal or categorical attribute. Otherwise, define attribute as string.
2. Repeat step 1 until end of all attributes in data set.
3. Display list of categorical and numerical valued attributes.

Output:

German Credit data Attributes:-
1. Checking status
2. Duration
3. Credit history
4. Purpose
5. Credit amount
6. Savings status
7. Employment duration
8. Installment rate
9. Personal status
10. Debitors
11. Residence since
12. Property
14. Installment plans
15. Housing

16. Existing credits
17. Job
18. Num_dependents
19. Telephone
20. Foreign worker

Categorical or Nomianal attributes:-
1. Checking status
2. Credit history
3. Purpose
4. Savings status
5. Employment
6. Personal status
7. Debtors
8. Property
9. Installment plans
10. Housing
11. Job
12. Telephone
13. Foreign worker

Real valued attributes:-
1. duration
2. credit amount
3. credit amount
4. residence
5. age
6. existing credits
7. num_dependents

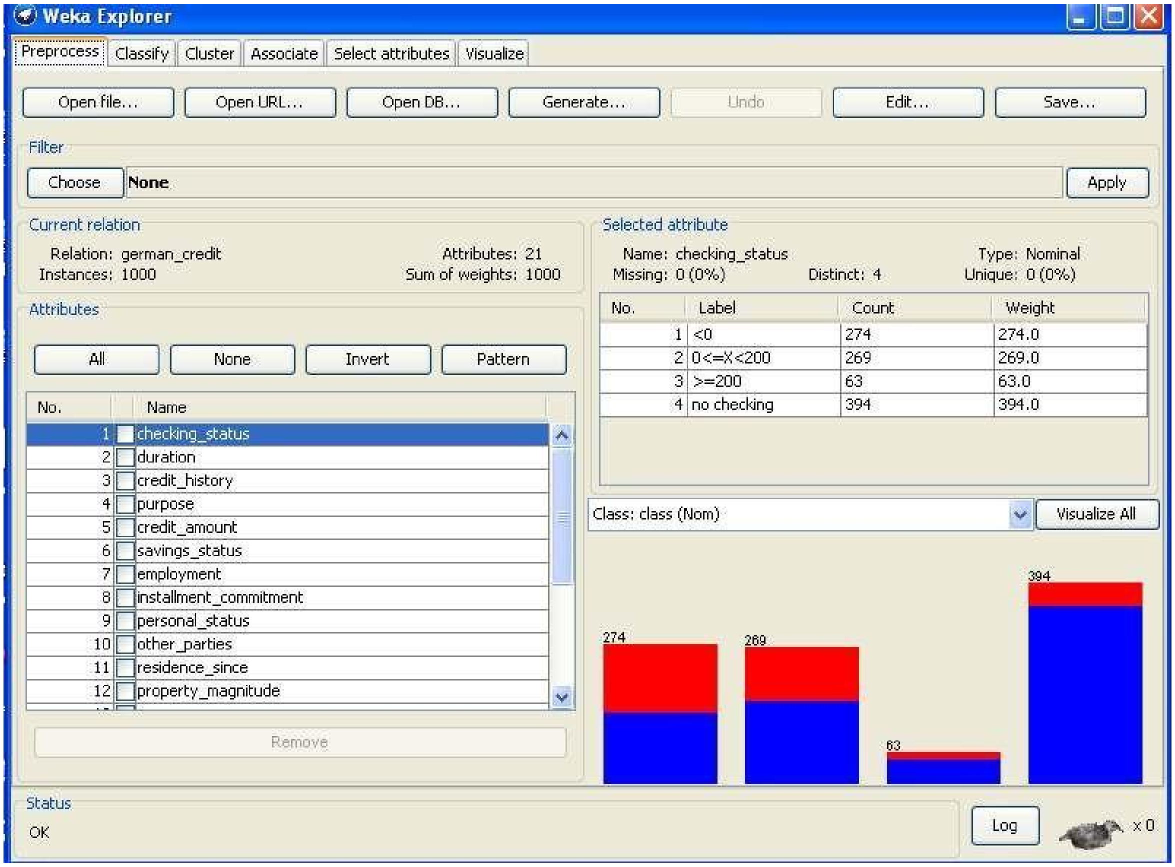

Fig 6: Nominal attributes

Viva Questions:

1. Define database?
2. Define Database management systems?
3. What is Database Management system?
4. Define types of Data in Database?
5. What is numeric or interval scaled variables?

***

Experiment 2

Aim: 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.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Algorithm / Procedure:

1. For each attribute of Germen data set,
   a. Analyze the values of attribute.
   b. Find attribute, which can be used for making decision on credit.
2. Form sample rules on selected attribute to classify the customer as good.
3. Form the sample rules on selected attribute to classify the customer as bad.

Output: According to me the following attributes may be crucial in making the credit risk assessment.

1. Credit_history
2. Employment
3. Property_magnitude
4. job
5. duration
6. crdit_amount
7. installment
8. existing credit

Basing on the above attributes, we can make a decision whether to give credit or not.

JRIP Rules:

1.If (checking_status = <0) and (job = skilled) => class=bad (172.0/76.0)

2.If(checking_status = 0<=X<200) and (duration >= 24) and (savings_status = <100) => class=bad (61.0/19.0)

=> class=good (767.0/162.0)

Viva Questions:

1. What is ARFF?
2. What is Tag present in ARFF file?
3. What is the purpose of Header tag?
4. How to declare the nominal attributes?
5. How to declare the attributes?

***

Experiment 3

Aim: One type of model that you can create is a Decision tree. Train a Decision tree using the complete data set as the training data. Report the model obtained after training.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code/Algorithm:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select training data set from test data option.
3. Start classification.

Output: The following model obtained after training the data set.

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: german_credit

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) ===

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 = none: bad (2.0)
|   |   |   |   |   own_telephone = yes: good (4.0)
|   |   |   |   property_magnitude = car: good (2.0/1.0)
|   |   |   |   property_magnitude = no known property: bad (3.0)
|   |   |   existing_credits > 1: good (14.0)
|   |   duration > 11
|   |   |   job = unemp/unskilled non res: bad (5.0/1.0)
|   |   |   job = unskilled resident
|   |   |   |   purpose = new car
|   |   |   |   |   own_telephone = none: bad (10.0/2.0)
|   |   |   |   |   own_telephone = yes: good (2.0)
|   |   |   |   purpose = used car: bad (1.0)
|   |   |   |   purpose = furniture/equipment
|   |   |   |   |   employment = unemployed: good (0.0)
```

```
| | | | | employment = <1: bad (3.0)
| | | | | employment = 1<=X<4: good (4.0)
| | | | | employment = 4<=X<7: good (1.0)
| | | | | employment = >=7: good (2.0)
| | | | purpose = radio/tv
| | | | | existing_credits <= 1: bad (10.0/3.0)
| | | | | existing_credits > 1: good (2.0)
| | | | purpose = domestic appliance: bad (1.0)
| | | | purpose = repairs: bad (1.0)
| | | | purpose = education: bad (1.0)
| | | | purpose = vacation: bad (0.0)
| | | | purpose = retraining: good (1.0)
| | | | purpose = business: good (3.0)
| | | | purpose = other: good (1.0)
| | | job = skilled
| | | | other_parties = none
| | | | | duration <= 30
| | | | | | savings_status = <100
| | | | | | | credit_history = no credits/all paid: bad (8.0/1.0)
| | | | | | | credit_history = all paid: bad (6.0)
| | | | | | | credit_history = existing paid
| | | | | | | | own_telephone = none
| | | | | | | | | existing_credits <= 1
| | | | | | | | | | property_magnitude = real estate
```

```
| | | | | | | | | | | age <= 26: bad (5.0)
| | | | | | | | | | | age > 26: good (2.0)
| | | | | | | | | | property_magnitude = life insurance: bad (7.0/2.0)
| | | | | | | | | | property_magnitude = car
| | | | | | | | | | | credit_amount <= 1386: bad (3.0)
| | | | | | | | | | | credit_amount > 1386: good (11.0/1.0)
| | | | | | | | | | property_magnitude = no known property: good (2.0)
| | | | | | | | | existing_credits > 1: bad (3.0)
| | | | | | | | own_telephone = yes: bad (5.0)
| | | | | | | credit_history = delayed previously: bad (4.0)
| | | | | | | credit_history = critical/other existing credit: good (14.0/4.0)
| | | | | | savings_status = 100<=X<500
| | | | | | | credit_history = no credits/all paid: good (0.0)
| | | | | | | credit_history = all paid: good (1.0)
| | | | | | | credit_history = existing paid: bad (3.0)
| | | | | | | credit_history = delayed previously: good (0.0)
| | | | | | | credit_history = critical/other existing credit: good (2.0)
| | | | | | savings_status = 500<=X<1000: good (4.0/1.0)
| | | | | | savings_status = >=1000: good (4.0)
| | | | | | savings_status = no known savings
| | | | | | | existing_credits <= 1
| | | | | | | | own_telephone = none: bad (9.0/1.0)
| | | | | | | | own_telephone = yes: good (4.0/1.0)
| | | | | | | existing_credits > 1: good (2.0)
```

```
|  |  |  |  |  duration > 30: bad (30.0/3.0)
|  |  |  |  other_parties = co applicant: bad (7.0/1.0)
|  |  |  |  other_parties = guarantor: good (12.0/3.0)
|  |  |  job = high qualif/self emp/mgmt: good (30.0/8.0)
|  foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
|  credit_amount <= 9857
|  |  savings_status = <100
|  |  |  other_parties = none
|  |  |  |  duration <= 42
|  |  |  |  |  personal_status = male div/sep: bad (8.0/2.0)
|  |  |  |  |  personal_status = female div/dep/mar
|  |  |  |  |  |  purpose = new car: bad (5.0/1.0)
|  |  |  |  |  |  purpose = used car: bad (1.0)
|  |  |  |  |  |  purpose = furniture/equipment
|  |  |  |  |  |  |  duration <= 10: bad (3.0)
|  |  |  |  |  |  |  duration > 10
|  |  |  |  |  |  |  |  duration <= 21: good (6.0/1.0)
|  |  |  |  |  |  |  |  duration > 21: bad (2.0)
|  |  |  |  |  |  purpose = radio/tv: good (8.0/2.0)
|  |  |  |  |  |  purpose = domestic appliance: good (0.0)
|  |  |  |  |  |  purpose = repairs: good (1.0)
|  |  |  |  |  |  purpose = education: good (4.0/2.0)
|  |  |  |  |  |  purpose = vacation: good (0.0)
```

```
|  |  |  |  |  |  purpose = retraining: good (0.0)
|  |  |  |  |  |  purpose = business
|  |  |  |  |  |  |  residence_since <= 2: good (3.0)
|  |  |  |  |  |  |  residence_since > 2: bad (2.0)
|  |  |  |  |  |  purpose = other: good (0.0)
|  |  |  |  |  personal_status = male single: good (52.0/15.0)
|  |  |  |  |  personal_status = male mar/wid
|  |  |  |  |  |  duration <= 10: good (6.0)
|  |  |  |  |  |  duration > 10: bad (10.0/3.0)
|  |  |  |  |  personal_status = female single: good (0.0)
|  |  |  |  duration > 42: bad (7.0)
|  |  |  other_parties = co applicant: good (2.0)
|  |  |  other_parties = guarantor
|  |  |  |  purpose = new car: bad (2.0)
|  |  |  |  purpose = used car: good (0.0)
|  |  |  |  purpose = furniture/equipment: good (0.0)
|  |  |  |  purpose = radio/tv: good (18.0/1.0)
|  |  |  |  purpose = domestic appliance: good (0.0)
|  |  |  |  purpose = repairs: good (0.0)
|  |  |  |  purpose = education: good (0.0)
|  |  |  |  purpose = vacation: good (0.0)
|  |  |  |  purpose = retraining: good (0.0)
|  |  |  |  purpose = business: good (0.0)
|  |  |  |  purpose = other: good (0.0)
```

```
| | savings_status = 100<=X<500
| | | purpose = new car: bad (15.0/5.0)
| | | purpose = used car: good (3.0)
| | | purpose = furniture/equipment: bad (4.0/1.0)
| | | purpose = radio/tv: bad (8.0/2.0)
| | | purpose = domestic appliance: good (0.0)
| | | purpose = repairs: good (2.0)
| | | purpose = education: good (0.0)
| | | purpose = vacation: good (0.0)
| | | purpose = retraining: good (0.0)
| | | purpose = business
| | | | housing = rent
| | | | | existing_credits <= 1: good (2.0)
| | | | | existing_credits > 1: bad (2.0)
| | | | housing = own: good (6.0)
| | | | housing = for free: bad (1.0)
| | | purpose = other: good (1.0)
| | savings_status = 500<=X<1000: good (11.0/3.0)
| | savings_status = >=1000: good (13.0/3.0)
| | savings_status = no known savings: good (41.0/5.0)
| credit_amount > 9857: bad (20.0/3.0)
checking_status = >=200: good (63.0/14.0)
checking_status = no checking: good (394.0/46.0)
Number of Leaves : 103
```

Size of the tree : 140

Time taken to build model: 0.05 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	855	85.5 %
Incorrectly Classified Instances	145	14.5 %
Kappa statistic	0.6251	
Mean absolute error	0.2312	
Root mean squared error	0.34	
Relative absolute error	55.0377 %	
Root relative squared error	74.2015 %	
Coverage of cases (0.95 level)	100 %	
Mean rel. region size (0.95 level)	93.3 %	
Total Number of Instances	1000	

=== 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	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 669  31 |   a = good
 114 186 |   b = bad
```

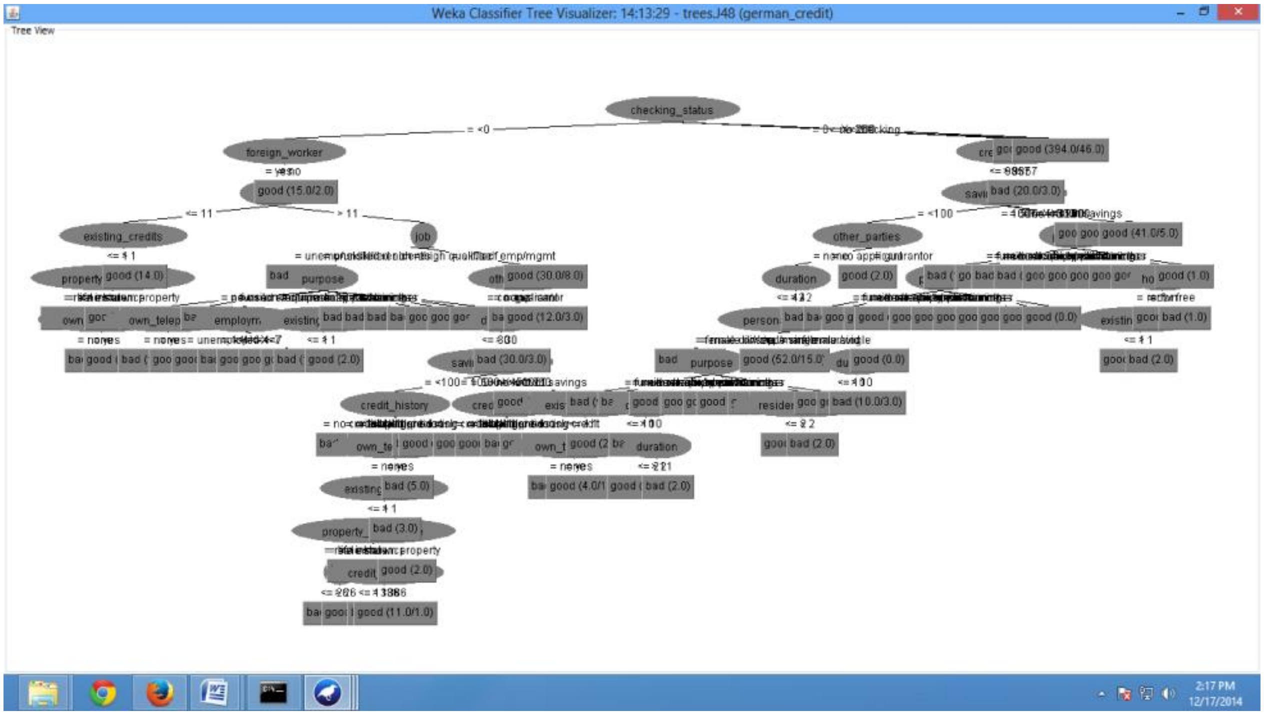

Fig 7:Decision tree

Viva Questions:

1. Define Data mining ?

2. What are the steps in KDD ?

3. List some of the Data Mining task ?

4. Which are the software tool used for Data mining ,available in market ?

5. Define WEKA ?

***

Experiment 4

Aim: 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 can not get 100% training accuracy?

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select training data set from test data option.
3. Start classification.

Output: The following model obtained after training the data set.

In the above model we trained complete dataset and we classified credit good/bad for each

of the examples in the dataset.
For example:
IF
purpose=vacation THEN
credit=bad
ELSE
purpose=business THEN
Credit=good
In this way we classified each of the examples in the dataset.
We classified 85.5% of examples correctly and the remaining 14.5% of examples are incorrectly classified. We can‘t get 100% training accuracy because out of the 20 attributes, we have some unnecessary attributes which are also been analyzed and trained.
Due to this the accuracy is affected and hence we can‘t get 100% training accuracy.

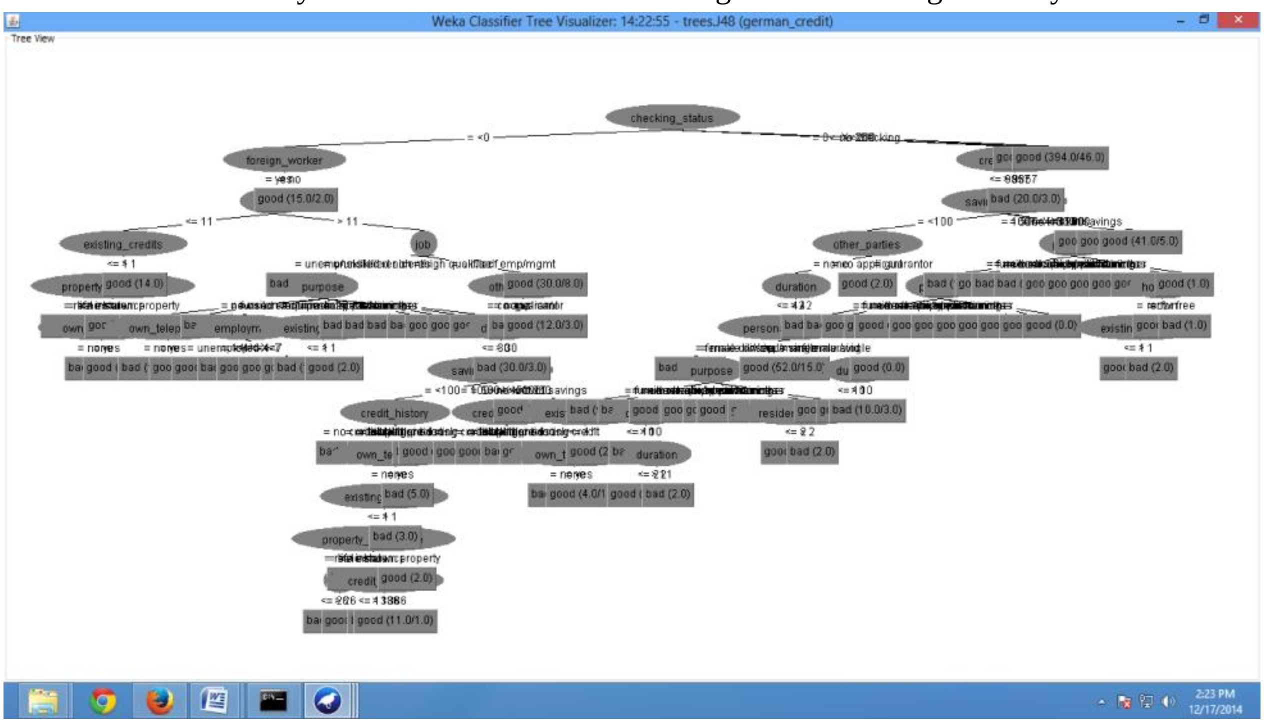

Fig 8.Dataset Decision tree

Conclusion: If we used our above model trained on the complete dataset and classified credit as good/bad for each of the examples in that dataset. We can not get 100% training accuracy only **85.5%** of examples, we can classify correctly.

Viva Questions:

1. Define data classification?
2. What are the steps in data classification?
3. Define the accuracy of classification ?
4. Give sum of the algorithms used in classification
5. Give decision tree induction algorithms in WEKA

***

Experiment 5

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

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select training data set from test data option.
3. Start classification.

Output:

1. According to the rules, for the maximum accuracy, we have to take 2/3 of the dataset as training set and the remaining 1/3 as test set. But here in the above model we have taken complete
dataset as training set which results only 85.5% accuracy.

2. This is done for the analyzing and training of the unnecessary attributes which does not make a crucial role in credit risk assessment. And by this complexity is increasing and finally it leads to
the minimum accuracy.
3. If some part of the dataset is used as a training set and the remaining as test set then it leads to the accurate results and the time for computation will be less.
4. This is why, we prefer not to take complete dataset as training set.

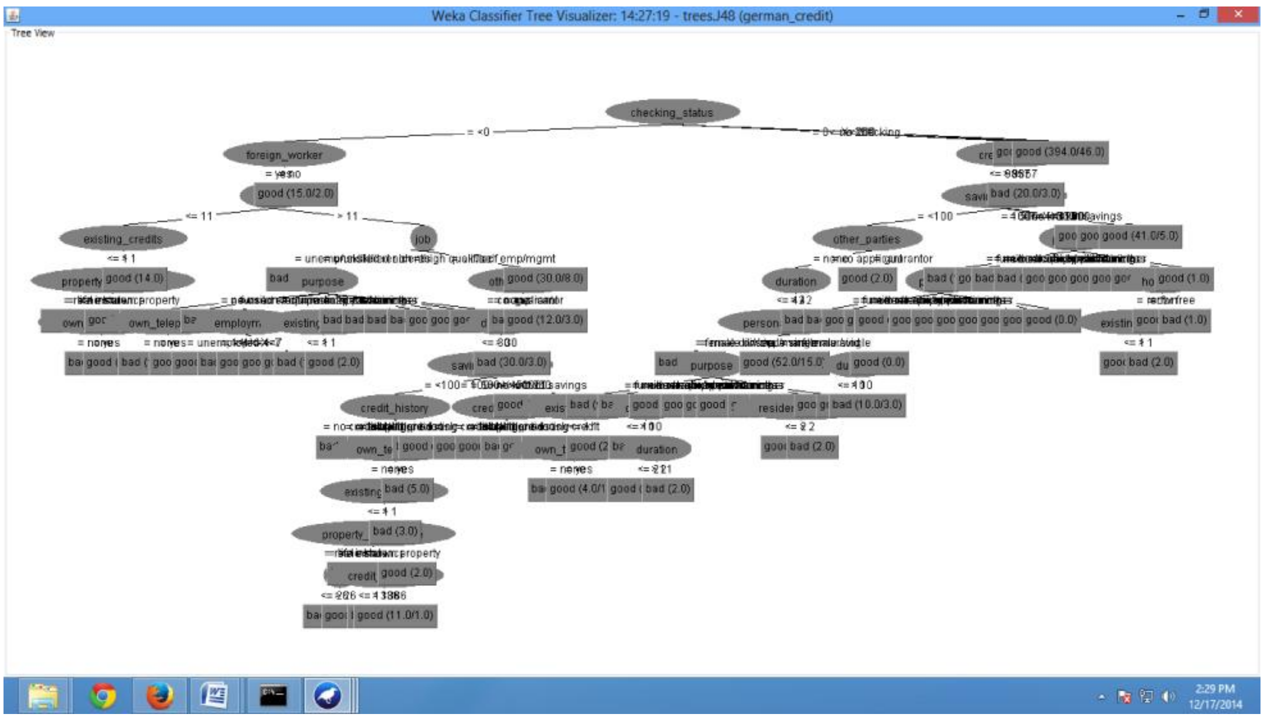

Fig 9. Decision tree by using J48

Conclusion: It is not good idea by using 100% training data set.

Viva Questions:

1. Define training data set?
2. What is test data set?
3. What are the test options?
4. Define Data warehousing?

***

Experiment 6

Aim: 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 accuracy increase/decrease? Why?

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Cross-Validation Definition: The classifier is evaluated by cross validation using the number of folds that are entered in the folds text field.

Cross validation:-

In k-fold cross-validation, the initial data are randomly portioned into ‗k' mutually exclusive subsets or folds D1, D2, D3... Dk. Each of approximately equal size. Training and testing is performed ‗k' times. In iteration I, partition Di is reserved as the test set and the remaining partitions are collectively used to train the model. That is in the first iteration subsets D2, D3… Dk collectively serve as the training set in order to obtain as first model. Which is tested on Di? The second trained on the subsets D1, D3. . . Dk and test on the D2 and so on….

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select cross validataion with fold size 2, 5 and 10 from test data option.
3. Start classification.

In Classify Tab, Select cross-validation option and folds size is 2 then Press Start Button, next time change as folds size is 5 then press start, and next time change as folds size is 10 then press start.

Output: The following model obtained after training the data set.

Fold Size – 2 output:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: german_credit

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: 2-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 = none: bad (2.0)
| | | | | own_telephone = yes: good (4.0)
| | | | property_magnitude = car: good (2.0/1.0)
| | | | property_magnitude = no known property: bad (3.0)
| | | existing_credits > 1: good (14.0)
| | duration > 11
| | | job = unemp/unskilled non res: bad (5.0/1.0)
| | | job = unskilled resident
| | | | purpose = new car
| | | | | own_telephone = none: bad (10.0/2.0)
| | | | | own_telephone = yes: good (2.0)
| | | | purpose = used car: bad (1.0)
| | | | purpose = furniture/equipment
| | | | | employment = unemployed: good (0.0)
| | | | | employment = <1: bad (3.0)
| | | | | employment = 1<=X<4: good (4.0)
| | | | | employment = 4<=X<7: good (1.0)
| | | | | employment = >=7: good (2.0)
| | | | purpose = radio/tv
| | | | | existing_credits <= 1: bad (10.0/3.0)
| | | | | existing_credits > 1: good (2.0)
| | | | purpose = domestic appliance: bad (1.0)
| | | | purpose = repairs: bad (1.0)
| | | | purpose = education: bad (1.0)
```

```
| | | | purpose = vacation: bad (0.0)
| | | | purpose = retraining: good (1.0)
| | | | purpose = business: good (3.0)
| | | | purpose = other: good (1.0)
| | | job = skilled
| | | | other_parties = none
| | | | | duration <= 30
| | | | | | savings_status = <100
| | | | | | | credit_history = no credits/all paid: bad (8.0/1.0)
| | | | | | | credit_history = all paid: bad (6.0)
| | | | | | | credit_history = existing paid
| | | | | | | | own_telephone = none
| | | | | | | | | existing_credits <= 1
| | | | | | | | | | property_magnitude = real estate
| | | | | | | | | | | age <= 26: bad (5.0)
| | | | | | | | | | | age > 26: good (2.0)
| | | | | | | | | | property_magnitude = life insurance: bad (7.0/2.0)
| | | | | | | | | | property_magnitude = car
| | | | | | | | | | | credit_amount <= 1386: bad (3.0)
| | | | | | | | | | | credit_amount > 1386: good (11.0/1.0)
| | | | | | | | | | property_magnitude = no known property: good (2.0)
| | | | | | | | | existing_credits > 1: bad (3.0)
| | | | | | | | own_telephone = yes: bad (5.0)
| | | | | | | credit_history = delayed previously: bad (4.0)
```

```
| | | | | | | credit_history = critical/other existing credit: good (14.0/4.0)
| | | | | | savings_status = 100<=X<500
| | | | | | | credit_history = no credits/all paid: good (0.0)
| | | | | | | credit_history = all paid: good (1.0)
| | | | | | | credit_history = existing paid: bad (3.0)
| | | | | | | credit_history = delayed previously: good (0.0)
| | | | | | | credit_history = critical/other existing credit: good (2.0)
| | | | | | savings_status = 500<=X<1000: good (4.0/1.0)
| | | | | | savings_status = >=1000: good (4.0)
| | | | | | savings_status = no known savings
| | | | | | | existing_credits <= 1
| | | | | | | | own_telephone = none: bad (9.0/1.0)
| | | | | | | | own_telephone = yes: good (4.0/1.0)
| | | | | | | existing_credits > 1: good (2.0)
| | | | | duration > 30: bad (30.0/3.0)
| | | | other_parties = co applicant: bad (7.0/1.0)
| | | | other_parties = guarantor: good (12.0/3.0)
| | | job = high qualif/self emp/mgmt: good (30.0/8.0)
| foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
| credit_amount <= 9857
| | savings_status = <100
| | | other_parties = none
| | | | duration <= 42
```

```
| | | | | personal_status = male div/sep: bad (8.0/2.0)
| | | | | personal_status = female div/dep/mar
| | | | | | purpose = new car: bad (5.0/1.0)
| | | | | | purpose = used car: bad (1.0)
| | | | | | purpose = furniture/equipment
| | | | | | | duration <= 10: bad (3.0)
| | | | | | | duration > 10
| | | | | | | | duration <= 21: good (6.0/1.0)
| | | | | | | | duration > 21: bad (2.0)
| | | | | | purpose = radio/tv: good (8.0/2.0)
| | | | | | purpose = domestic appliance: good (0.0)
| | | | | | purpose = repairs: good (1.0)
| | | | | | purpose = education: good (4.0/2.0)
| | | | | | purpose = vacation: good (0.0)
| | | | | | purpose = retraining: good (0.0)
| | | | | | purpose = business
| | | | | | | residence_since <= 2: good (3.0)
| | | | | | | residence_since > 2: bad (2.0)
| | | | | | purpose = other: good (0.0)
| | | | | personal_status = male single: good (52.0/15.0)
| | | | | personal_status = male mar/wid
| | | | | | duration <= 10: good (6.0)
| | | | | | duration > 10: bad (10.0/3.0)
| | | | | personal_status = female single: good (0.0)
```

```
| | | | duration > 42: bad (7.0)
| | | other_parties = co applicant: good (2.0)
| | | other_parties = guarantor
| | | | purpose = new car: bad (2.0)
| | | | purpose = used car: good (0.0)
| | | | purpose = furniture/equipment: good (0.0)
| | | | purpose = radio/tv: good (18.0/1.0)
| | | | purpose = domestic appliance: good (0.0)
| | | | purpose = repairs: good (0.0)
| | | | purpose = education: good (0.0)
| | | | purpose = vacation: good (0.0)
| | | | purpose = retraining: good (0.0)
| | | | purpose = business: good (0.0)
| | | | purpose = other: good (0.0)
| | savings_status = 100<=X<500
| | | purpose = new car: bad (15.0/5.0)
| | | purpose = used car: good (3.0)
| | | purpose = furniture/equipment: bad (4.0/1.0)
| | | purpose = radio/tv: bad (8.0/2.0)
| | | purpose = domestic appliance: good (0.0)
| | | purpose = repairs: good (2.0)
| | | purpose = education: good (0.0)
| | | purpose = vacation: good (0.0)
| | | purpose = retraining: good (0.0)
```

| | | purpose = business

| | | | housing = rent

| | | | | existing_credits <= 1: good (2.0)

| | | | | existing_credits > 1: bad (2.0)

| | | | housing = own: good (6.0)

| | | | housing = for free: bad (1.0)

| | | purpose = other: good (1.0)

| | savings_status = 500<=X<1000: good (11.0/3.0)

| | savings_status = >=1000: good (13.0/3.0)

| | savings_status = no known savings: good (41.0/5.0)

| credit_amount > 9857: bad (20.0/3.0)

checking_status = >=200: good (63.0/14.0)

checking_status = no checking: good (394.0/46.0)

Number of Leaves : 103

Size of the tree : 140

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 721 72.1 %

Incorrectly Classified Instances 279 27.9 %

Kappa statistic 0.2443

Mean absolute error 0.3407

Root mean squared error 0.4669

Relative absolute error 81.0491 %

Root relative squared error 101.8806 %

Coverage of cases (0.95 level) 92.8 %

Mean rel. region size (0.95 level) 91.3 %

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.891	0.677	0.755	0.891	0.817	0.662	good
	0.323	0.109	0.561	0.323	0.41	0.662	bad
Weighted Avg.	0.721	0.506	0.696	0.721	0.695	0.662	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 624  76 |   a = good
 203  97 |   b = bad
```

Fold Size – 5 output:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: german_credit

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: 5-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 = none: bad (2.0)
|   |   |   |   |   own_telephone = yes: good (4.0)
|   |   |   |   property_magnitude = car: good (2.0/1.0)
|   |   |   |   property_magnitude = no known property: bad (3.0)
|   |   |   existing_credits > 1: good (14.0)
|   |   duration > 11
|   |   |   job = unemp/unskilled non res: bad (5.0/1.0)
|   |   |   job = unskilled resident
|   |   |   |   purpose = new car
|   |   |   |   |   own_telephone = none: bad (10.0/2.0)
|   |   |   |   |   own_telephone = yes: good (2.0)
|   |   |   |   purpose = used car: bad (1.0)
|   |   |   |   purpose = furniture/equipment
|   |   |   |   |   employment = unemployed: good (0.0)
|   |   |   |   |   employment = <1: bad (3.0)
|   |   |   |   |   employment = 1<=X<4: good (4.0)
|   |   |   |   |   employment = 4<=X<7: good (1.0)
|   |   |   |   |   employment = >=7: good (2.0)
|   |   |   |   purpose = radio/tv
```

```
| | | | | existing_credits <= 1: bad (10.0/3.0)
| | | | | existing_credits > 1: good (2.0)
| | | | purpose = domestic appliance: bad (1.0)
| | | | purpose = repairs: bad (1.0)
| | | | purpose = education: bad (1.0)
| | | | purpose = vacation: bad (0.0)
| | | | purpose = retraining: good (1.0)
| | | | purpose = business: good (3.0)
| | | | purpose = other: good (1.0)
| | | job = skilled
| | | | other_parties = none
| | | | | duration <= 30
| | | | | | savings_status = <100
| | | | | | | credit_history = no credits/all paid: bad (8.0/1.0)
| | | | | | | credit_history = all paid: bad (6.0)
| | | | | | | credit_history = existing paid
| | | | | | | | own_telephone = none
| | | | | | | | | existing_credits <= 1
| | | | | | | | | | property_magnitude = real estate
| | | | | | | | | | | age <= 26: bad (5.0)
| | | | | | | | | | | age > 26: good (2.0)
| | | | | | | | | | property_magnitude = life insurance: bad (7.0/2.0)
| | | | | | | | | | property_magnitude = car
| | | | | | | | | | | credit_amount <= 1386: bad (3.0)
```

```
| | | | | | | | | | | credit_amount > 1386: good (11.0/1.0)
| | | | | | | | | | property_magnitude = no known property: good (2.0)
| | | | | | | | | existing_credits > 1: bad (3.0)
| | | | | | | | own_telephone = yes: bad (5.0)
| | | | | | | credit_history = delayed previously: bad (4.0)
| | | | | | | credit_history = critical/other existing credit: good (14.0/4.0)
| | | | | | savings_status = 100<=X<500
| | | | | | | credit_history = no credits/all paid: good (0.0)
| | | | | | | credit_history = all paid: good (1.0)
| | | | | | | credit_history = existing paid: bad (3.0)
| | | | | | | credit_history = delayed previously: good (0.0)
| | | | | | | credit_history = critical/other existing credit: good (2.0)
| | | | | | savings_status = 500<=X<1000: good (4.0/1.0)
| | | | | | savings_status = >=1000: good (4.0)
| | | | | | savings_status = no known savings
| | | | | | | existing_credits <= 1
| | | | | | | | own_telephone = none: bad (9.0/1.0)
| | | | | | | | own_telephone = yes: good (4.0/1.0)
| | | | | | | existing_credits > 1: good (2.0)
| | | | | duration > 30: bad (30.0/3.0)
| | | | other_parties = co applicant: bad (7.0/1.0)
| | | | other_parties = guarantor: good (12.0/3.0)
| | | job = high qualif/self emp/mgmt: good (30.0/8.0)
| foreign_worker = no: good (15.0/2.0)
```

```
checking_status = 0<=X<200
|   credit_amount <= 9857
|   |   savings_status = <100
|   |   |   other_parties = none
|   |   |   |   duration <= 42
|   |   |   |   |   personal_status = male div/sep: bad (8.0/2.0)
|   |   |   |   |   personal_status = female div/dep/mar
|   |   |   |   |   |   purpose = new car: bad (5.0/1.0)
|   |   |   |   |   |   purpose = used car: bad (1.0)
|   |   |   |   |   |   purpose = furniture/equipment
|   |   |   |   |   |   |   duration <= 10: bad (3.0)
|   |   |   |   |   |   |   duration > 10
|   |   |   |   |   |   |   |   duration <= 21: good (6.0/1.0)
|   |   |   |   |   |   |   |   duration > 21: bad (2.0)
|   |   |   |   |   |   purpose = radio/tv: good (8.0/2.0)
|   |   |   |   |   |   purpose = domestic appliance: good (0.0)
|   |   |   |   |   |   purpose = repairs: good (1.0)
|   |   |   |   |   |   purpose = education: good (4.0/2.0)
|   |   |   |   |   |   purpose = vacation: good (0.0)
|   |   |   |   |   |   purpose = retraining: good (0.0)
|   |   |   |   |   |   purpose = business
|   |   |   |   |   |   |   residence_since <= 2: good (3.0)
|   |   |   |   |   |   |   residence_since > 2: bad (2.0)
|   |   |   |   |   |   purpose = other: good (0.0)
```

```
| | | | | personal_status = male single: good (52.0/15.0)
| | | | | personal_status = male mar/wid
| | | | | | duration <= 10: good (6.0)
| | | | | | duration > 10: bad (10.0/3.0)
| | | | | personal_status = female single: good (0.0)
| | | | duration > 42: bad (7.0)
| | | other_parties = co applicant: good (2.0)
| | | other_parties = guarantor
| | | | purpose = new car: bad (2.0)
| | | | purpose = used car: good (0.0)
| | | | purpose = furniture/equipment: good (0.0)
| | | | purpose = radio/tv: good (18.0/1.0)
| | | | purpose = domestic appliance: good (0.0)
| | | | purpose = repairs: good (0.0)
| | | | purpose = education: good (0.0)
| | | | purpose = vacation: good (0.0)
| | | | purpose = retraining: good (0.0)
| | | | purpose = business: good (0.0)
| | | | purpose = other: good (0.0)
| | savings_status = 100<=X<500
| | | purpose = new car: bad (15.0/5.0)
| | | purpose = used car: good (3.0)
| | | purpose = furniture/equipment: bad (4.0/1.0)
| | | purpose = radio/tv: bad (8.0/2.0)
```

| | | purpose = domestic appliance: good (0.0)

| | | purpose = repairs: good (2.0)

| | | purpose = education: good (0.0)

| | | purpose = vacation: good (0.0)

| | | purpose = retraining: good (0.0)

| | | purpose = business

| | | | housing = rent

| | | | | existing_credits <= 1: good (2.0)

| | | | | existing_credits > 1: bad (2.0)

| | | | housing = own: good (6.0)

| | | | housing = for free: bad (1.0)

| | | purpose = other: good (1.0)

| | savings_status = 500<=X<1000: good (11.0/3.0)

| | savings_status = >=1000: good (13.0/3.0)

| | savings_status = no known savings: good (41.0/5.0)

| credit_amount > 9857: bad (20.0/3.0)

checking_status = >=200: good (63.0/14.0)

checking_status = no checking: good (394.0/46.0)

Number of Leaves : 103

Size of the tree : 140

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 733 73.3 %

Incorrectly Classified Instances 267 26.7 %

Kappa statistic 0.3264

Mean absolute error 0.3293

Root mean squared error 0.4579

Relative absolute error 78.3705 %

Root relative squared error 99.914 %

Coverage of cases (0.95 level) 94.7 %

Mean rel. region size (0.95 level) 93 %

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.851	0.543	0.785	0.851	0.817	0.685	good
	0.457	0.149	0.568	0.457	0.506	0.685	bad
Weighted Avg.	0.733	0.425	0.72	0.733	0.724	0.685	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 596 104 |   a = good
 163 137 |   b = bad
```

Fold Size – 10 output:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: german_credit

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: 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 = none: bad (2.0)

| | | | | own_telephone = yes: good (4.0)

| | | | property_magnitude = car: good (2.0/1.0)

| | | | property_magnitude = no known property: bad (3.0)

| | | existing_credits > 1: good (14.0)

| | duration > 11

| | | job = unemp/unskilled non res: bad (5.0/1.0)

| | | job = unskilled resident

| | | | purpose = new car

| | | | | own_telephone = none: bad (10.0/2.0)

| | | | | own_telephone = yes: good (2.0)

| | | | purpose = used car: bad (1.0)

| | | | purpose = furniture/equipment

| | | | | employment = unemployed: good (0.0)

```
| | | | | employment = <1: bad (3.0)
| | | | | employment = 1<=X<4: good (4.0)
| | | | | employment = 4<=X<7: good (1.0)
| | | | | employment = >=7: good (2.0)
| | | | purpose = radio/tv
| | | | | existing_credits <= 1: bad (10.0/3.0)
| | | | | existing_credits > 1: good (2.0)
| | | | purpose = domestic appliance: bad (1.0)
| | | | purpose = repairs: bad (1.0)
| | | | purpose = education: bad (1.0)
| | | | purpose = vacation: bad (0.0)
| | | | purpose = retraining: good (1.0)
| | | | purpose = business: good (3.0)
| | | | purpose = other: good (1.0)
| | | job = skilled
| | | | other_parties = none
| | | | | duration <= 30
| | | | | | savings_status = <100
| | | | | | | credit_history = no credits/all paid: bad (8.0/1.0)
| | | | | | | credit_history = all paid: bad (6.0)
| | | | | | | credit_history = existing paid
| | | | | | | | own_telephone = none
| | | | | | | | | existing_credits <= 1
| | | | | | | | | | property_magnitude = real estate
```

```
| | | | | | | | | | | age <= 26: bad (5.0)
| | | | | | | | | | | age > 26: good (2.0)
| | | | | | | | | | property_magnitude = life insurance: bad (7.0/2.0)
| | | | | | | | | | property_magnitude = car
| | | | | | | | | | | credit_amount <= 1386: bad (3.0)
| | | | | | | | | | | credit_amount > 1386: good (11.0/1.0)
| | | | | | | | | | property_magnitude = no known property: good (2.0)
| | | | | | | | | existing_credits > 1: bad (3.0)
| | | | | | | | own_telephone = yes: bad (5.0)
| | | | | | | credit_history = delayed previously: bad (4.0)
| | | | | | | credit_history = critical/other existing credit: good (14.0/4.0)
| | | | | | savings_status = 100<=X<500
| | | | | | | credit_history = no credits/all paid: good (0.0)
| | | | | | | credit_history = all paid: good (1.0)
| | | | | | | credit_history = existing paid: bad (3.0)
| | | | | | | credit_history = delayed previously: good (0.0)
| | | | | | | credit_history = critical/other existing credit: good (2.0)
| | | | | | savings_status = 500<=X<1000: good (4.0/1.0)
| | | | | | savings_status = >=1000: good (4.0)
| | | | | | savings_status = no known savings
| | | | | | | existing_credits <= 1
| | | | | | | | own_telephone = none: bad (9.0/1.0)
| | | | | | | | own_telephone = yes: good (4.0/1.0)
| | | | | | | existing_credits > 1: good (2.0)
```

```
|   |   |   |   |   duration > 30: bad (30.0/3.0)
|   |   |   |   other_parties = co applicant: bad (7.0/1.0)
|   |   |   |   other_parties = guarantor: good (12.0/3.0)
|   |   |   job = high qualif/self emp/mgmt: good (30.0/8.0)
|   foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
|   credit_amount <= 9857
|   |   savings_status = <100
|   |   |   other_parties = none
|   |   |   |   duration <= 42
|   |   |   |   |   personal_status = male div/sep: bad (8.0/2.0)
|   |   |   |   |   personal_status = female div/dep/mar
|   |   |   |   |   |   purpose = new car: bad (5.0/1.0)
|   |   |   |   |   |   purpose = used car: bad (1.0)
|   |   |   |   |   |   purpose = furniture/equipment
|   |   |   |   |   |   |   duration <= 10: bad (3.0)
|   |   |   |   |   |   |   duration > 10
|   |   |   |   |   |   |   |   duration <= 21: good (6.0/1.0)
|   |   |   |   |   |   |   |   duration > 21: bad (2.0)
|   |   |   |   |   |   purpose = radio/tv: good (8.0/2.0)
|   |   |   |   |   |   purpose = domestic appliance: good (0.0)
|   |   |   |   |   |   purpose = repairs: good (1.0)
|   |   |   |   |   |   purpose = education: good (4.0/2.0)
|   |   |   |   |   |   purpose = vacation: good (0.0)
```

```
| | | | | | purpose = retraining: good (0.0)
| | | | | | purpose = business
| | | | | | | residence_since <= 2: good (3.0)
| | | | | | | residence_since > 2: bad (2.0)
| | | | | | purpose = other: good (0.0)
| | | | | personal_status = male single: good (52.0/15.0)
| | | | | personal_status = male mar/wid
| | | | | | duration <= 10: good (6.0)
| | | | | | duration > 10: bad (10.0/3.0)
| | | | | personal_status = female single: good (0.0)
| | | | duration > 42: bad (7.0)
| | | other_parties = co applicant: good (2.0)
| | | other_parties = guarantor
| | | | purpose = new car: bad (2.0)
| | | | purpose = used car: good (0.0)
| | | | purpose = furniture/equipment: good (0.0)
| | | | purpose = radio/tv: good (18.0/1.0)
| | | | purpose = domestic appliance: good (0.0)
| | | | purpose = repairs: good (0.0)
| | | | purpose = education: good (0.0)
| | | | purpose = vacation: good (0.0)
| | | | purpose = retraining: good (0.0)
| | | | purpose = business: good (0.0)
| | | | purpose = other: good (0.0)
```

```
|  |  savings_status = 100<=X<500
|  |  |  purpose = new car: bad (15.0/5.0)
|  |  |  purpose = used car: good (3.0)
|  |  |  purpose = furniture/equipment: bad (4.0/1.0)
|  |  |  purpose = radio/tv: bad (8.0/2.0)
|  |  |  purpose = domestic appliance: good (0.0)
|  |  |  purpose = repairs: good (2.0)
|  |  |  purpose = education: good (0.0)
|  |  |  purpose = vacation: good (0.0)
|  |  |  purpose = retraining: good (0.0)
|  |  |  purpose = business
|  |  |  |  housing = rent
|  |  |  |  |  existing_credits <= 1: good (2.0)
|  |  |  |  |  existing_credits > 1: bad (2.0)
|  |  |  |  housing = own: good (6.0)
|  |  |  |  housing = for free: bad (1.0)
|  |  |  purpose = other: good (1.0)
|  |  savings_status = 500<=X<1000: good (11.0/3.0)
|  |  savings_status = >=1000: good (13.0/3.0)
|  |  savings_status = no known savings: good (41.0/5.0)
|  credit_amount > 9857: bad (20.0/3.0)
checking_status = >=200: good (63.0/14.0)
checking_status = no checking: good (394.0/46.0)
Number of Leaves  :    103
```

Size of the tree : 140

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 705 70.5 %

Incorrectly Classified Instances 295 29.5 %

Kappa statistic 0.2467

Mean absolute error 0.3467

Root mean squared error 0.4796

Relative absolute error 82.5233 %

Root relative squared error 104.6565 %

Coverage of cases (0.95 level) 92.8 %

Mean rel. region size (0.95 level) 91.7 %

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.84	0.61	0.763	0.84	0.799	0.639	good
	0.39	0.16	0.511	0.39	0.442	0.639	bad
Weighted Avg.	0.705	0.475	0.687	0.705	0.692	0.639	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 588 112 |   a = good
 183 117 |   b = bad
```

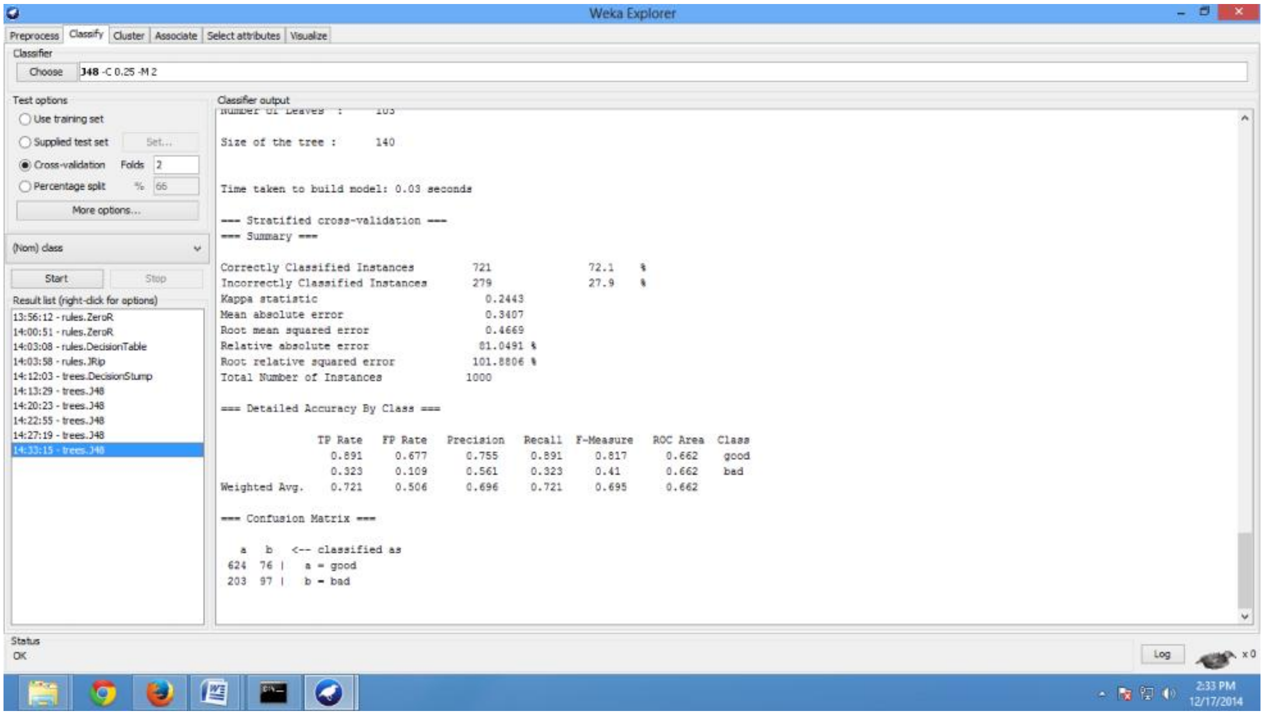

Fig 10.1 cross validation and fold size is 2

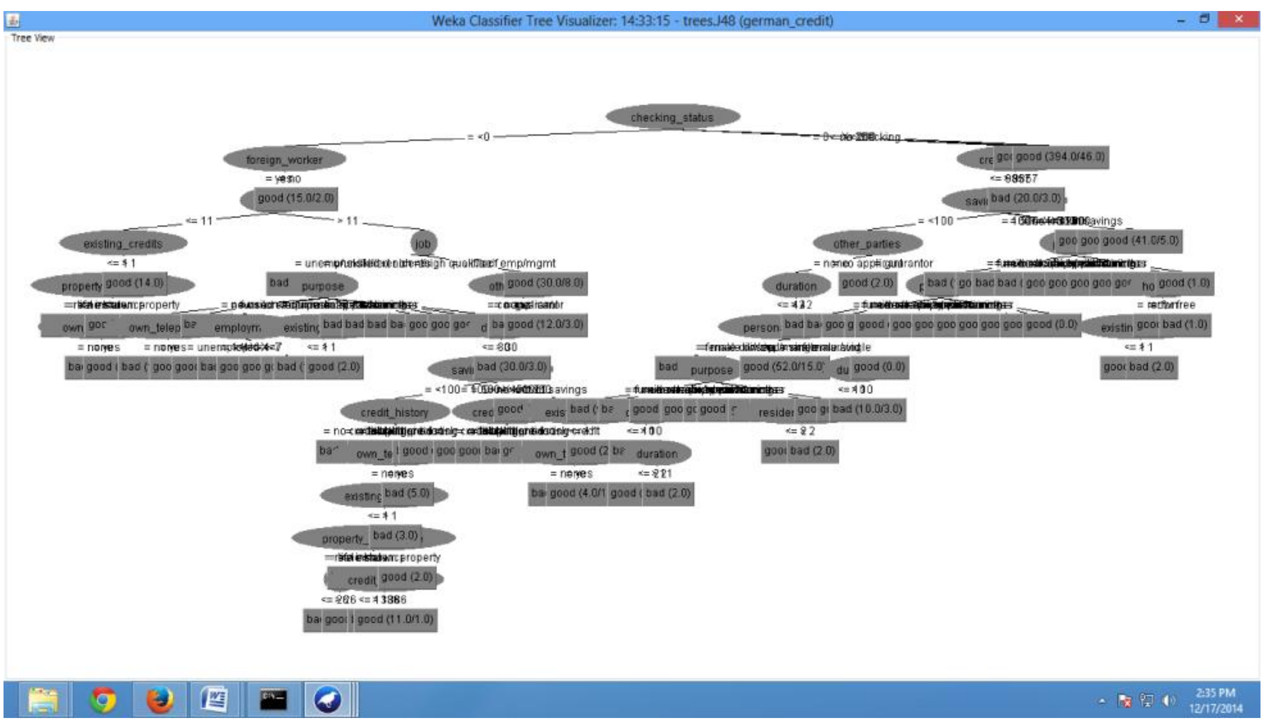

Fig 10.2 cross validation Tree.

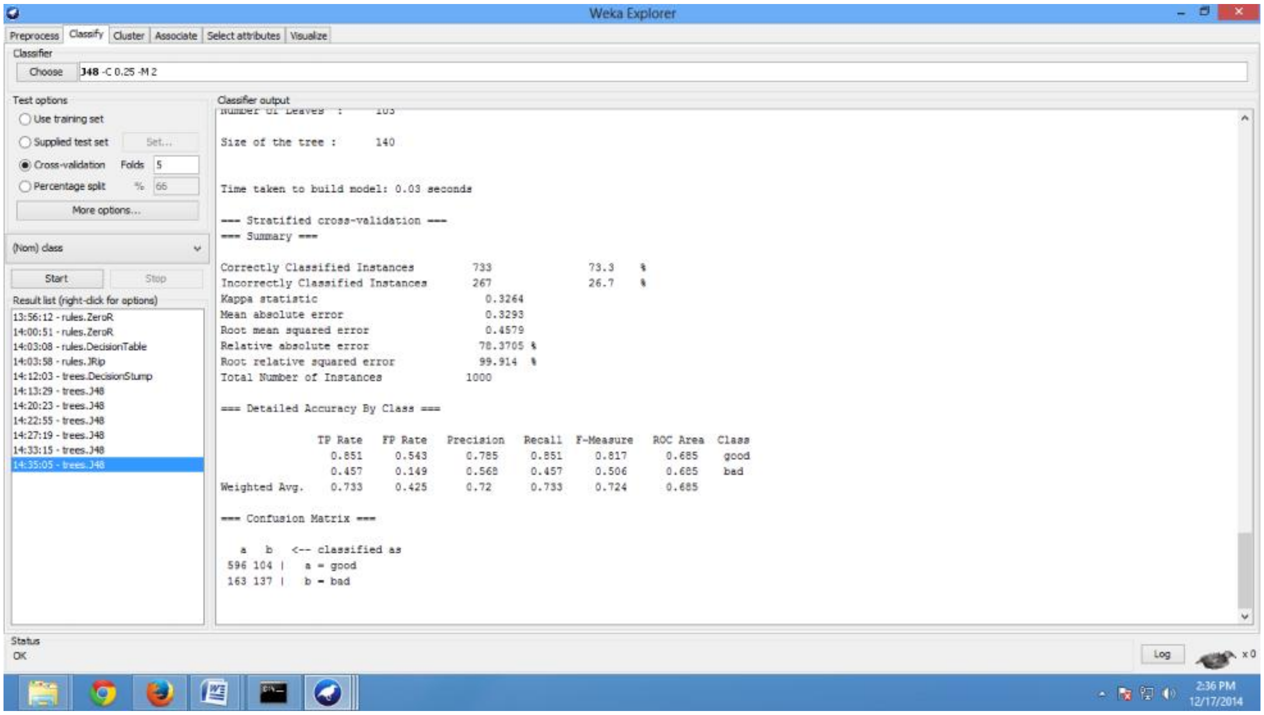

Fig 10.3 cross validation and fold size is 5

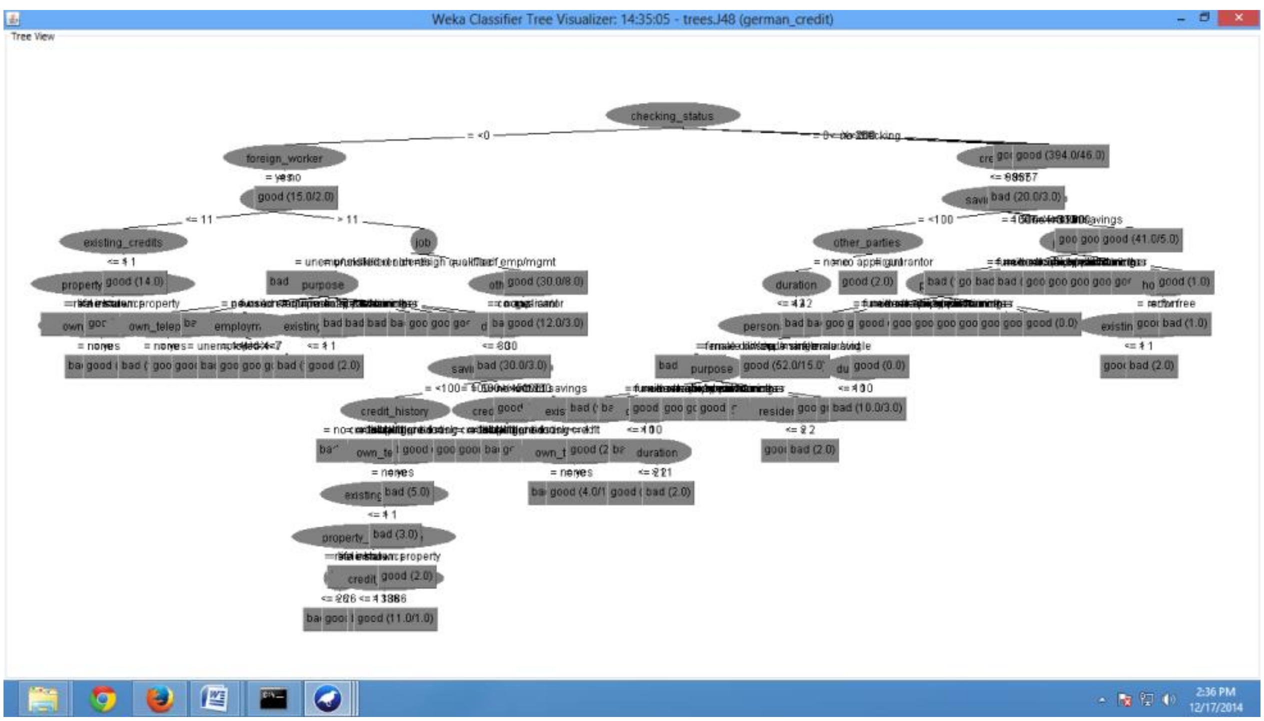

Fig 10.4 cross validation Tree.

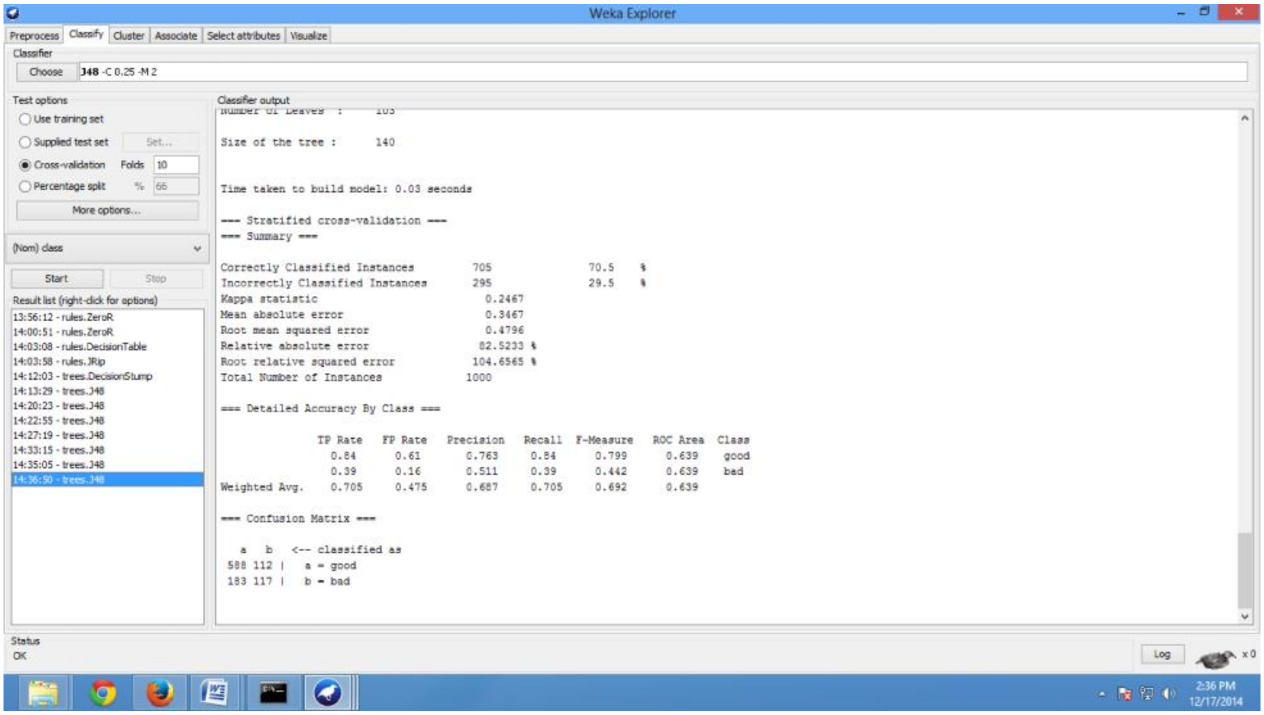

Fig 10.5 cross validation and fold size is 10

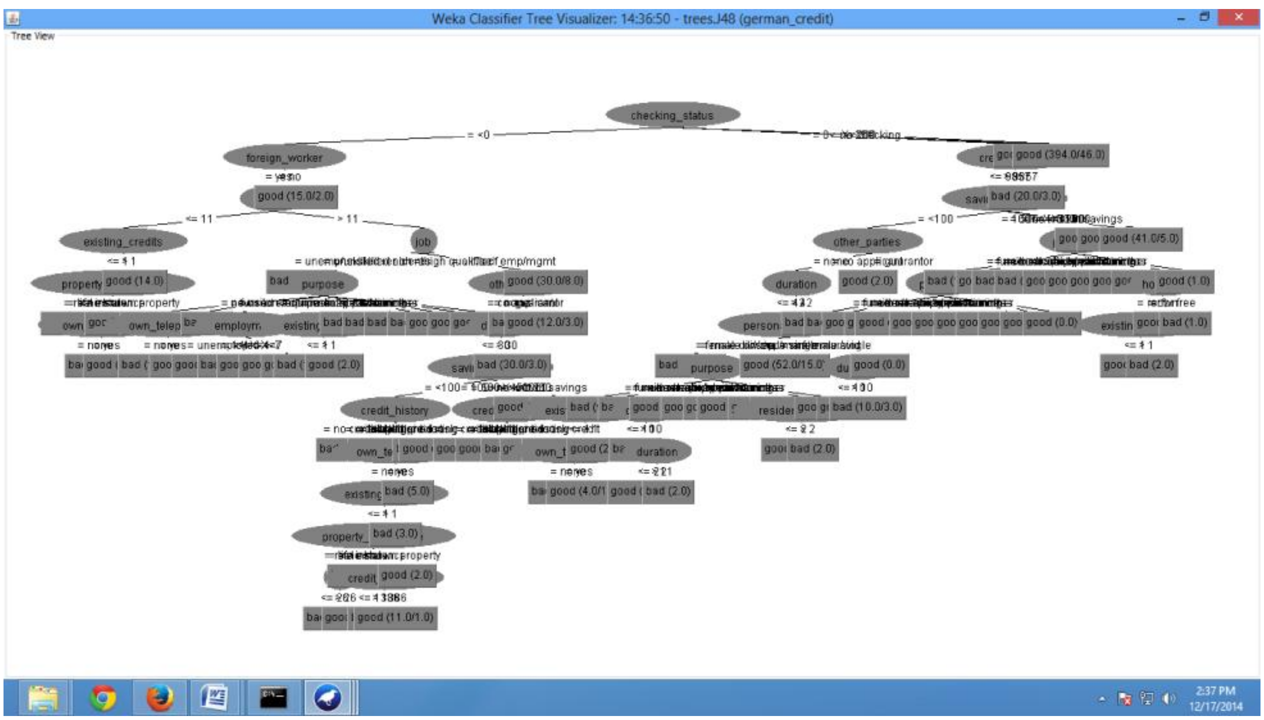

Fig 10.6 cross validation Tree.

Viva Questions:

1. what is cross validation?

2. how to evaluate the classifier accuracy?

3. What are methods of portions?

4. Define accuracy?

***

Experiment 7

Aim: Check to see if the data shows a bias against "foreign workers" or "personal-status". One way to do this is to remove these attributes from the data set and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. Did removing these attributes have any significantly effect? Discuss.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*

Find the normalized information gain ratio from splitting on *a*

3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Classification after removing –foreign worker‖ attribute.

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. In preprocessor, select –foreign worker‖ attribute from attribute list and remove.
3. Select classifier tab, choose J48 decision tree and select training data set from test data option.
4. Start classification.

Output: The following model obtained after training the data set.

After removing foreign worker

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	859	85.9	%
Incorrectly Classified Instances	141	14.1	%
Kappa statistic	0.6377		
Mean absolute error	0.2233		
Root mean squared error	0.3341		
Relative absolute error	53.1347 %		
Root relative squared error	72.9074 %		
Coverage of cases (0.95 level)	100 %		
Mean rel. region size (0.95 level)	91.9 %		
Total Number of Instances	1000		

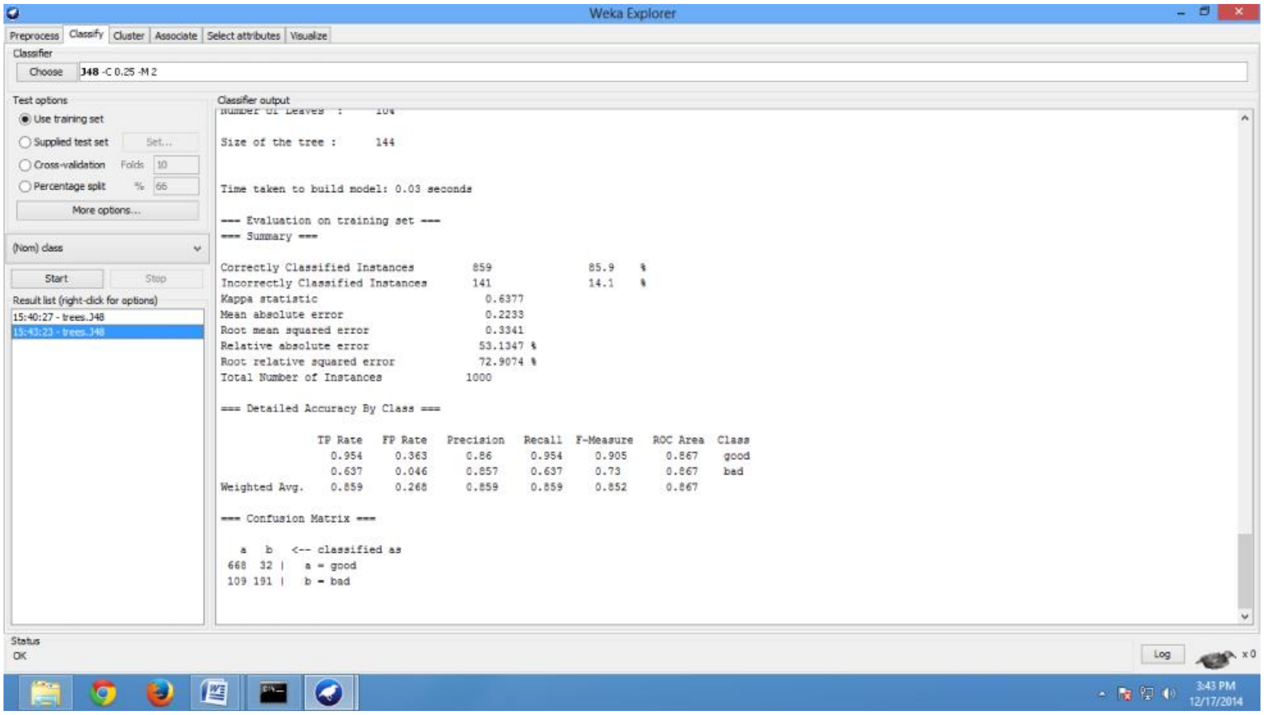

Fig 11.1 Removing “Foreign worker” attribute.

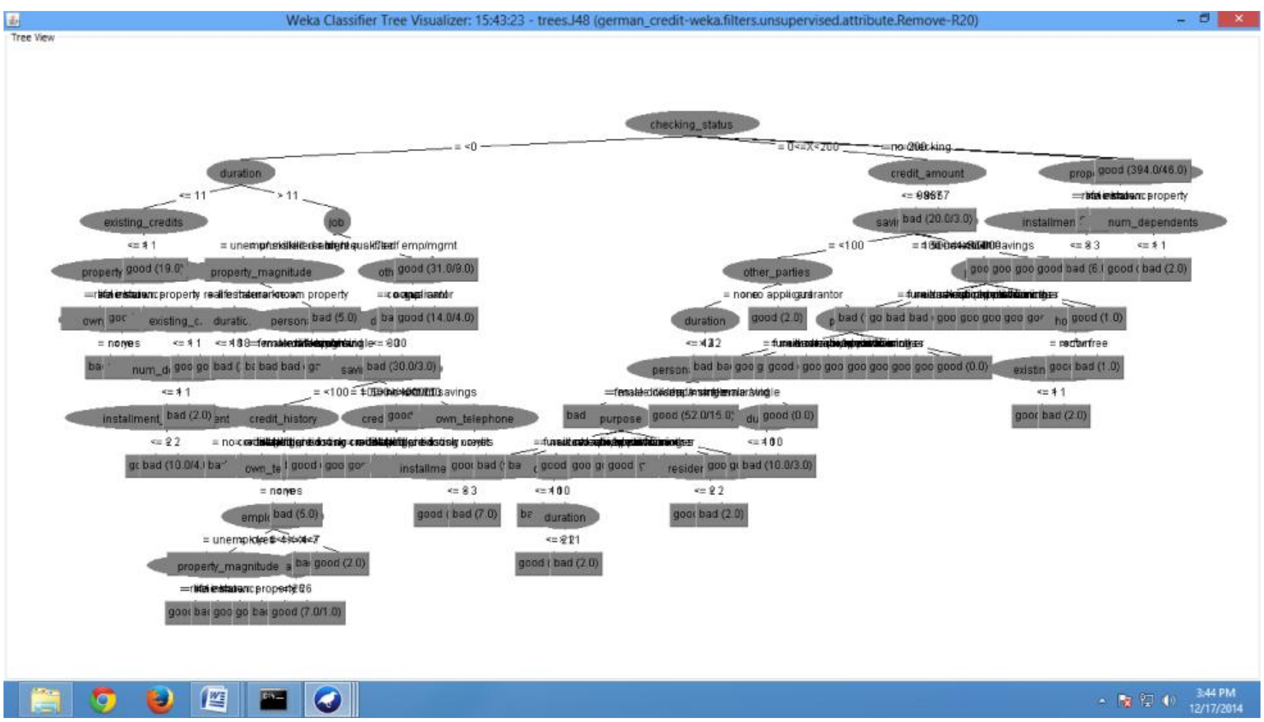

Fig 11.2 Foreign worker dataset

Procedure:

Classification after removing –personal status‖ attribute.

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. In preprocessor, select –personal status‖ attribute from attribute list and remove.
3. Select classifier tab, choose J48 decision tree and select training data set from test data option.
4. Start classification.

Output:

After removing personal status

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	866	86.6	%
Incorrectly Classified Instances	134	13.4	%
Kappa statistic	0.6582		
Mean absolute error	0.2162		
Root mean squared error	0.3288		
Relative absolute error	51.4483 %		
Root relative squared error	71.7411 %		
Coverage of cases (0.95 level)	100 %		
Mean rel. region size (0.95 level)	91.7 %		
Total Number of Instances	1000		

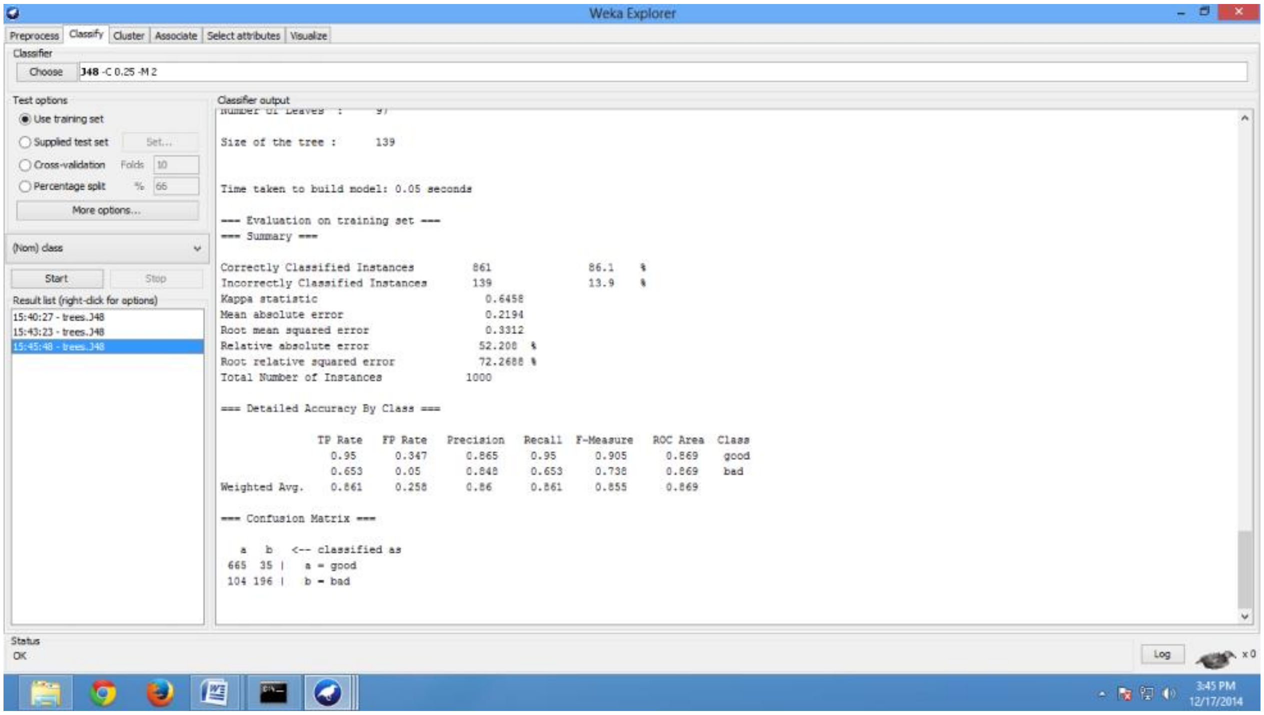

Fig 11.3 Removing "Personal status" attribute

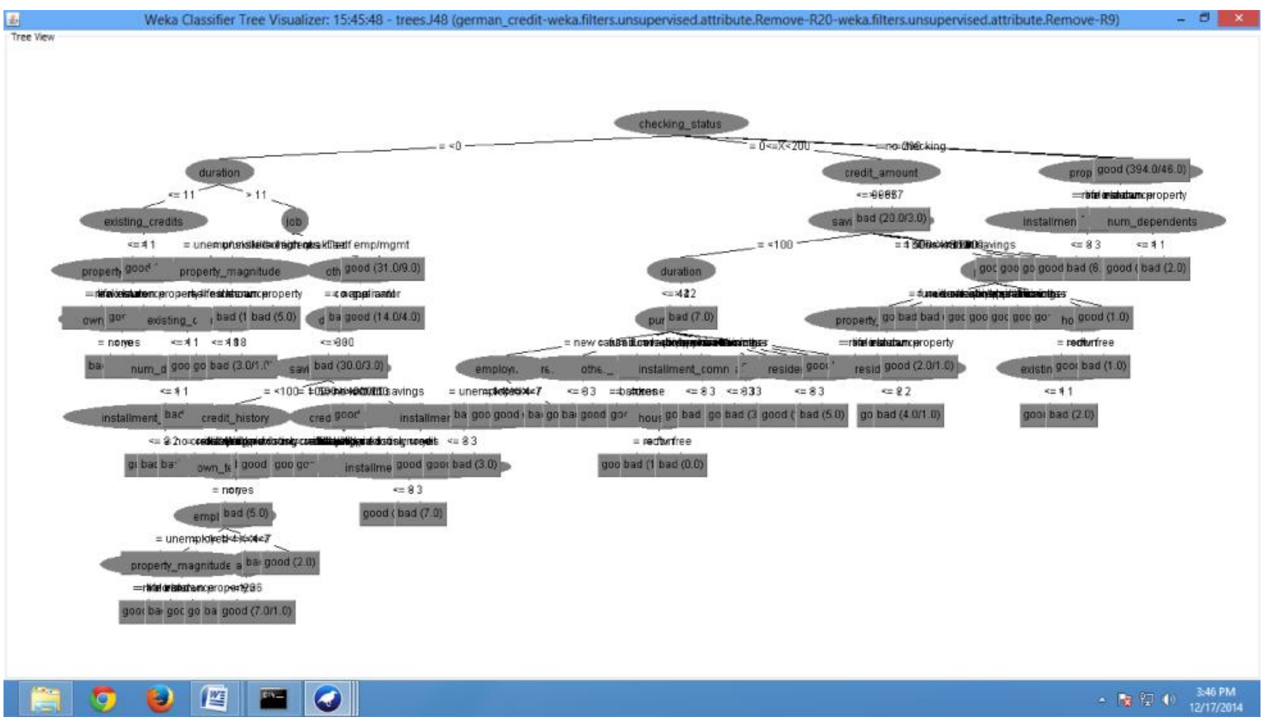

Fig 11.4 Personal status Dataset

Conclusion: With this observation we have seen, when –Foreign_worker –attribute is removed from the Dataset, the accuracy is decreased. So this attribute is important for classification.

VIVA QUESTIONS

1. What are the applications of data mining?
2. Define OLAP?
3. Define Cross-validation?
4. What is fold?
5. Define decision Tree?

***

Experiment 8

Aim: 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. 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.)

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Cross-Validation Definition: The classifier is evaluated by cross validation using the number of folds that are entered in the folds text field.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure: Classification after removing 2nd attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. In preprocessor, select 2nd attribute from attribute list and remove.
3. Select classifier tab, choose J48 decision tree and select training data set from test data option.

4. Start classification.

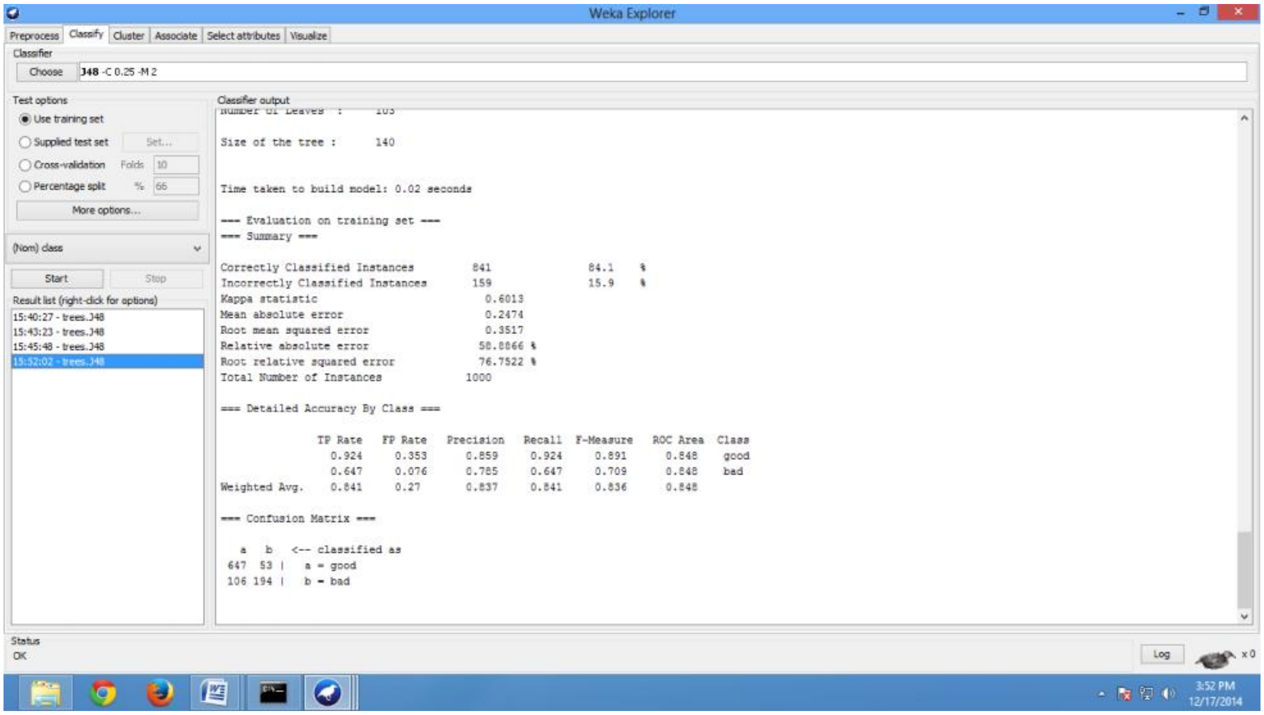

Fig 12 After removing 2nd attribute

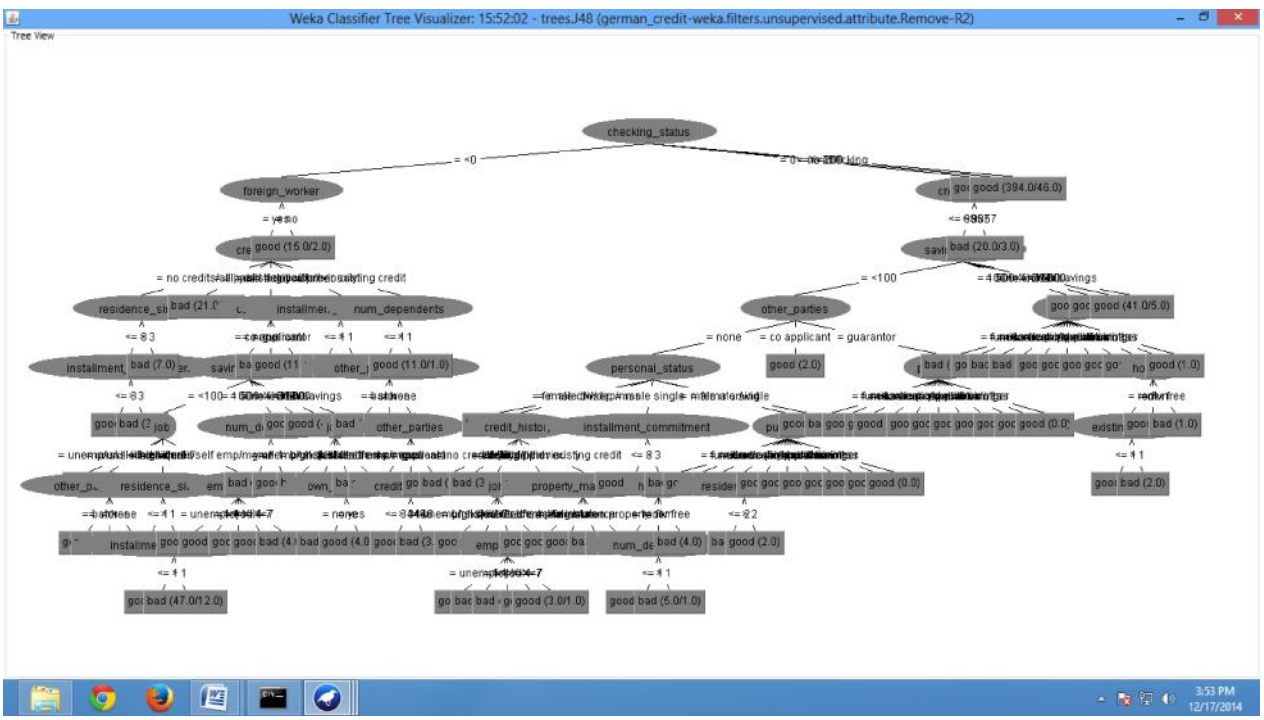

Fig 12.1 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 2nd attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	841	84.1 %
Incorrectly Classified Instances	159	15.9 %
Kappa statistic	0.6013	
Mean absolute error	0.2474	
Root mean squared error	0.3517	
Relative absolute error	58.8866 %	
Root relative squared error	76.7522 %	
Coverage of cases (0.95 level)	100 %	
Mean rel. region size (0.95 level)	94.95 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.924	0.353	0.859	0.924	0.891	0.848	good
	0.647	0.076	0.785	0.647	0.709	0.848	bad
Weighted Avg.	0.841	0.27	0.837	0.841	0.836	0.848	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 647  53 |   a = good
 106 194 |   b = bad
```

Procedure: Classification after removing 3rd attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. In preprocessor, select 3rd attribute from attribute list and remove.
3. Select classifier tab, choose J48 decision tree and select training data set from test data option.
4. Start classification.

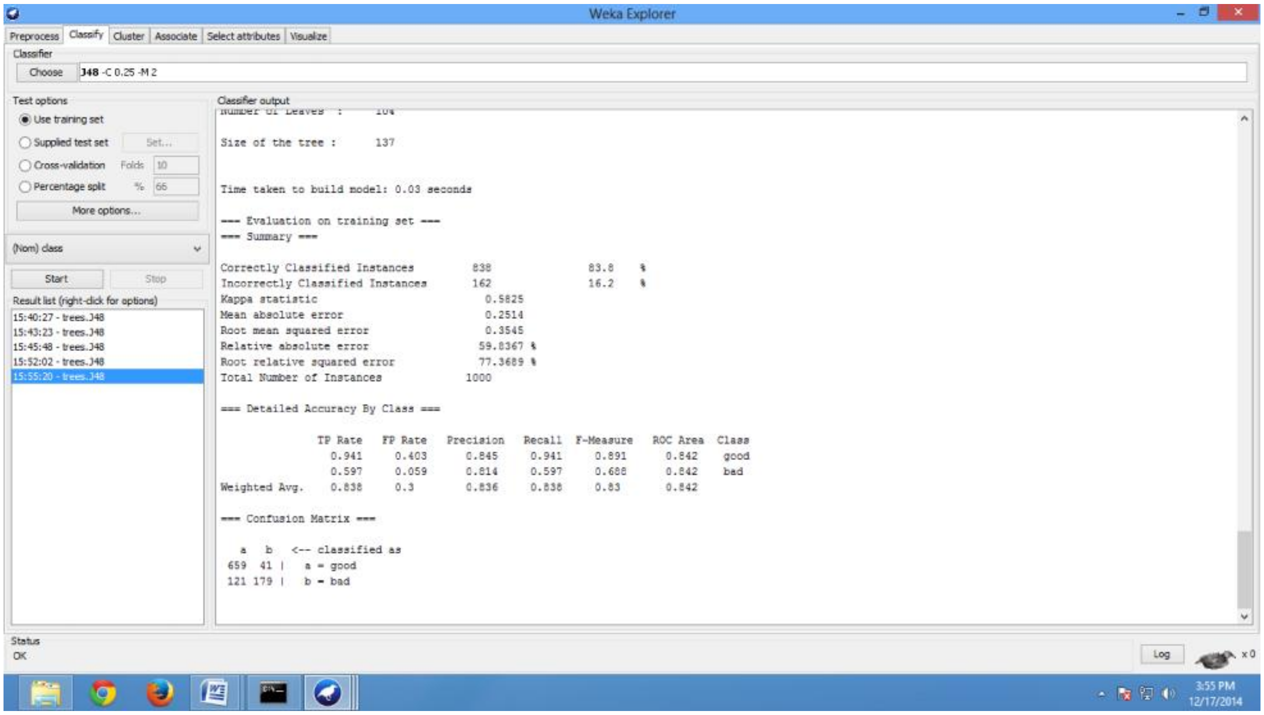

Fig 12.3 After removing 3rd attribute

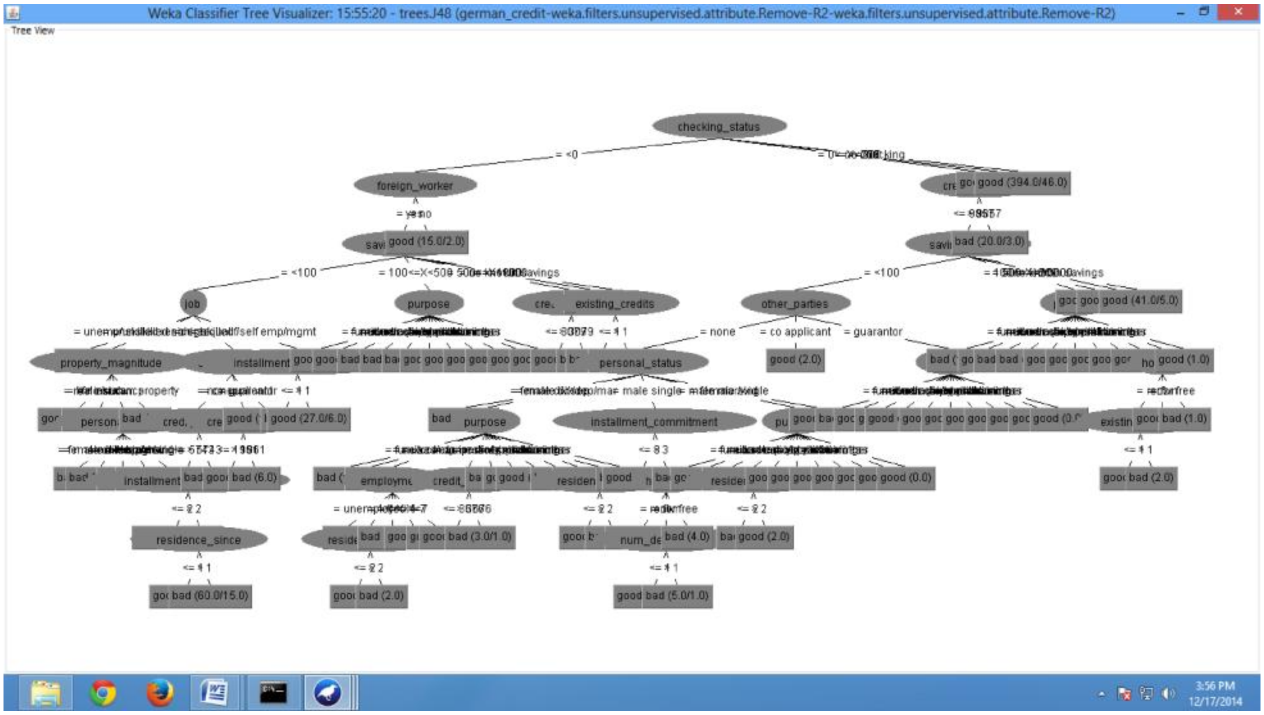

Fig 12.4 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 3rd attribute

=== 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 %

Coverage of cases (0.95 level) 100 %

Mean rel. region size (0.95 level) 94.6 %

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
```

Procedure: Classification after removing 5th attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

5. Open German data set arff file in Weka Explorer.
6. In preprocessor, select 5th attribute from attribute list and remove.
7. Select classifier tab, choose J48 decision tree and select training data set from test data option.
8. Start classification.

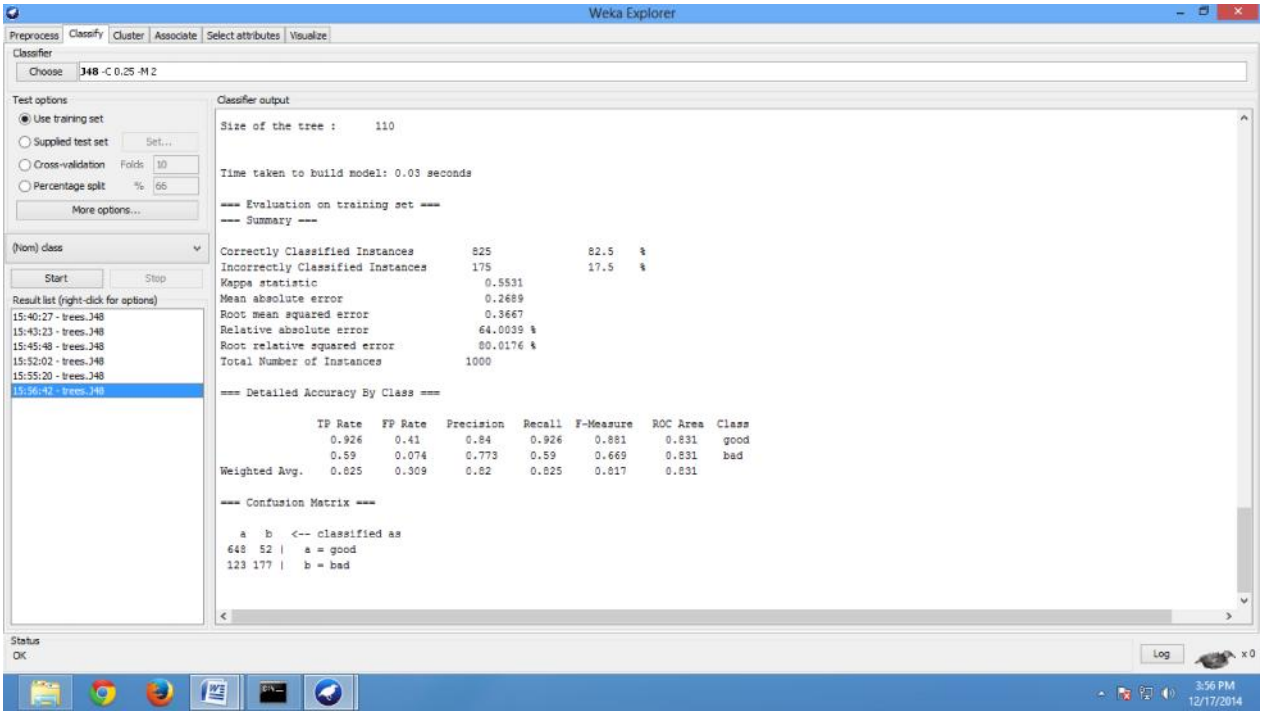

Fig 12.5 After removing 5th attribute

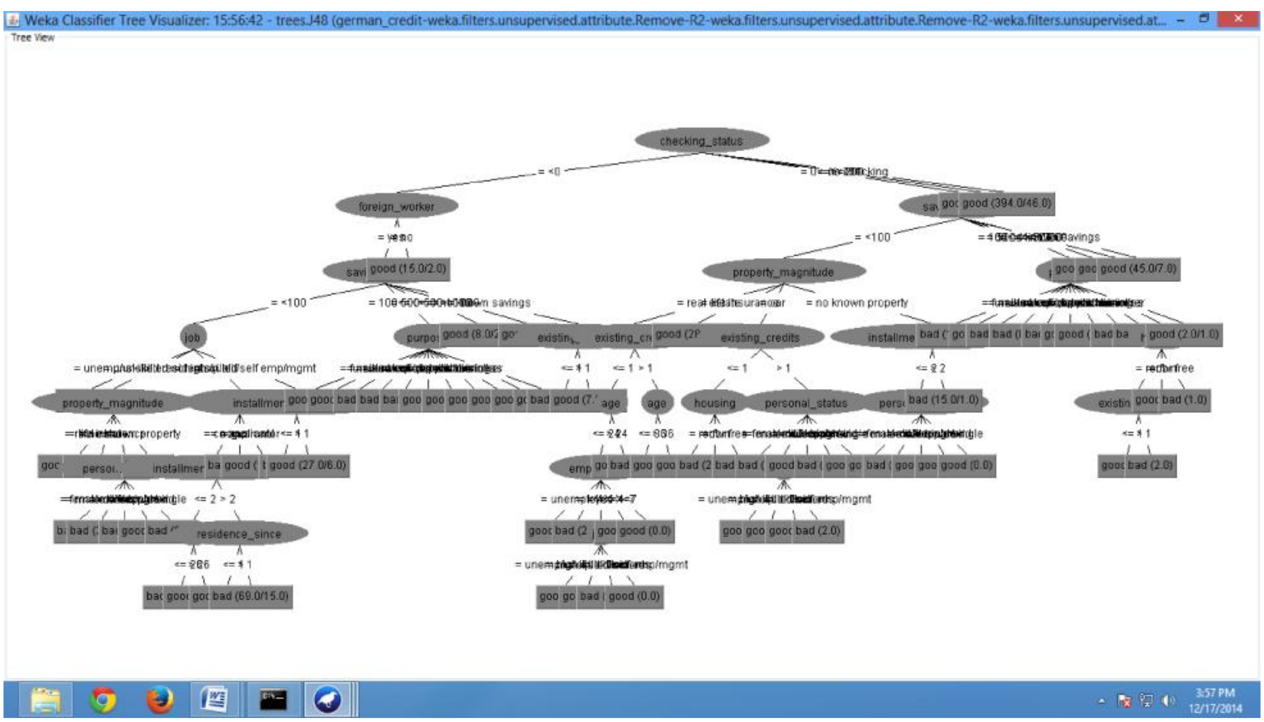

Fig 12.6 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 5th attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	864	86.4	%
Incorrectly Classified Instances	136	13.6	%
Kappa statistic	0.6473		
Mean absolute error	0.2191		
Root mean squared error	0.331		
Relative absolute error	52.1462 %		
Root relative squared error	72.226 %		
Coverage of cases (0.95 level)	99.9 %		
Mean rel. region size (0.95 level)	90.65 %		
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.964	0.37	0.859	0.964	0.908	0.866	good
	0.63	0.036	0.883	0.63	0.735	0.866	bad
Weighted Avg.	0.864	0.27	0.866	0.864	0.857	0.866	

=== Confusion Matrix ===

a b <-- classified as

675 25 | a = good

111 189 | b = bad

Procedure: Classification after removing 7th attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

5. Open German data set arff file in Weka Explorer.
6. In preprocessor, select 7th attribute from attribute list and remove.
7. Select classifier tab, choose J48 decision tree and select training data set from test data option.
8. Start classification.

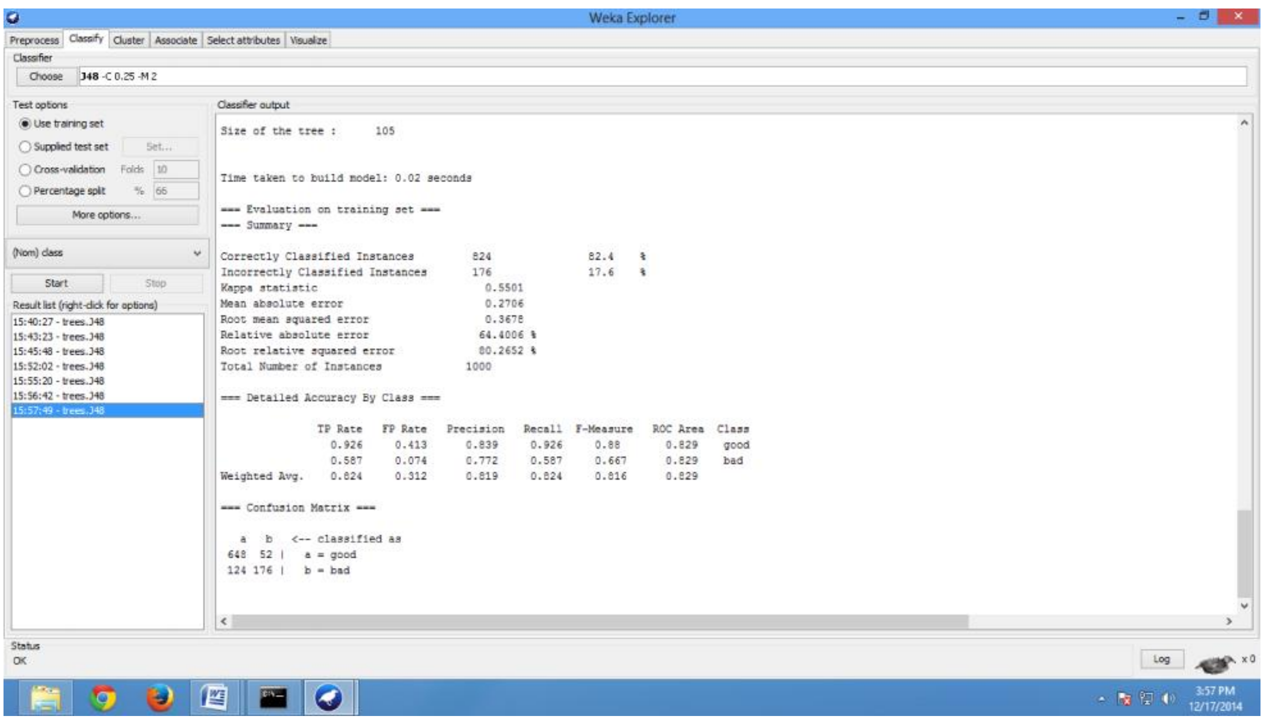

Fig 12.7 After removing 7th attribute

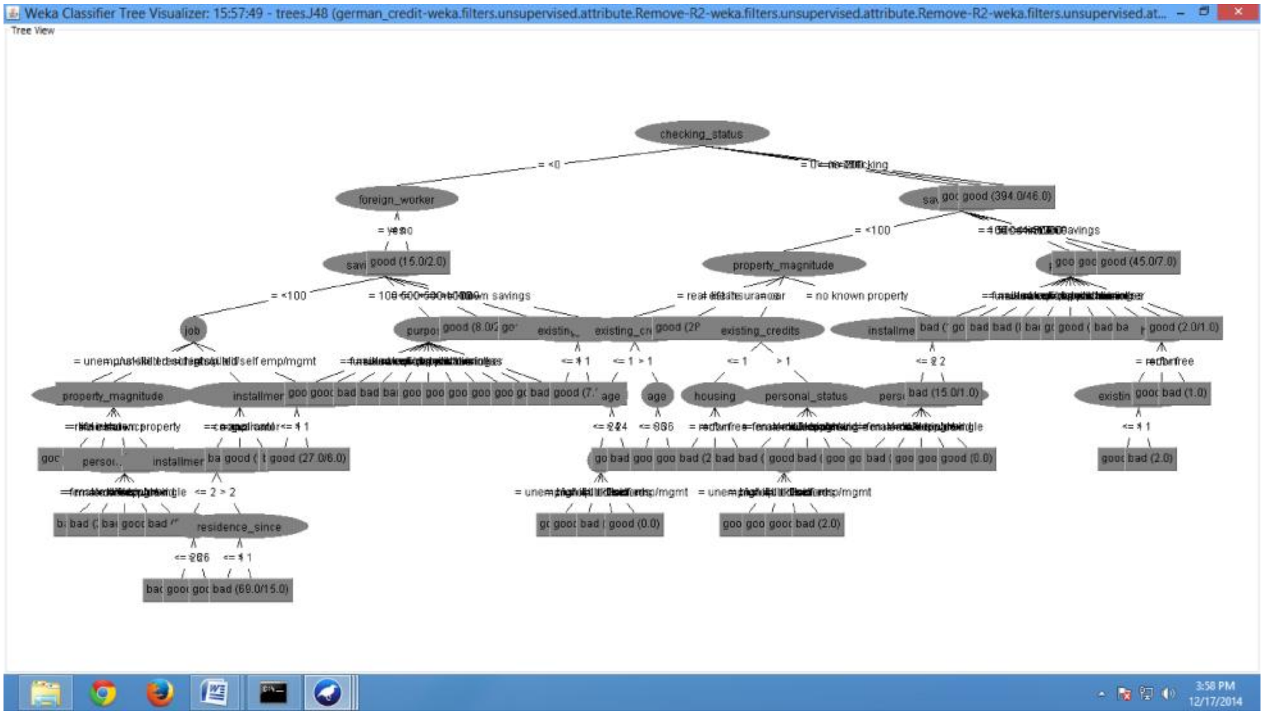

Fig 12.8 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 7th attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	858	85.8	%
Incorrectly Classified Instances	142	14.2	%
Kappa statistic	0.6333		
Mean absolute error	0.227		
Root mean squared error	0.3369		
Relative absolute error	54.0381 %		
Root relative squared error	73.5245 %		
Coverage of cases (0.95 level)	100	%	

Mean rel. region size (0.95 level) 93 %

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.957	0.373	0.857	0.957	0.904	0.86	good
	0.627	0.043	0.862	0.627	0.726	0.86	bad
Weighted Avg	0.858	0.274	0.858	0.858	0.851	0.86	

=== Confusion Matrix ===

```
  a   b   <-- classified as
670  30 |   a = good
112 188 |   b = bad
```

Procedure: Classification after removing 10th attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

9. Open German data set arff file in Weka Explorer.
10. In preprocessor, select 10th attribute from attribute list and remove.
11. Select classifier tab, choose J48 decision tree and select training data set from test data option.
12. Start classification.

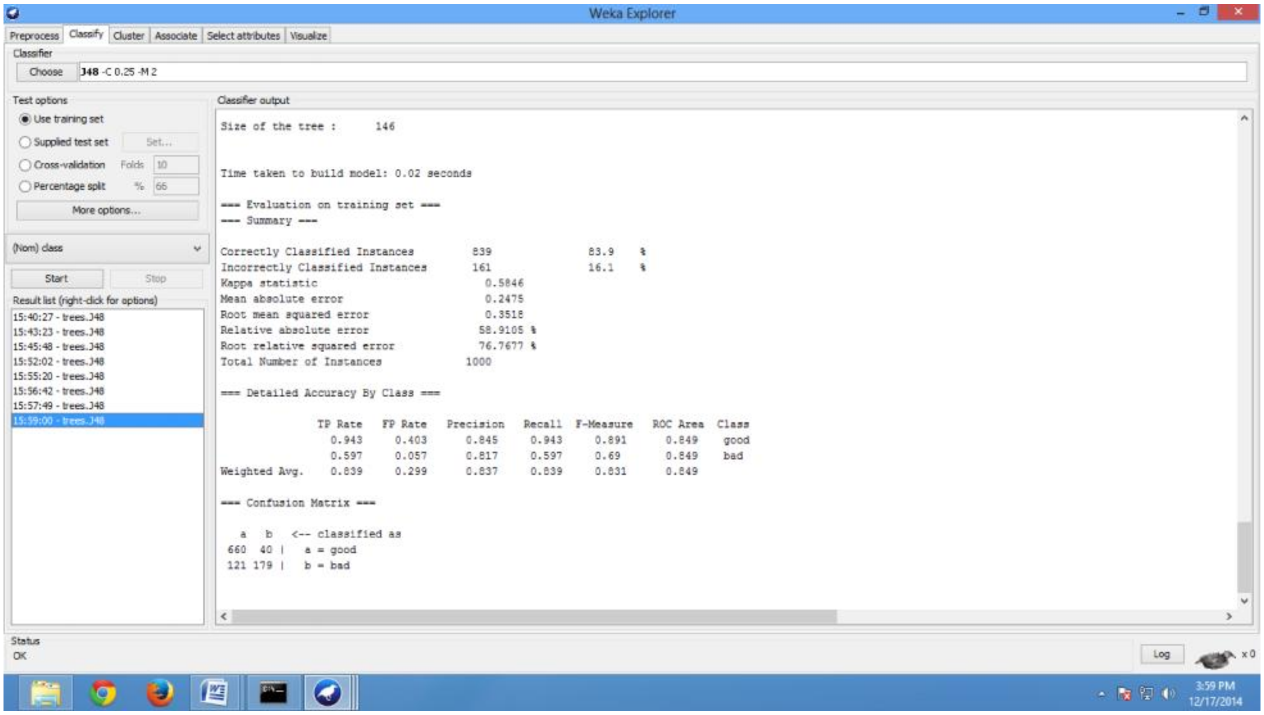

Fig 12.9 After removing 10th attribute

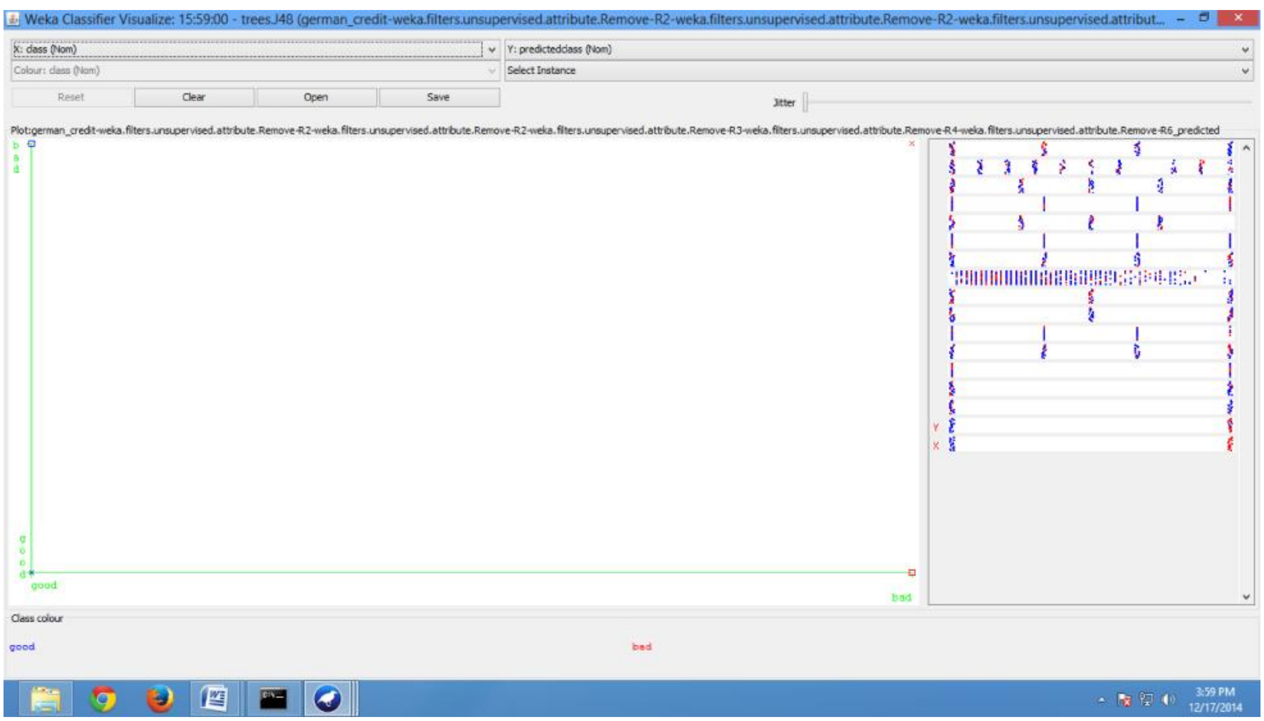

Fig 12.10 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 10th attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	845	84.5	%
Incorrectly Classified Instances	155	15.5	%
Kappa statistic	0.6001		
Mean absolute error	0.2427		
Root mean squared error	0.3483		
Relative absolute error	57.7623 %		
Root relative squared error	76.0159 %		
Coverage of cases (0.95 level)	100	%	
Mean rel. region size (0.95 level)	92.55	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.947	0.393	0.849	0.947	0.895	0.848	good
0.607	0.053	0.831	0.607	0.701	0.848	bad

Weighted Avg.   0.845   0.291   0.844   0.845   0.837   0.848

=== Confusion Matrix ===

a   b   <-- classified as

663 37 | a = good

118 182 | b = bad

Procedure: Classification after removing 17th attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

9. Open German data set arff file in Weka Explorer.
10. In preprocessor, select 17th attribute from attribute list and remove.
11. Select classifier tab, choose J48 decision tree and select training data set from test data option.
12. Start classification.

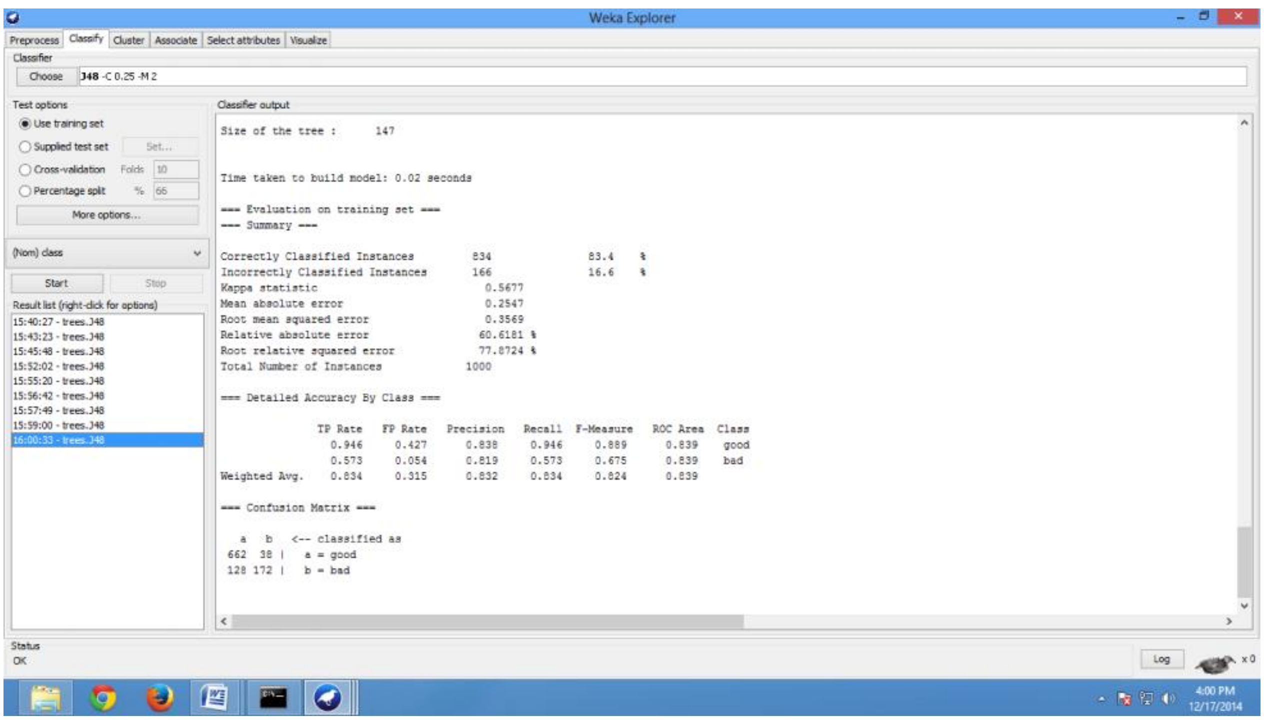

Fig 12.11 After removing 17th attribute

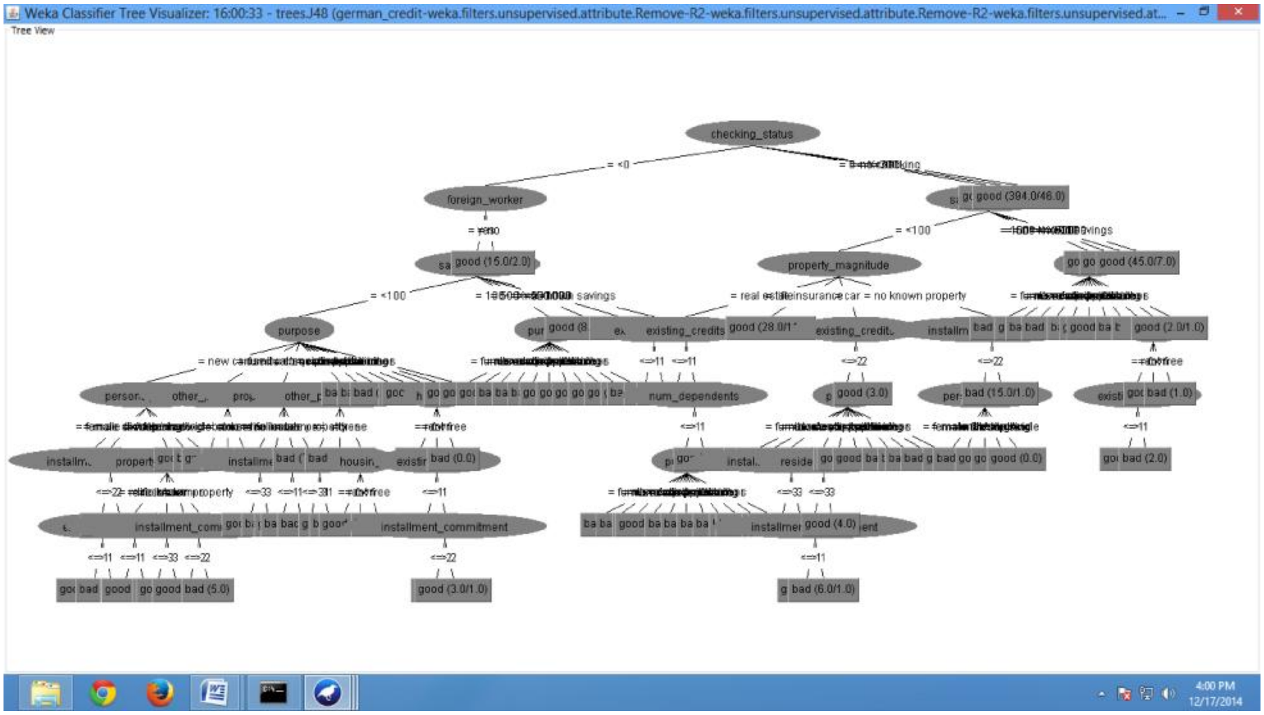

Fig 12.12 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 17th attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	859	85.9	%
Incorrectly Classified Instances	141	14.1	%
Kappa statistic	0.6324		
Mean absolute error	0.2254		

Root mean squared error 0.3357

Relative absolute error 53.6486 %

Root relative squared error 73.2591 %

Coverage of cases (0.95 level) 100 %

Mean rel. region size (0.95 level) 91.9 %

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.964	0.387	0.853	0.964	0.905	0.859	good
	0.613	0.036	0.88	0.613	0.723	0.859	bad
Weighted Avg.	0.859	0.281	0.861	0.859	0.851	0.859	

=== Confusion Matrix ===

a b <-- classified as

675 25 | a = good

116 184 | b = bad

Procedure: Classification after removing 21st attribute:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

13. Open German data set arff file in Weka Explorer.
14. In preprocessor, select 21st attribute from attribute list and remove.
15. Select classifier tab, choose J48 decision tree and select training data set from test data option.
16. Start classification.

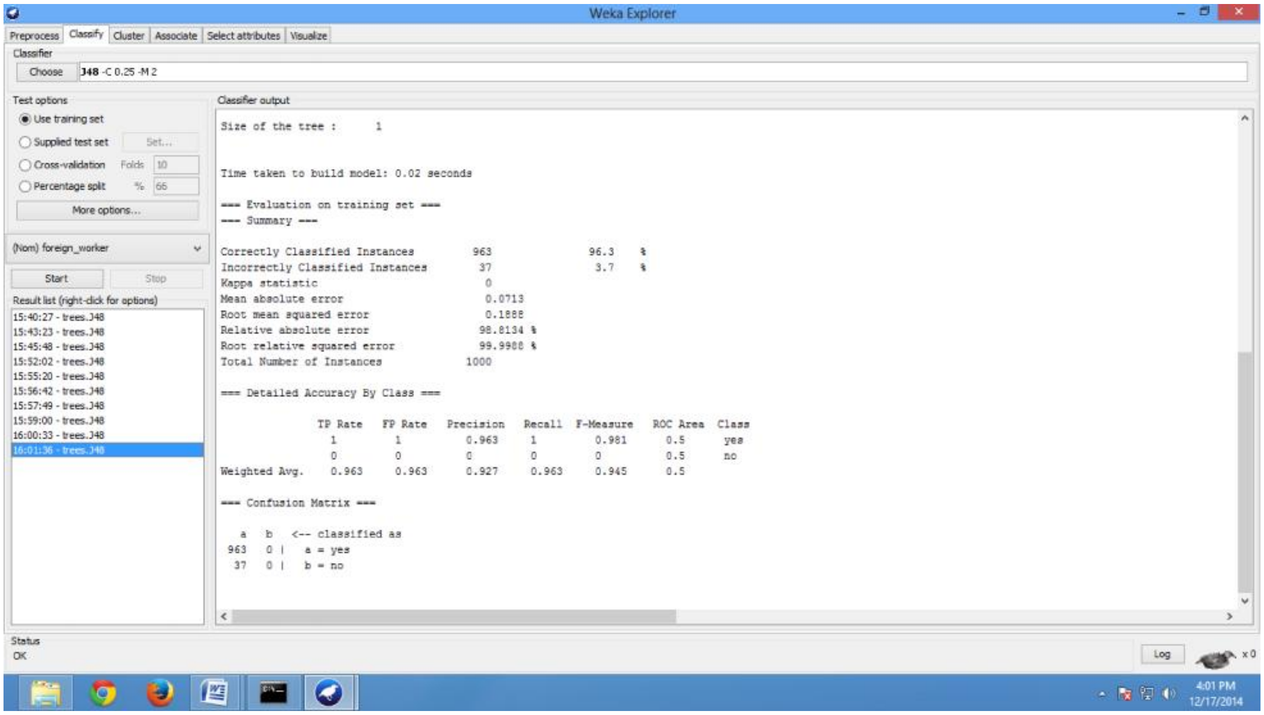

Fig 12.13 After removing 21st attribute

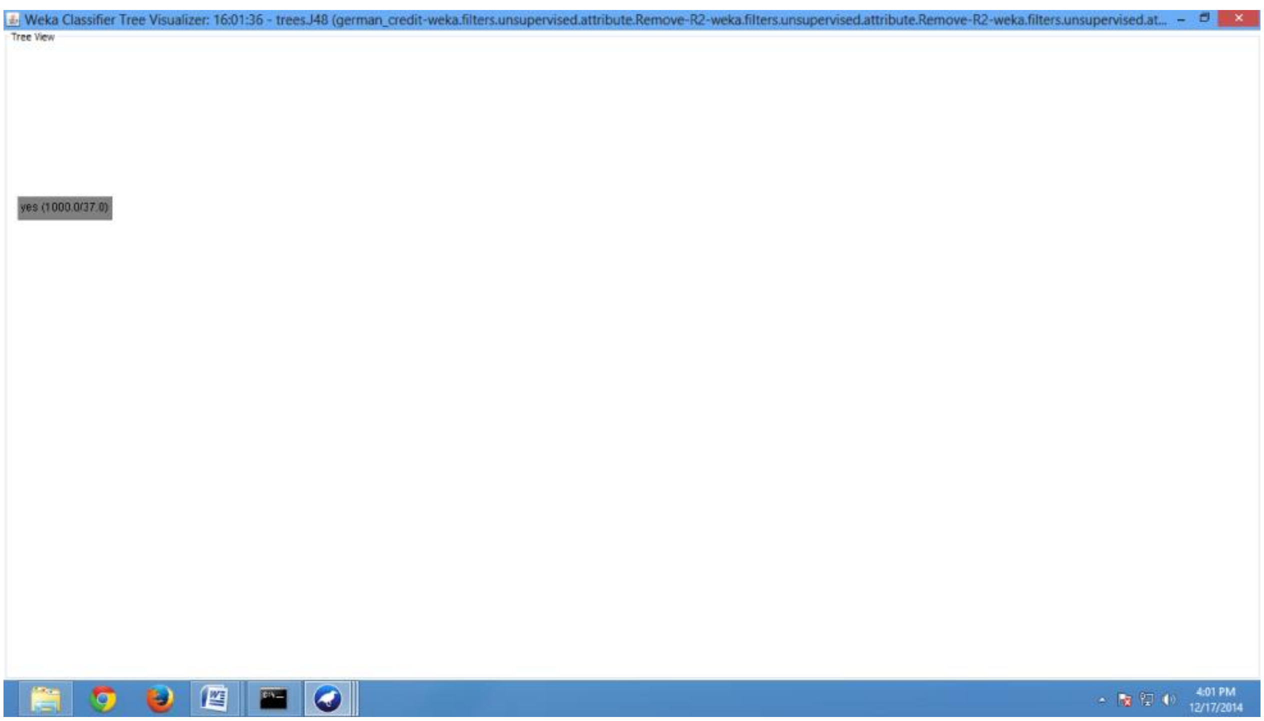

Fig 12.12 Decision tree by using J48

Output: The following model obtained after training the data set.

After removing 21st attribute

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	963	96.3	%
Incorrectly Classified Instances	37	3.7	%
Kappa statistic	0		
Mean absolute error	0.0713		
Root mean squared error	0.1888		
Relative absolute error	98.8134 %		
Root relative squared error	99.9988 %		
Coverage of cases (0.95 level)	96.3	%	
Mean rel. region size (0.95 level)	50	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.963	1	0.981	0.5	yes
	0	0	0	0	0	0.5	no
Weighted Avg.	0.963	0.963	0.927	0.963	0.945	0.5	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 963   0 |   a = yes
  37   0 |   b = no
```

Conclusion: With this observation we have seen, when 3rd attribute is removed from the Dataset, the accuracy (83%) is decreased. So this attribute is important for classification. when 2nd and 10th attributes are removed from the Dataset, the accuracy(84%) is same. So we can remove any one among them. when 7th and 17th attributes are removed from the Dataset, the accuracy(85%) is same. So we can remove any one among them. If we remove 5th and 21st attributes the accuracy is increased, so these attributes may not be needed for the classification.

Viva Questions:

1. What are the components of Decision tree?
2. Define attribute selection measure?
3. List attribute selection measure?

***

Experiment 9

Aim: Sometimes, The cost of rejecting an applicant who actually has good credit might be higher than accepting an applicant who has bad credit. Instead of counting the misclassification equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. By using a cost matrix in weak. Train your decision tree and report the Decision Tree and cross validation results. Are they significantly different from results obtained in problem 6.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka GUI Explorer.
2. In classify tab then press choose button in that Select J48 decision tree and select Use training data set option from test data option.

3. In classify tab press More options button then we get classifier evaluation options window in that select cost sensitive evaluation the press set button then we get Cost Matrix Editor.
4. Change classes as 2 then press Resize button. We get 2 by 2 Cost Matrix. In cost matrix (0,1) location change value as 5, we get modified cost matrix is as follows:0.0 5.0
   1.0 0.0
5. Then close the cost matrix editor, then press ok button.

Then press start button.

Output:

In the Problem 6, we used equal cost and we trained the decision tree. But here, we consider two cases with different cost.
Let us take cost 5 in case 1 and cost 2 in case 2.
When we give such costs in both cases and after training the decision tree, we can observe that almost equal to that of the decision tree obtained in problem 6.
But we find some difference in cost factor which is in summary in the difference in cost factor.
Case1 (cost 5) Case2 (cost 5)
Total Cost 3820 1705
Average cost 3.82 1.705
We don‘t find this cost factor in problem 6. As there we use equal cost. This is the major difference between the results of problem 6 and problem 9.

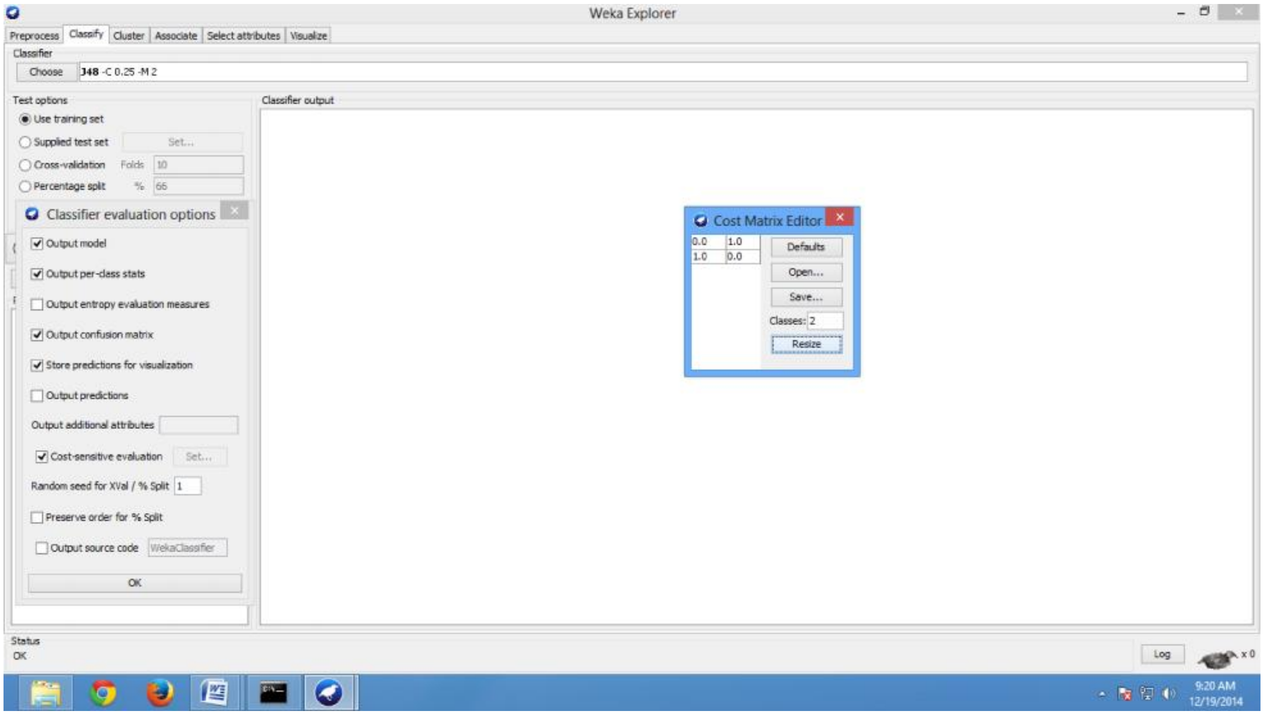

Fig 13. model obtained after training the data set

Conclusion: With this observation we have seen that ,total 700 customers in that 669 classified as good customers and 31 misclassified as bad customers. In total 300cusotmers, 186 classified as bad customers and 114 misclassified as good customers.

Viva Questions:

1. What are the type of prediction problems?
2. What is confusion matrix?
3. How many ways we can evaluate the classifier?

***

Experiment-10

Aim: Do you think it is a good idea to prefect simple decision trees instead of having long complex decision tress? How does the complexity of a Decision Tree relate to the bias of the model?

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select training data set from test data option.
3. Start classification.

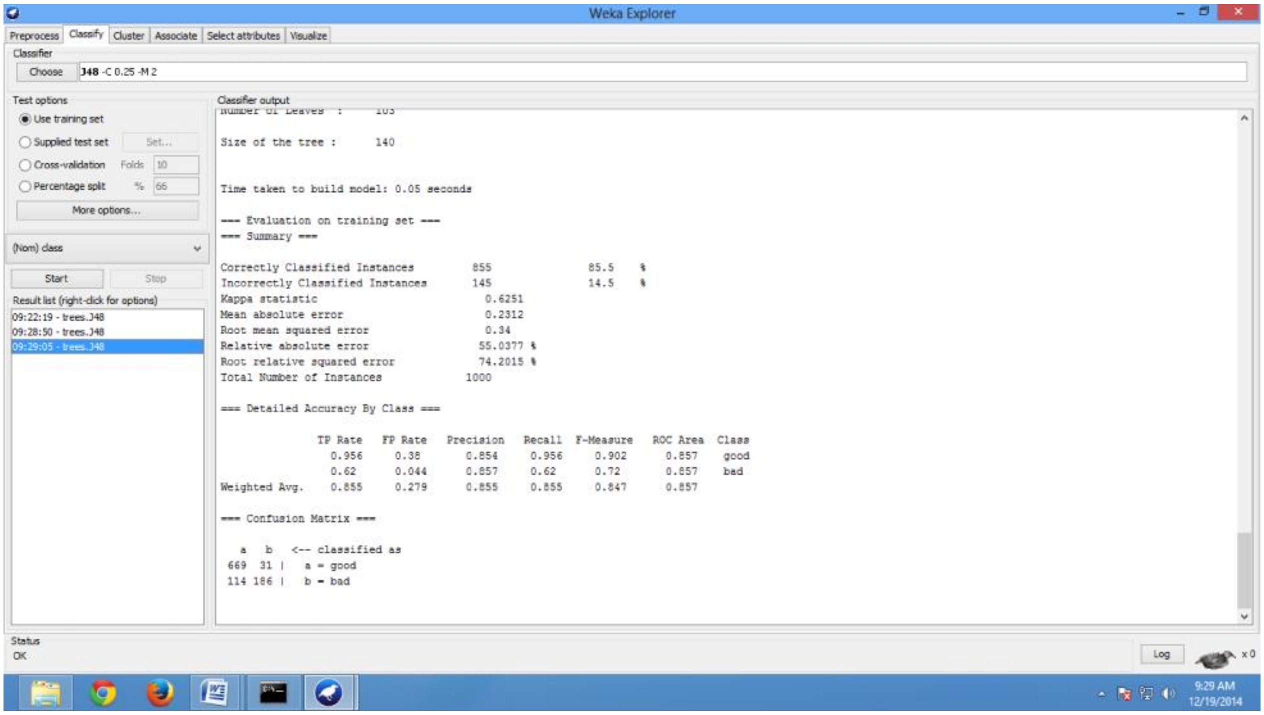

Fig 14.1: Training the data set.

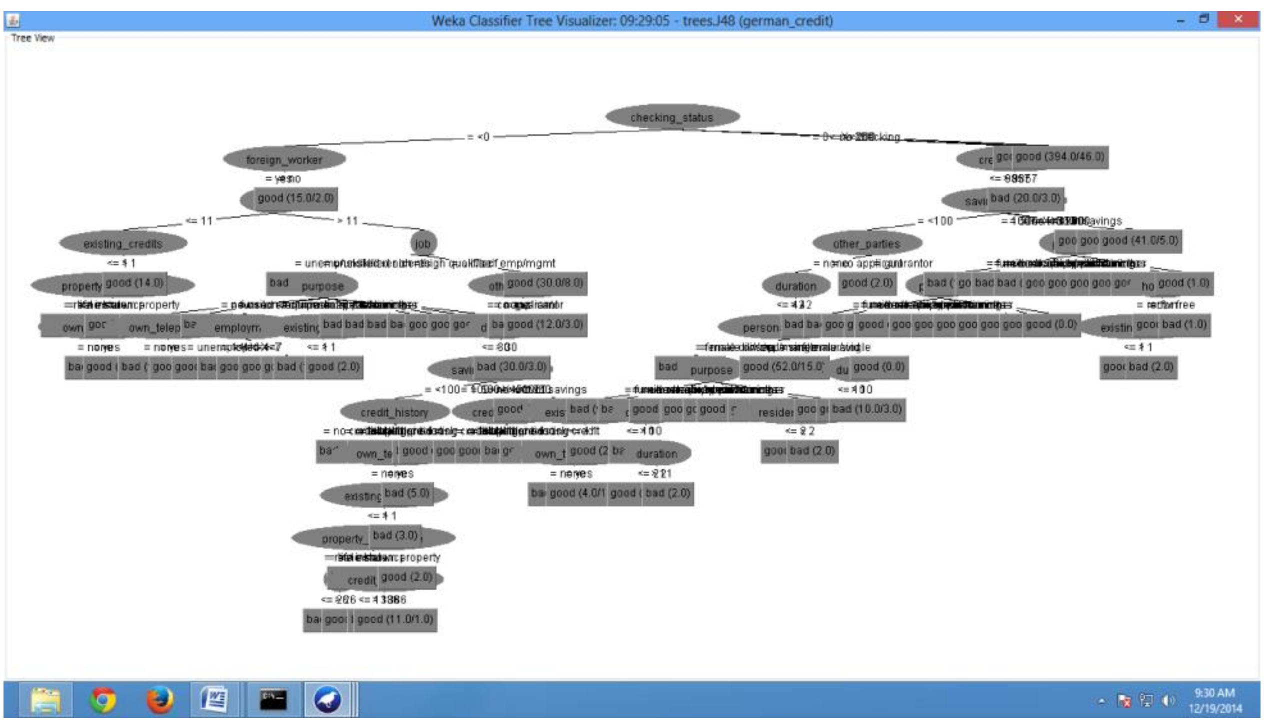

Fig 14.2: complexity of a Decision Tree

Output: The following model obtained after training the data set.

When we consider long complex decision trees, we will have many unnecessary attributes in the tree which results in increase of the bias of the model. Because of this, the accuracy of the model can also affected.
This problem can be reduced by considering simple decision tree. The attributes will be less and it decreases the bias of the model. Due to this the result will be more accurate.
So it is a good idea to prefer simple decision trees instead of long complex trees.

Conclusion: It is Good idea to prefer simple Decision trees, instead of having complex Decision tree.

Viva Questions:

1. Define decision tree.

2. Write the steps in decision tree algorithm.

3. Give an example of decision tree.

4. How to split the data set?

***

Experiment-11

Aim: You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning. Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross validation and report the Decision Trees you obtain? Also Report your accuracy using the pruned model Does your Accuracy increase?

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure:

We can make our decision tree simpler by pruning the nodes. Created a decision tree by using J48 Technique for the complete dataset as the training data in Weka Explorer.

1. Open German data set arff file in Weka Explorer.
2. Select classifier tab, choose J48 decision tree and select training data set from test data option. Beside Choose Button press on J48 –c 0.25 –M2 text, it displays Generic Object Editor. Select Reduced as True then press OK.
3. Start classification.

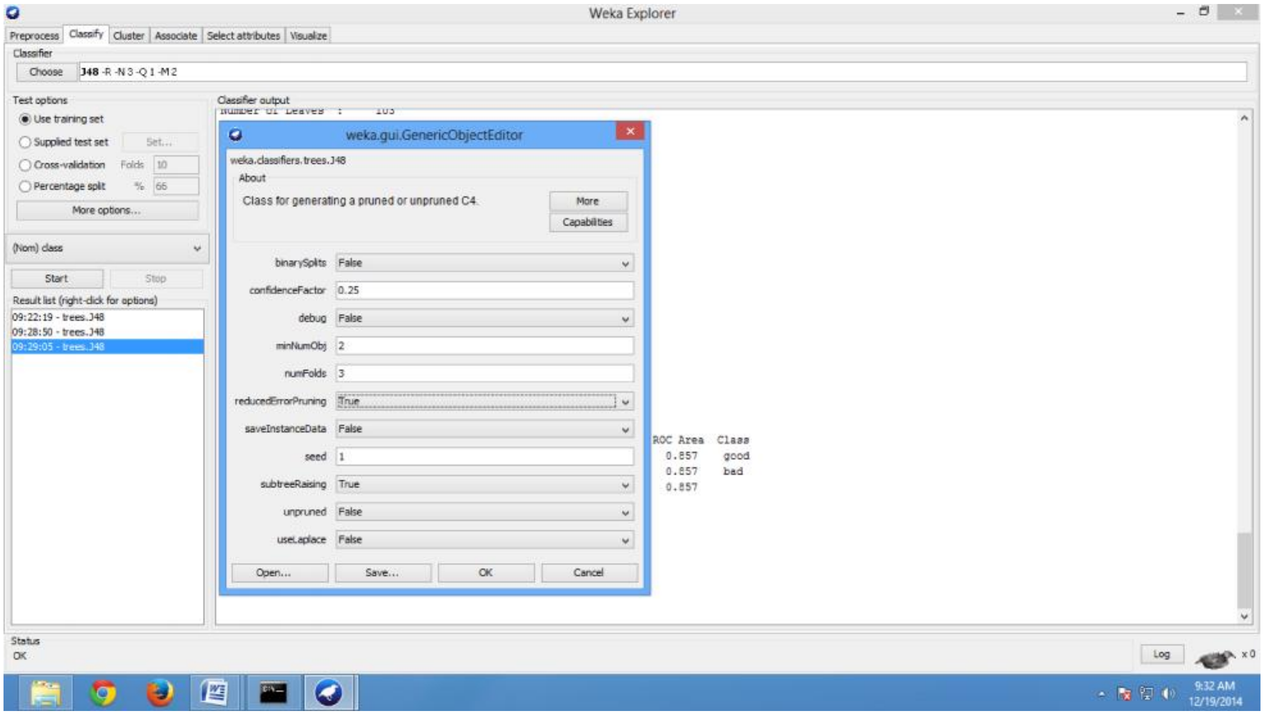

Fig 15.1:Generic object editor

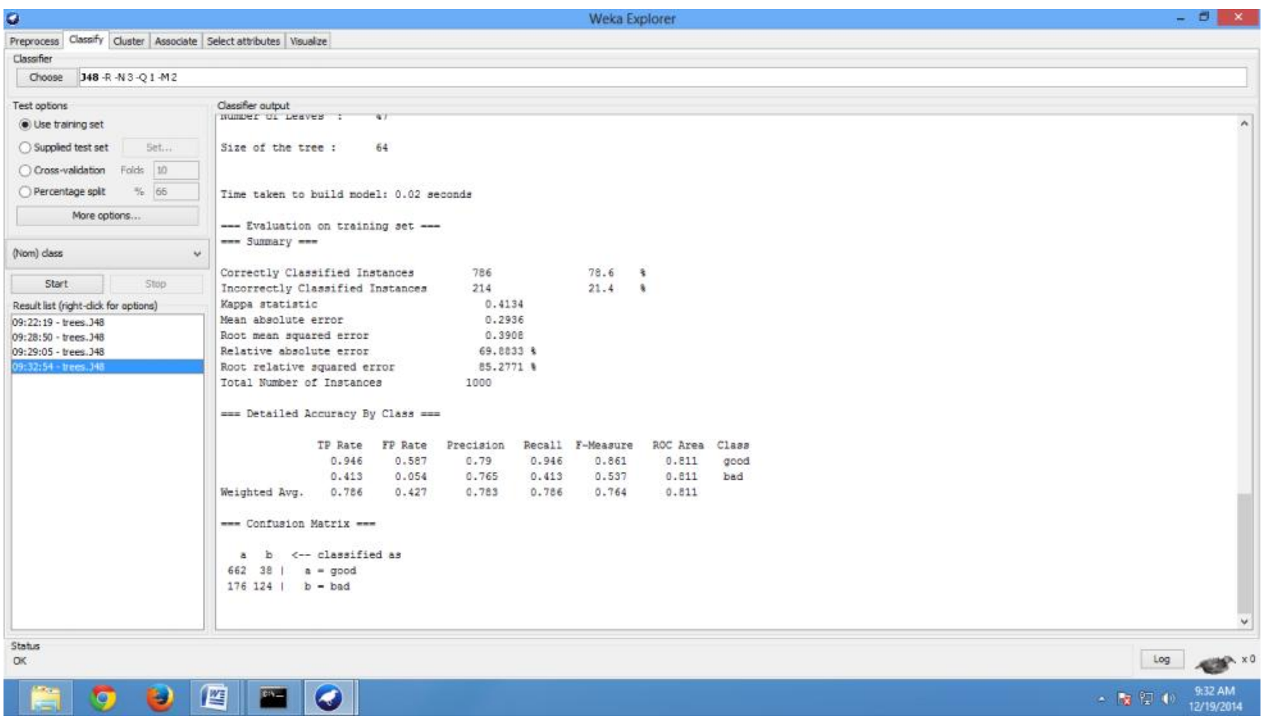

Fig15.2:Data set classification

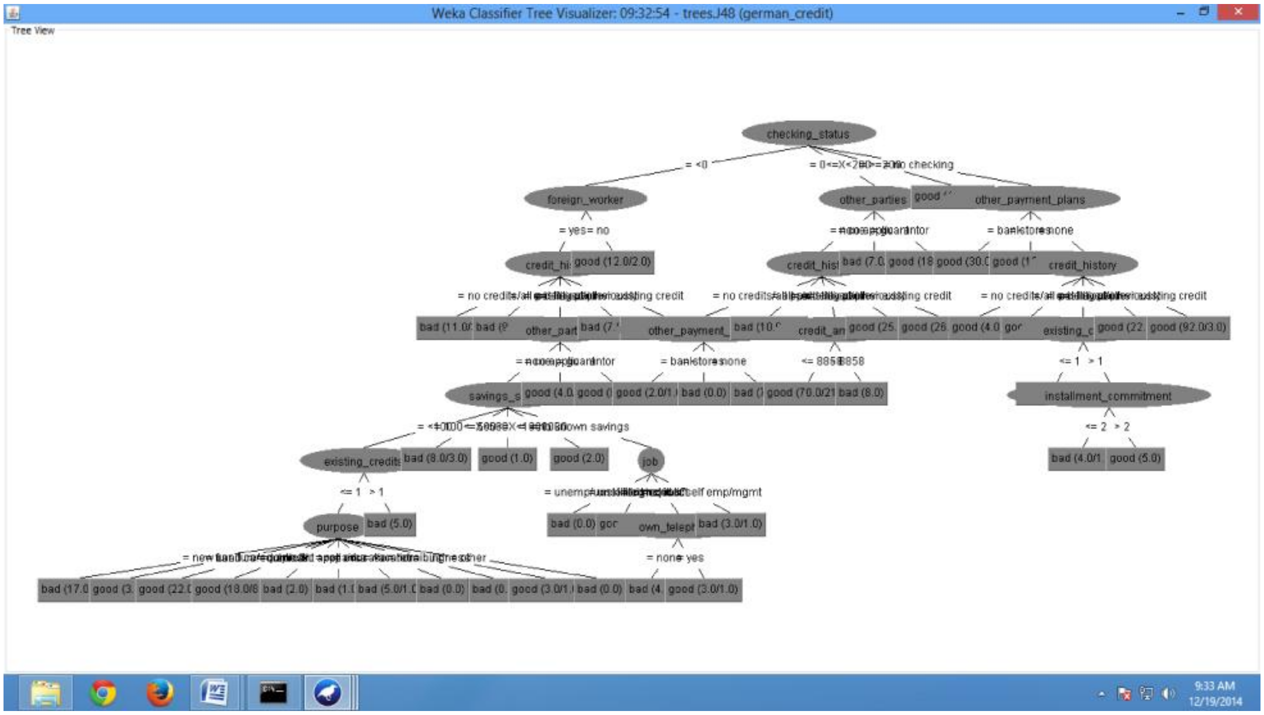

Fig 15.3: J48 decision tree

Output: The following model obtained after training the data set.

The idea of using a separate pruning set for pruning—which is applicable to decision trees as well as rule sets—is called reduced-error pruning. The variant described previously prunes a rule immediately after it has been grown and is called incremental reduced-error pruning.

Another possibility is to build a full, unpruned rule set first, pruning it afterwards by discarding individual tests. However, this method is much slower. Of course, there are many different ways to assess the worth of a rule based on the pruning set. A simple measure is to consider how well the rule would do at discriminating the predicted class from other classes if it were the only rule in the theory, operating under the closed world assumption. If it gets p instances right out of the t instances that it covers, and there are P instances of this class out of a total T of instances altogether, then it gets positive instances right. The instances that it does not cover include N - n negative ones, where n = t – p is the number of negative instances that the rule covers and N = T - P is the total number of negative instances. Thus the rule has an overall success ratio of [p +(N - n)] T , and this quantity, evaluated on the test set, has been used to evaluate the success of a rule when using reduced-error pruning.

J48 pruned tree

0/1.0)
| | | | | | purpose = furniture/equipment: good (22.0/11.0)
| | | | | | purpose = radio/tv: good (18.0/8.0)
| | | | | | purpose = domestic appliance: bad (2.0)
| | | | | | purpose = repairs: bad (1.0)
| | | | | | purpose = education: bad (5.0/1.0)
| | | | | | purpose = vacation: bad (0.0)
| | | | | | purpose = retraining: bad (0.0)
| | | | | | purpose = business: good (3.0/1.0)
| | | | | | purpose = other: bad (0.0)
| | | | | existing_credits > 1: bad (5.0) checking_status = <0
| foreign_worker = yes
| | credit_history = no credits/all paid: bad (11.0/3.0)
| | credit_history = all paid: bad (9.0/1.0)
| | credit_history = existing paid
| | | other_parties = none
| | | | savings_status = <100
| | | | | existing_credits <= 1
| | | | | | purpose = new car: bad (17.0/4.0)
| | | | | | purpose = used car: good (3.
| | | | savings_status = 100<=X<500: bad (8.0/3.0)
| | | | savings_status = 500<=X<1000: good (1.0)
| | | | savings_status = >=1000: good (2.0)
| | | | savings_status = no known savings
| | | | | job = unemp/unskilled non res: bad (0.0)
| | | | | job = unskilled resident: good (2.0)
| | | | | job = skilled
| | | | | | own_telephone = none: bad (4.0)
| | | | | | own_telephone = yes: good (3.0/1.0)
| | | | | job = high qualif/self emp/mgmt: bad (3.0/1.0)
| | | other_parties = co applicant: good (4.0/2.0)
| | | other_parties = guarantor: good (8.0/1.0)
| | credit_history = delayed previously: bad (7.0/2.0)
| | credit_history = critical/other existing credit: good (38.0/10.0)
| foreign_worker = no: good (12.0/2.0)
checking_status = 0<=X<200
| other_parties = none
| | credit_history = no credits/all paid
| | | other_payment_plans = bank: good (2.0/1.0)
| | | other_payment_plans = stores: bad (0.0)

| | | other_payment_plans = none: bad (7.0)
| | credit_history = all paid: bad (10.0/4.0)
| | credit_history = existing paid
| | | credit_amount <= 8858: good (70.0/21.0)
| | | credit_amount > 8858: bad (8.0)

| | credit_history = delayed previously: good (25.0/6.0)
| | credit_history = critical/other existing credit: good (26.0/7.0)
| other_parties = co applicant: bad (7.0/1.0)
| other_parties = guarantor: good (18.0/4.0)
checking_status = >=200: good (44.0/9.0)
checking_status = no checking
| other_payment_plans = bank: good (30.0/10.0)
| other_payment_plans = stores: good (12.0/2.0)
| other_payment_plans = none
| | credit_history = no credits/all paid: good (4.0)
| | credit_history = all paid: good (1.0)
| | credit_history = existing paid
| | | existing_credits <= 1: good (92.0/7.0)
| | | existing_credits > 1
| | | | installment_commitment <= 2: bad (4.0/1.0)
| | | | installment_commitment > 2: good (5.0)
| | credit_history = delayed previously: good (22.0/6.0)
| | credit_history = critical/other existing credit: good (92.0/3.0)
Number of Leaves : 47
Size of the tree : 64
Time taken to build model: 0.49 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 725 72.5 %
Incorrectly Classified Instances 275 27.5 %
Kappa statistic 0.2786
Mean absolute error 0.3331
Root mean squared error 0.4562
Relative absolute error 79.2826 %
Root relative squared error 99.5538 %
Total Number of Instances 1000

Conclusion: By using pruned model, the accuracy decreased. Therefore by pruning the nodes we can make our decision tree simpler.

Viva Questions:

1. define decision tree induction.

2. Define data classification.

3. define attribute ranking.

4. how to find the attribute ranking?

***

Experiment-12

Aim: How can you convert a Decision Tree into "if-then-else rules". Make up your own small Decision Tree consisting 2-3 levels and convert 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 data set? One R 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, one R.

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudocode, the general algorithm for building decision trees is:

2. Check for base cases
3. For each attribute *a*
1. Find the normalized information gain ratio from splitting on *a*
4. Let *a_best* be the attribute with the highest normalized information gain
5. Create a decision *node* that splits on *a_best*
6. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

Procedure: In Weka GUI Explorer, Select Classify Tab, In that Select **Use Training set** option .There also exist different classifiers that output the model in the form of Rules. Such classifiers in weka are "PART" and "OneR" . Then go to Choose and select **Rules** in that select PART and press start Button.

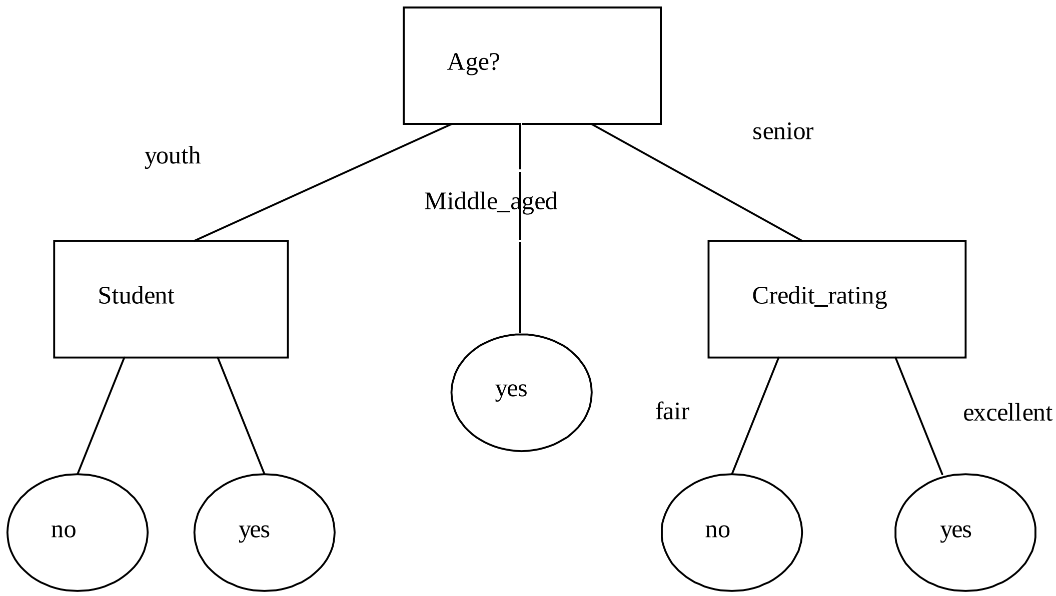

Fig 16: Sample Decision Tree of 2-3 levles

Output: The following model obtained after training the data set.

In weka, rules.PART is one of the classifier which converts the decision trees into

–IF-THEN-ELSE‖ rules.

Converting Decision trees into “IF-THEN-ELSE” rules using rules.PART classifier:-
PART decision list
outlook = overcast: yes (4.0)
windy = TRUE: no (4.0/1.0)
outlook = sunny: no (3.0/1.0)
: yes (3.0)
Number of Rules : 4
Yes, sometimes just one attribute can be good enough in making the decision.
In this dataset (Weather), Single attribute for making the decision is **“outlook”**
outlook:
sunny -> no
overcast -> yes
rainy -> yes
(10/14 instances correct)
With respect to the **time**, the oneR classifier has higher ranking and J48 is in 2nd place and PART gets 3rd place.

J48 PART oneR
TIME (sec) 0.12 0.14 0.04
RANK II III I
But if you consider the **accuracy,** The J48 classifier has higher ranking, PART gets second place and oneR
gets lst place
J48 PART oneR
ACCURACY (%) 70.5 70.2% 66.8%
RANK I II III

Viva Questions:

1. Define decision tree.

2. Write the steps in decision tree algorithm.

3. Give an example of decision tree.

4. How to split the data set?

***

8. Content of Additional Experiments

Additional Experiment-1

Aim: Perform cluster analysis on German credit data set using partition clustering algorithm

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudo code, the general algorithm for k-means clustering algorithm is:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Procedure: In Weka GUI Explorer, Select Cluster Tab, In that Select **Simplekmeans**. Then go to Choose and select use training set. Click on start.

Output: cluster analysis on k-means clustering algorithm

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -O -S 10
Relation:     german_credit-weka.filters.unsupervised.attribute.Remove-R21
Instances:    1000
Attributes:   20
              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

Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans
======

Number of iterations: 8
Within cluster sum of squared errors: 5145.269062855846
Missing values globally replaced with mean/mode

Cluster centroids:

		Cluster#		
Attribute	Full Data	0	1	2
	(1000)	(484)	(190)	(326)
checking_status	no checking	no checking	<0	0<=X<200
duration	20.903	20.7314	26.0526	18.1564
credit_history	existing paid	existing paid	existing paid	existing paid

purpose	radio/tv	new car	used car	radio/tv
credit_amount	3271.258	3293.1281	4844.6474	2321.7822
savings_status	<100	<100	<100	<100
employment	1<=X<4	1<=X<4	>=7	>=7
installment_commitment	2.973	2.8822	3.0579	3.0583
personal_status	male single	male single	male single	male single
other_parties	none	none	none	none
residence_since	2.845	2.4483	3.5211	3.0399
property_magnitude	car	car	no known property	real estate
age	35.546	33.155	41.0526	35.8865
other_payment_plans	none	none	none	none
housing	own	own	for free	own
existing_credits	1.407	1.3967	1.4474	1.3988
job	skilled	skilled	skilled	skilled
num_dependents	1.155	1.155	1.2474	1.1012
own_telephone	none	none	yes	none
foreign_worker	yes	yes	yes	yes

Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 484 (48%)
1 190 (19%)
2 326 (33%)

Additional Experiment-2

Aim: Perform cluster analysis on German credit data set using hierarchal clustering algorithm

Recommended Hardware / Software Requirements:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
- Weka

Prerequisites: Student should have knowledge about data mining techniques and should know how to use automated tools.

Pseudo code:

In pseudo code, the general algorithm using EM clustering algorithm is

Procedure: In Weka GUI Explorer, Select Cluster Tab, In that Select **EM**. Then go to Choose and select use training set. Click on start.

Output:

```
=== Run information ===

Scheme:       weka.clusterers.EM -I 100 -N -1 -M 1.0E-6 -S 100
Relation:     german_credit-weka.filters.unsupervised.attribute.Remove-R21
Instances:    1000
Attributes:   20
              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
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


EM
==

Number of clusters selected by cross validation: 4


                              Cluster
Attribute                          0         1         2         3
                               (0.26)    (0.26)     (0.2)    (0.29)
======================================================================
============
checking_status
  <0                            100.8097   58.5666   51.7958   66.8279
  0<=X<200                       69.3481   63.6477   34.9535  105.0507
  >=200                          17.6736   20.0978   11.9012   17.3274
  no checking                    73.012  119.2995  101.8966  103.7918
  [total]                       260.8434  261.6116  200.5471  292.9978
duration
  mean                           17.7484   14.3572   23.4112   27.8358
  std. dev.                       8.0841    7.1757   12.1018   14.1317

credit_history
  no credits/all paid            10.1705    6.0326    8.4795   19.3174 all
  paid                           17.9296   11.0899    9.6553   14.3252 existing
  paid                          175.3951  142.1934   53.3962  163.0153 delayed
  previously                     10.1938   18.0432   24.9273   38.8357
  critical/other existing credit 48.1544   85.2526  105.0888   58.5041
  [total]                       261.8434  262.6116  201.5471  293.9978
```

```
purpose
  new car                    57.7025   76.7946    47.734   55.7689 used
  car                   14.504    7.9487   40.7163    43.831
  furniture/equipment             95.3943   25.2704   24.1583   40.1769
  radio/tv                   53.3828  106.3023   48.3866   75.9283
  domestic appliance              7.9495    3.4917     1.161     3.3979
  repairs                    5.5771    9.5832    6.9408    3.8988
  education                     9.921   10.7236   11.9789   21.3766
  vacation                        1         1         1         1
  retraining                  4.7356    4.1209     2.311    1.8324
business                    16.6708    22.302   19.5059   42.5213
other                      1.0059    1.0743    3.6542   10.2656
[total]                  267.8434  268.6116  207.5471  299.9978
credit_amount
  mean                     2288.8498 1812.2911 3638.3737 5195.2049
  std. dev.                1342.8531  995.7303  2694.223 3683.9507

savings_status
  <100                      170.6648  165.5967   96.2641  174.4744
  100<=X<500                     26.3033   25.4915   18.3092   36.8959
  500<=X<1000                    15.6275   21.5273   15.5765   14.2688
  >=1000                     12.2318    18.448    12.513    8.8072
  no known savings               37.0161   31.5481   58.8844   59.5515
[total]                  261.8434  262.6116  201.5471  293.9978
employment
  unemployed                    14.0219    3.1801   16.0683   32.7298
  <1                       90.51   34.2062    8.4379    42.846
  1<=X<4                     84.9242   128.879   27.7645  101.4323
  4<=X<7                     50.6437   42.1897   31.3087    53.858
  >=7                      21.7437   54.1567  117.9679   63.1317
[total]                  261.8434  262.6116  201.5471  293.9978
installment_commitment
  mean                       2.8557    3.0212     3.312    2.8038
  std. dev.                  1.1596    1.1124    0.9515    1.1363

personal_status
  male div/sep                 15.737    9.9518    4.6205   23.6907
  female div/dep/mar              151.4625   48.4321   18.2787   95.8267
  male single                67.3068  159.5075  172.5861  152.5996
```

male mar/wid	26.3371	43.7203	5.0618	20.8808
female single	1	1	1	1
[total]	261.8434	262.6116	201.5471	293.9978
other_parties				
none	235.863	218.7895	186.4245	269.923
co applicant	12.5526	10.6977	6.9588	14.7909
guarantor	11.4278	31.1244	6.1638	7.2839
[total]	259.8434	260.6116	199.5471	291.9978
residence_since				
mean	2.6862	2.5399	3.5434	2.7831
std. dev.	1.1732	1.0186	0.7654	1.1061
property_magnitude				
real estate	69.0217	148.9943	30.8391	37.1449
life insurance	81.2718	54.4192	41.9034	58.4056
car	95.7773	51.1875	60.6462	128.389
no known property	14.7725	7.0107	67.1584	69.0583
[total]	260.8434	261.6116	200.5471	292.9978
age				
mean	27.7345	36.1057	43.8079	36.3705
std. dev.	5.7953	10.3158	11.3129	11.5738
other_payment_plans				
bank	34.4988	32.0758	33.984	42.4414
stores	10.9742	12.5287	10.4947	17.0024
none	214.3704	216.0071	155.0685	232.554
[total]	259.8434	260.6116	199.5471	291.9978
housing				
rent	85.8549	31.7206	15.9015	49.523
own	168.499	226.2291	124.0089	198.2629
for free	5.4895	2.6619	59.6367	44.2118
[total]	259.8434	260.6116	199.5471	291.9978
existing_credits				
mean	1.213	1.4137	1.7961	1.3088
std. dev.	0.4142	0.5377	0.7406	0.4734
job				
unemp/unskilled non res	11.7711	2.5192	6.8364	4.8733
unskilled resident	52.9713	105.4029	24.5489	21.0769

skilled	188.0096	147.8359	128.9987	169.1558
high qualif/self emp/mgmt	8.0914	5.8537	40.1631	97.8918
[total]	260.8434	261.6116	200.5471	292.9978
num_dependents				
mean	1	1.2978	1.3983	1
std. dev.	0.3621	0.4573	0.4895	0.3621
own_telephone				
none	219.2961	215.7304	81.1575	83.816
yes	39.5473	43.8813	117.3896	207.1818
[total]	258.8434	259.6116	198.5471	290.9978
foreign_worker				
yes	248.5954	234.0215	197.4796	286.9034
no	10.248	25.5901	1.0675	4.0944
[total]	258.8434	259.6116	198.5471	290.9978

Time taken to build model (full training data) : 22.43 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 279 (28%)
1 279 (28%)
2 194 (19%)
3 248 (25%)

Log likelihood: -33.06046

9.Text Book References

Andrew Moore‘s Data Mining Tutorial (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