Politecnico di Torino

Corso di Laurea Magistrale

in Ingegneria Informatica

(Computer Engineering)

Master thesis

A Lossy Compression Technique
For Structured Data Classification

Supervisor:
prof. Paolo Garza

Candidate:
Jiayi Li

December 2020

Acknowledgements

I would like to thank my thesis supervisor at Polytechnic of Turin, Paolo Garza, for his consistent support and guidance during the running of this project.

Jiayi Li, Turin, December 2020

Abstract

With the rapid development of data collection and data storage technology, machine learning has become an essential part of data analysis. As part of machine learning, Weka plays an important role in completing machine learning tasks (such as data preprocessing, classification, regression, clustering, and association rules). The advantage is that many algorithms are integrated in Weka. Due to the open-source code, developers can combine the Java language with the Weka API to continuously optimize the algorithm to improve the accuracy of data classification. Through data mining methods, algorithms such as ID3C4.5CART are also well known. Based on the advantages of small calculations, they are the most widely used, and the rules they generate can be easily implemented and understood.

With the development of machine learning, data mining and technical data compression is an inevitable trend, At the same time, establishing the right data model to improve the classification accuracy is very important. After an in-depth study of the classic classification methods in data mining. This paper combined with data compression and data classification following studies:

This article first briefly introduces the concept of machine learning, then introduces the use of Weka and the basic implementation architecture, and then details the implementation process of the classic decision tree algorithm and related interfaces in Weka. Use the data set selected from the UCI data set to meet the classification attributes for related classification and initialize the model training on the data set.

Secondly, combine the classified data with JAVA language and compression algorithm to perform lossy compression on the data and conduct model training through WEKA. After the two models are trained, compare their classification accuracy and the recall and predictability of each class.

Finally, the main work in the research process and the experimental results obtained from the analysis are summarized. At the same time, the optimization and improvement plan of lossy compression technology in classification technology is proposed, also pointed out the development prospects of further research.

Keywords: data mining, classification, decision tree, Weka, big data, compression data

Contents

List of Figures

List of Tables

1
Introduction

We live in a world full of information and data. In recent years, with the rapid development of communication technology and computer network technology, more and more data is accumulated in human databases, and the amount of data information stored digitally has reached unprecedented levels.

The massive increase in data is undoubtedly a very precious resource, mainly because there is a large amount of available data for people to use, and it has become urgent to convert this data into useful information and knowledge. People can obtain this information and can be widely used in various industries. Not only in the financial and commercial fields but also the medical and insurance industries. At this stage, many related algorithms and software are derived from managing data, but these conventional tools cannot meet the needs of people using data. At the same time, these technologies and tools cannot provide strong support for decision-makers. The effective basis for judgment and decision-making. This has also led to a phenomenon in the field of data management, where there is a large amount of relevant data information. Still, the required knowledge cannot be directly observed. Given the above problems, data mining technology has been discovered and applied in various industries in a short time. Big Data, this describes a large amount of super-complex and short-lived data that is unstructured and difficult to handle with ordinary computers. This is because after the data has been collected, it needs to be analysed for in-depth information. The real goal is to benefit from the information.

Some experts and scholars have translated data mining as data exploration and data mining, generally defined as searching in a large amount of data, to arrive at some hidden in the special relationship information, this information belongs to the Association rule learning, then this process is data mining. In layman's terms, it is in the huge, incomplete, noisy, uncertain data information, mining was hidden in the unknown but contained the knowledge of information, this process is the popular description of data mining. Data mining spans multiple disciplines, and it not only combines the latest technological achievements such as artificial intelligence, database technology, machine learning but also has a wide range of applications.

The process of the classification method can be briefly described as follows: filter out the information of irrelevant attributes by preprocessing the data of the sample set, and at the same time find out the attributes of a data category to calibrate it, after the data set is calibrated, And then select a certain number of records as the data of the training sample set, and finally, use a certain classification algorithm to mine the training sample set, and output the classification results.

The classification method mainly includes decision tree, Bayesian network, association-based rules, proximity method, genetic algorithm, support vector machine, and so on. Among all the classification methods, the decision tree classification algorithm is widely used because it has the advantages of relatively small computation, simple rules, and can handle discrete and continuous properties at the same time.

To sum up, as an important branch of data mining, decision tree classification will provide decision support for many industries and affect people's lives and the development of social sciences to a certain extent. In the field of data classification, decision tree algorithms have received great attention in recent years. It has irreplaceable advantages and has been widely used.

Purpose

The purpose of this article is to implement data mining based on java and WEKA and to analyze how to improve the accuracy of data classification. The data set is classified based on the open-source WEKA API and the existing integration algorithms in the WEKA software. Because of the diversity of data, we take a variety of real data from UCI data set that can achieve the nature of classification. The large data is split into training data sets and test data sets, WEKA API and java languages are combined; the original training data set is classified by algorithms. Decision tree models are created to understand further the nature and applicability of classification algorithms in WEKA and to be familiar with how to create decision tree models for data mining and data compression. Finally, by analyzing the classification model of the original data and the classification model of the compressed files, the accuracy of each classification model and the regression rate and prediction rate of each class are compared.

Context

The decision tree classification algorithm is one of the most common classification algorithms we know now, with the advantages of simple generation rules, simple learning and classification steps, quick construction of classifiers, the ability to work with different types of data, and so on, but at the same time, some problems are unavoidable.

1. How to select the right algorithm for classification for different types of data (numeric, enumeration)
2. How to handle missing data
3. How to improve the accuracy of the algorithm
4. How to handle and optimize large data set classifications
5. The construction and portability of the algorithm

Since the WEKA system integrates cutting-edge machine learning algorithms and data preprocessing tools, users can quickly and flexibly apply existing data processing

methods to new data sets. It provides comprehensive support for the entire data mining process, including the preparation of input data, the statistical evaluation of learning programs, and the visualization of learning input data. In addition to providing a large number of learning algorithms, WEKA also provides a wide range of pre-adaptive tools, operating various components through a unified user interface, and comparing different learning algorithms to find the most effective way to solve the problem.

Thesis Outline

The rest of this article is organized as follows:
Chapter 2 quickly and comprehensively introduces the current big data and the problems involved in using WEKA machine learning, focusing on some classic classification algorithms and decision tree algorithms to provide effective theoretical support for subsequent chapters. In the third chapter, the project generation process is described in depth. We introduced in detail the compression processing method of the original data set and the method used for model training and evaluation data set. Chapter 4 introduces and displays the model results and evaluation results obtained from each data set.Chapter 5 summarizes the results and suggestions of the project summary process for future research.

2
Problem Description

This chapter introduces the definition, functions and methods of large data mining, while the classification for a more detailed introduction to classical decision tree algorithm in-depth study.

Since the project is based on WEKA, we will also discover it including how to compress big data which apply in WEKA.

2.1. Big data

2.1.1. The Concept of Big data

Big data technology refers to the technology that quickly obtains valuable information from various types of huge amounts of data. The core of solving big data problems is big data technology. Big data, or huge amount of data, refers to the amount of data involved is so large that it cannot be retrieved, managed, processed, and organized within a reasonable time by the current mainstream software tools to help enterprises make business decisions More positive information. Compared with traditional data warehouse applications, big data analysis has the characteristics of large data volume and complex query analysis.

The three V's" "volume, velocity and variety, definition of big data originally coined by Doug Laney in 2001 to refer to the challenge of data management was quite in place to define big data for a few years. It basically interpreted big data as being a lot of data that is in a scattered form and needs to be processed quickly for proper interpretation. [1]
In August 2013, the definition was further enhanced to include "accuracy, variability, visualization and value", providing it with a new perspective.[2]With this new definition, big data now seems not only to describe itself in terms of its quantity but also to be further enhanced in terms of its interpretation and usability.

2.1.2. Data mining applications in Big data

Big data mainly includes data collection, data storage, data management, and data analysis and mining technologies. Here we focus on analyzing data mining technology. Data mining is a technique for discovering interesting patterns as well as descriptive and understandable models from large scale data. Data mining can be used to find correlations or patterns among dozens of fields in large relational database. [3]

It includes clustering analysis, classification, regression, and association rule learning, etc. [4]

Classification

Classification is to find the common characteristics of a group of data objects in the database and divide them into different classes according to the classification mode. The purpose is to map the data items in the database to a given class through the classification model. It can be applied to customer classification, customer buying trends and forecast attributes, such as a car retailer in accordance with customer preferences for cars divided into different classes, so that you can pamphlet new car to have this direct mail preferences of the customer, thereby increasing business opportunities greatly.
Several major kinds of classification algorithms in data mining are decision tree, k-nearest neighbor (KNN) classifier, Naive Bayes, Apriori and AdaBoost [3].

Regression

Regression analysis identifies dependence relationships among variables hidden by randomness [5], it generate a mapping data item to the function of a real-valued predictors found dependencies between variables or attributes, its main issues include trend data sequence Features, predictions of data series, and correlations between data, etc. It can be applied to all aspects of marketing, such as customer seeking, maintaining and preventing customer loss activities, product life cycle analysis, sales trend forecasting, and targeted promotional activities.

Cluster

Clustering can be considered the most important unsupervised learning problem. [5] It is to divide a group of data into several categories according to similarity and difference. The purpose is to make the similarity between data belonging to the same category as large as possible, and the similarity between data in different categories as small as possible. It can be applied to the classification of customer groups, customer background analysis, customer purchase trend prediction, market segmentation, etc.

Association rule

Association rules are rules that describe the relationship between data items in the database. According to the appearance of certain items in a transaction, it can be concluded that other items also appear in the same transaction as associated or interrelated data. In customer relationship management, by mining a large amount of data in the company's customer database, interesting relationships can be discovered

from a large number of records, and key factors affecting marketing effects can be analyzed and determined.

Web mining

There is no doubt that with the rapid development of the Internet and the Web's global popularity, the amount of information on the Web is very rich. Through mining the Web, a large amount of Web data can be used for analysis, collecting politics, economy, policy, technology and finance, various markets, competitors, supply and demand information, customers and other related information. The focus is on analysis and processing that have a significant impact on the company, it may have a major impact on external environmental information and internal business information, also can find out various problems and precursors that may lead to crises based on the analysis results.

2.1.3. The role of big data

Data is the core of the Internet. Whether it is a traditional industry or a new industry, whoever successfully integrates with the Internet first can obtain rules from big data and obtain opportunities for reform. [6] Use big data technology to mine and analyze data, reveal regular things, and put forward research conclusions and countermeasures. Big Data is a term that has invaded our daily world. From commercial applications to research in multiple fields, Big Data holds the promise of solving some of the world's most challenging problems. Also within academics, Big Data is popular in most disciplines, from the social sciences to psychology, geography, humanities (now also called digital humanities), and healthcare. Whether within society or enterprises, big data plays an important role in promoting economic development and maintaining social stability.

2.2. Classification problems

Classification problems are an important part of data mining processing. In the field of machine learning, classification problems are usually considered to belong to supervised learning. [7]That is to say, the goal of classification problems is to determine whether a new sample belongs to What kind of known sample class. According to the number of categories, the classification problem can be further divided into binary classification and multiclass classification.

2.2.1. Normal classification algorithms

Classification can predict discrete values and establish continuous value function models. Nowadays, many classification and prediction methods have been proposed in the technical fields of machine learning, statistics, neurobiology, etc. This section will

briefly introduce decision trees, Bayesian, and artificial neural network classification algorithms. [8]

2.2.1.1. Bayes

Bayes classification algorithm is a class of algorithms that use probability and statistics knowledge for classification, such as Naive Bayes algorithm. These algorithms mainly use Bayes' theorem to predict the probability that a sample of an unknown category belongs to each category, and select the most likely category as the final category of the sample. Due to the establishment of Bayes' theorem, it itself requires a strong assumption of conditional independence, and this assumption is often invalid in actual situations, so its classification accuracy will decrease. For this reason, there have been many Bayesian classification algorithms that reduce the assumption of independence, such as the TAN (Tree Augmented Naive Bayes) algorithm, which is implemented by increasing the association between attribute pairs on the basis of the Bayesian network structure.

2.2.1.2. Neural network

Artificial neural network is a mathematical model that uses a structure similar to the synaptic connections of brain nerves to process information.
In this model, a large number of nodes (neurons or units) are interconnected to form a network, that is, a "neural network" to achieve the purpose of processing information.
Neural networks usually need to be trained, and the training process is the learning process of the network.
Training changes the value of the connection weight of the network node to make it have a classification function, and the trained network can be used for object recognition. [9]

2.2.1.3. Decision tree

Decision tree is a tree structure, which can be a binary tree or a non-binary tree. It can also be regarded as a collection of if-else rules, or as a conditional probability distribution in a feature space.
The decision tree consists of the following elements: [10]
Root node: contains the complete set of samples
Internal node: corresponding characteristic attribute test
Leaf node: represents the result of the decision
When predicting, the attribute value is used to make judgments at the internal nodes of the tree. According to the judgment result, it can be determined where the branch node enters. The classification result is obtained until the leaf node is reached.

2.3. Decision tree algorithms

A decision tree is essentially a tree, but the meaning of its elements is different from that of a traditional tree. Each non-leaf node represents an attribute, each branch represents an output, and each leaf node represents one class. Decision Tree is a greedy algorithm, which constructs a decision tree in a recursive manner from top to bottom. Decision tree is a kind of supervised learning. According to the structure of the decision tree, the decision tree can be divided into binary decision tree and multi-branch tree. For example, some decision tree algorithms only produce binary trees (where each internal node forks exactly two branches), while other decision tree algorithms May produce non-binary trees. (For example, data of enumeration type is usually multi-tree). The following will briefly introduce the decision tree generation process, pruning technology and common decision tree algorithms.

2.3.1. Generation of the decision tree

The core issue of the decision tree algorithm is how to select attributes. After the decision model is established, a certain algorithm is used to prune the tree through the test set. Usually, attribute selection depends on information gain, information gain ratio, Gini coefficient and chi-square test.
The classic algorithms are ID3, C4.5, CART and CHAID in sequence. The construction of the decision tree is generally described as：

1. Operate the data training set according to user, starting with an empty tree, and then dividing it into appropriate categories based on attribute testing.
2. Acquire knowledge through training dataset, building decision model through recursion from top.
3. Through measurement optimization algorithm then calculate the possible division of each sample set and use a specific algorithm to prune the tree through the testing dataset.
4. After later pruning process to eliminate anomalies that may exist, eventually forming a complete decision tree.

2.3.2. Pruning of decision trees

If the established decision tree is too complex, then the decision tree will be difficult to understand. The more concise the decision tree, the smaller the overhead required to store the decision tree. In order to reduce the complexity of the tree, reducing the degree of fit model with training data set, improve computational efficiency and classification accuracy, not only to ensure the correctness of the algorithm, but also to ensure the efficiency of the algorithm, then the premise of a higher accuracy rate under the decision tree algorithm constructed as simple as possible. Among them, there are many

algorithms for simplifying decision trees, and the pruning operation is the most common algorithm. There are usually two pruning, pruning methods are first and pruning methods

Prepruning：

This operation is to "prune" the tree by stopping the construction of the decision tree in advance when the training set and the classification are not completely correct. Once the branching is stopped, the current node becomes a leaf node. It is realized by judging whether the current node needs to divide the training sample set contained in the node. It needs to specify a certain relevant threshold in advance, and the growth of the tree will stop when the relevant parameters reach the threshold.

Postpruning：

The postpruning method is to prune a "fully grown" decision tree. By deleting the branches of the nodes, pruning the nodes of the tree, and improving the accuracy of classification in the continuous pruning process, especially in the training data set When the noise level is high, the effect of the post-pruning method is quite obvious. This method is from bottom to top, and the node starts from the bottom of the leaf node and conforms to the pruning rule.

2.3.2 Classical decision tree algorithms

ID3

The ID3 algorithm was invented by Ross Quinlan. ID3 decision tree algorithm is a classic algorithm; it started from the root node. The root node is one of the best attributes. Then the property values are generated corresponding to each branch. Each branch has generated a new node. For the best attributes of the selection criteria, ID3 using the entropy-based definition of information gain to select the test attribute within the node. Entropy characterizes the purity of any sample set.
Pseudocode as follows:

```
ID3 (Examples, Target_Attribute, Attributes)
    Create a root node for the tree
    If all examples are positive, Return the single-node tree Root, with label = +.
    If all examples are negative, Return the single-node tree Root, with label = -.
    If number of predicting attributes is empty, then Return the single node tree Root,
    with label = most common value of the target attribute in the examples.
    Otherwise Begin
        A ← The Attribute that best classifies examples.
        Decision Tree attribute for Root = A.
        For each possible value, v_i, of A,
```

```
        Add a new tree branch below Root, corresponding to the test A = v_i.
        Let Examples(v_i) be the subset of examples that have the value v_i for A
        If Examples(v_i) is empty
            Then below this new branch add a leaf node with label = most common target
value in the examples
        Else below this new branch add the subtree ID3 (Examples(v_i), Target_Attribute,
Attributes – {A})
    End
    Return Root
```

C4.5

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The C4.5 algorithm considers all the possible tests that can split the data and selects a test that gives the best information gain (i.e. highest gain ratio). [11]
Pseudocode as follows:

```
C4.5
create a node N,
if samples has the same class, C, then
  return N as leaf node with class C label
  if list of attributes is empty then
  return N as leaf node with class label that is the most class in the samples.
  Choose test-attribute, that has the most GainRatio using attribute_selection_method
  give node N with test-attribute label
for each ai pada test-attribute
Add branch in node N to test-attribute = ai
Make partition for sample si from samples where test-attribute = ai;
  if si is empty then
  attach leaf node with the most class in samples
else attach node that generate by Generate_decision_tree (si, attributelist, test-attribute);
endfor
return N;
```

CART

CART is a classification and regression tree. It uses historical data with predefined categories. The purpose of the CART tree is to classify new observations into known categories.

CART is a binary tree, and each non-leaf node has two children, so the number of leaf nodes for the first subtree is one more than the number of non-leaf nodes. [12]
Pseudocode as follows:

```
CART
    info_gain, best_feature = find_best_split(D)
    if info_gain == 0: return Leaf(features)
    true_D, false_D = partition(D, best_feature)
    true_branch = build_tree(true_D)
    false_branch = build_tree(false_D)
    return Node(best_feature, true_branch, false_branch)
```

2.4. Implementation of decision tree algorithm in WEKA

Mainly introduces the development background, user interface, file type and main operation interface of WEKA platform. And used WEKA in the introduction to the Java environment API.
Here we take the classic dataset iris as an example for part of the demonstration.

2.4.1 What is WEKA

WEKA is called Waikato Environment for Knowledge Analysis. WEKA is a comprehensive data mining system developed by Waikato University.
Nowadays, WEKA is recognized as a landmark system in data mining and machine learning. [13]Because it is a complete and systematic data mining tool, it has a wide range of applications in the industry. WEKA is a public data mining platform implemented using java language. It can perform preprocessing operations on all current data types and can also evaluate the performance of its algorithms. Therefore, WEKA has unprecedented scalability and compatibility. Improvement. At the same time, WEKA integrates the most cutting-edge machine learning algorithms and data preprocessing tools, so users can add their own algorithms to WEKA according to actual needs, but the program needs to conform to WEKA's interface. The data mining methods integrated in WEKA include data preprocessing, classification, clustering, regression, association rules, and so on. WEKA can be obtained from the following website: http://www.cs.waikato.ac.nz/ml/WEKA.
Before implementing relevant data mining on the WEKA platform, we need to understand the user interface and data formats of the WEKA platform.

2.4.2 User Interface in WEK

WEKA is an open source data mining tool. It includes two ways for people to use. It can not only process data independently, but also add algorithm programs according to

actual needs. Among them, WEKA itself provides users with four user interfaces: Explorer, Experimenter, Knowledge Flow, Workbeach, Simple CLI. In this chapter, we will use WEKA's Explorer graphical user interface for operation experiments. Users can freely call various libraries in WEKA during the actual experiment of data mining. After downloading and installing, run WEKA, and first display the GUI Chooser interface.

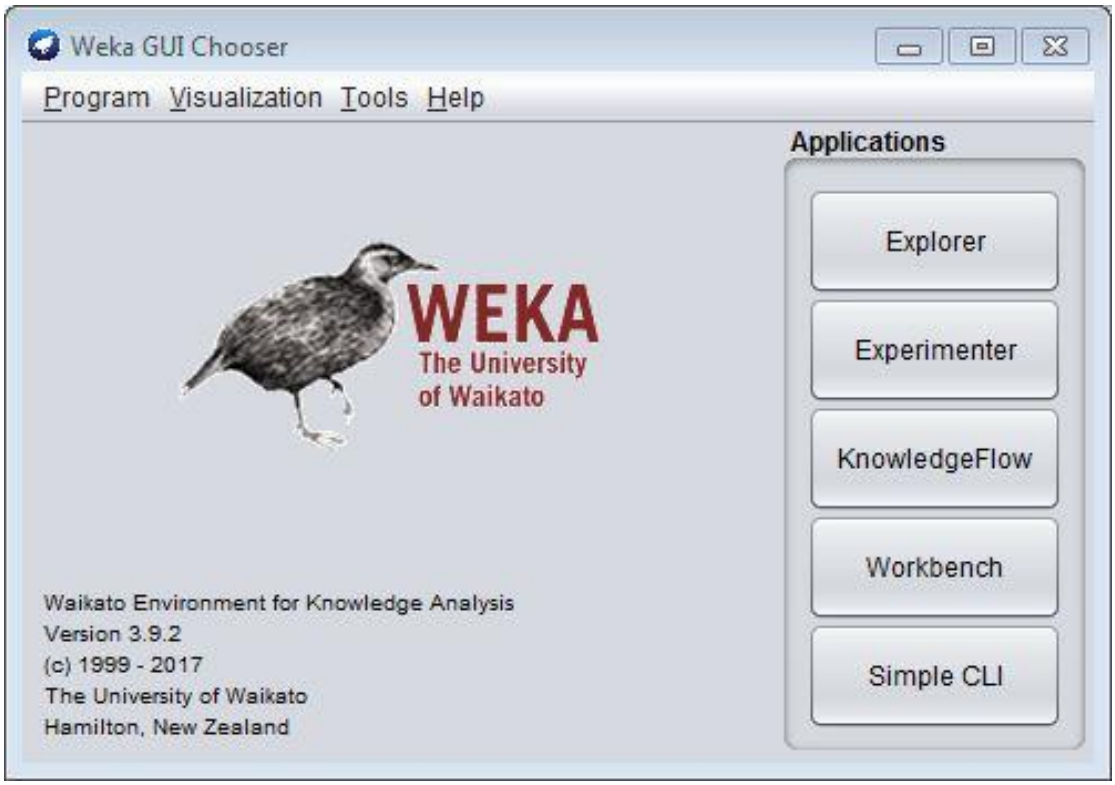

Figure 1 shown the WEKA interface

Explorer

Explorer has good interactivity. You can use all the data mining functions in WEKA through its graphical interface and convert all operations into familiar graphical modes, it has good interactivity.
The Explorer interface is divided into 6 different tabs:

Preprocess:

Load the data set and process the data into the form you want to use.

1. Open file: Obtain an instance set from the file;

2. Open URL: Obtain an instance set from the URL path;
3. Open DB: Open an instance set from the database;
4. Generate: Generate a manually fabricated data set

The function used here is to obtain the instance set by opening a file, because the instance set that needs to be used in the experiment has been stored in the computer in advance. The data set file can be imported into WEKA through the file selection button on this interface as a test data set. Before using other functions of WEKA, the corresponding data set must be imported through this label, and other label functions can be selected.

We choose IRIS dataset as a source, The IRIS Flower dataset is a famous dataset from statistics and is heavily borrowed by researchers in machine learning. It contains 150 instances (rows) and 4 attributes (columns) and a class attribute for the species of iris flower (one of setosa, versicolor, and virginica).

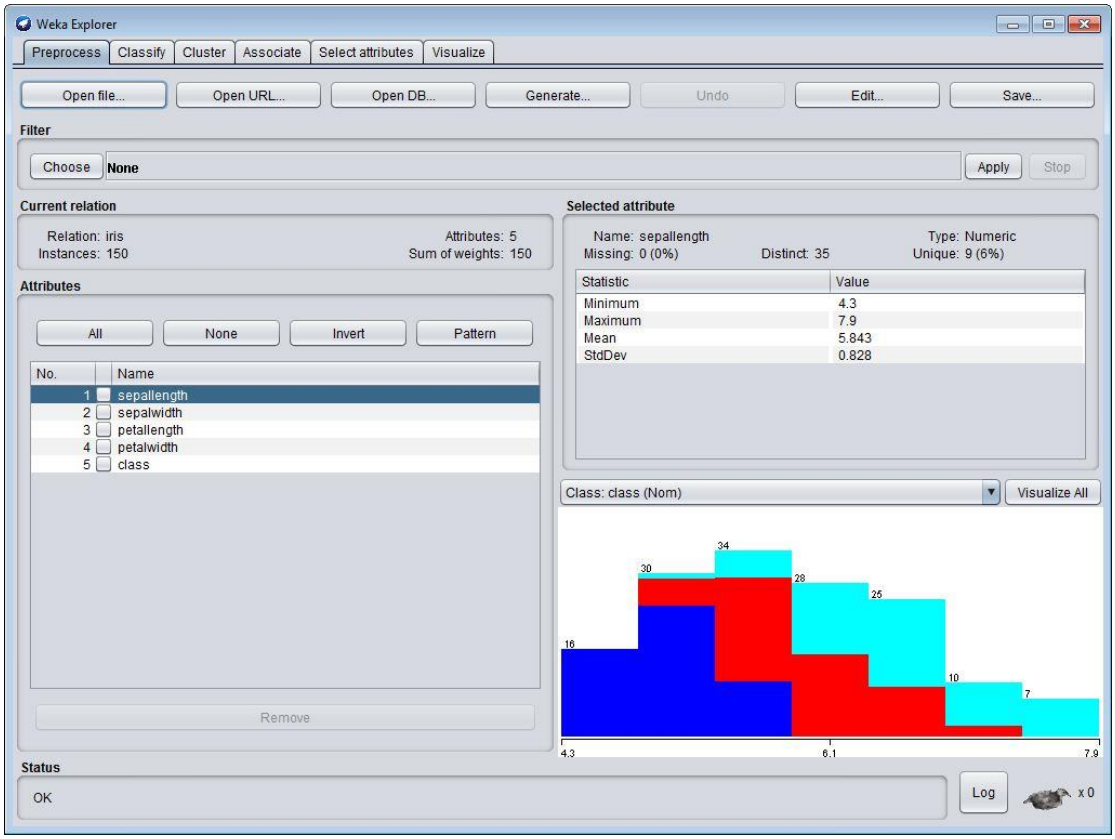

Figure 2 after open the IRIS.ARFF, what we can see from the WEKA interface

Classify:

Select and run classification and regression algorithms to perform operations on the data.

This research work mainly uses classify function in WEKA, so you must choose the data set before entering the function part.

The Choose button can select the integrated algorithm in WEKA. After selecting the test verification method and confirming the class attribute, click the Start button to start training. As shown in (pic), the default ZeroR algorithm is used.

The ZeroR algorithm selects the majority class in the dataset (all three species of iris are equally present in the data, so it picks the first one: setosa) and uses that to make all predictions. This is the baseline for the dataset and the measure by which all algorithms can be compared. You will also note that the test options selects Cross Validation by default with 10 folds. This means that the dataset is split into 10 parts: the first 9 are used to train the algorithm, and the 10th is used to assess the algorithm. This process is repeated, allowing each of the 10 parts of the split dataset a chance to be the held-out test set.

There are four model evaluation methods available in WEKA:

1. Use the training data set as the test data set. When using this method for model evaluation, the results can be directly evaluated. This method is suitable for any data set, but the reliability may be lower, because the test The data set is the same as the training data set, which will reduce the credibility of the results;
2. Provide an external test data set. When using this method for model evaluation, the user is required to specify a file containing the test set. This method is suitable for data sets that have been provided for ready-made tests;
3. Cross-validation method (setting the number of folds), when using this method for model evaluation, the training set is divided into the specified number of copies, one of which is used as the test data set and the rest is used as the training data set. After each one is trained in turn, the average of the results is obtained. This method is suitable for the case of relatively small data sets.
4. Percentage segmentation method (set percentage). When using this method for model evaluation, the training data set is directly divided, and part of the data set is reserved for testing. This method is suitable for large data sets.

Every training run results are displayed in the Classifier output below the blank box, you can view the content and results of the evaluation results of each run. At the bottom left of the interface, there is a small Result list, which is the history list of the results, but only one list is lit. Every time the user runs a classifier, the explorer will add a new history list.

When you need to see the results of the previous run of the display, just click the appropriate cross bar list, corresponding to the output of this run will appear in the classifier output window.

The result is 33%, as expected we based on ZeroR and corss-validation with 10 folds(3 classes, each equally represented, assigning one of the three to each prediction results in 33% classification accuracy). [14]

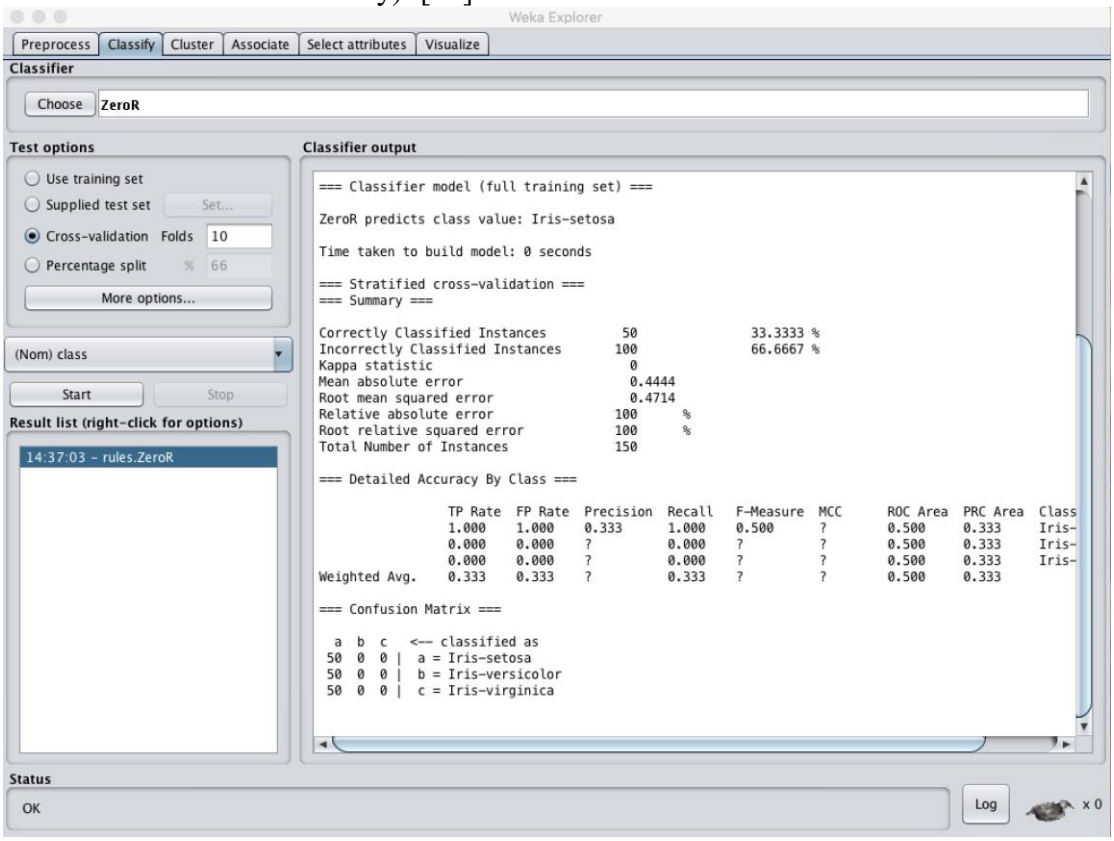

Figure 3 Normally the defaut set of WEKA is ZeroR and Cross-validation with Folds 10

Since we want to know more about decision tree classification, click the "Select" button in the "Classifier" section, then click "Tree", and then click the "J48" algorithm.
This is the implementation of the C4.8 algorithm in Java ("J" in Java, 48 in C4.8, hence the name of J48), and is a minor extension of the famous C4.5 algorithm. [14]

After running the J48 algorithm, you can note the results in the “Classifier output” section.Firstly, note the Classification Accuracy. You can see that the model achieved a result of 144/150 correct or 96%, which seems a lot better than the baseline of 33%. Secondly, look at the Confusion Matrix. You can see a table of actual classes compared to predicted classes and you can see that there was 1 error where an Iris-setosa was classified as an Iris-versicolor, 2 cases where Iris-virginica was classified as an Iris-versicolor, and 3 cases where an Iris-versicolor was classified as an Iris-setosa (a total of 6 errors). This table can help to explain the accuracy achieved by the algorithm. [14]

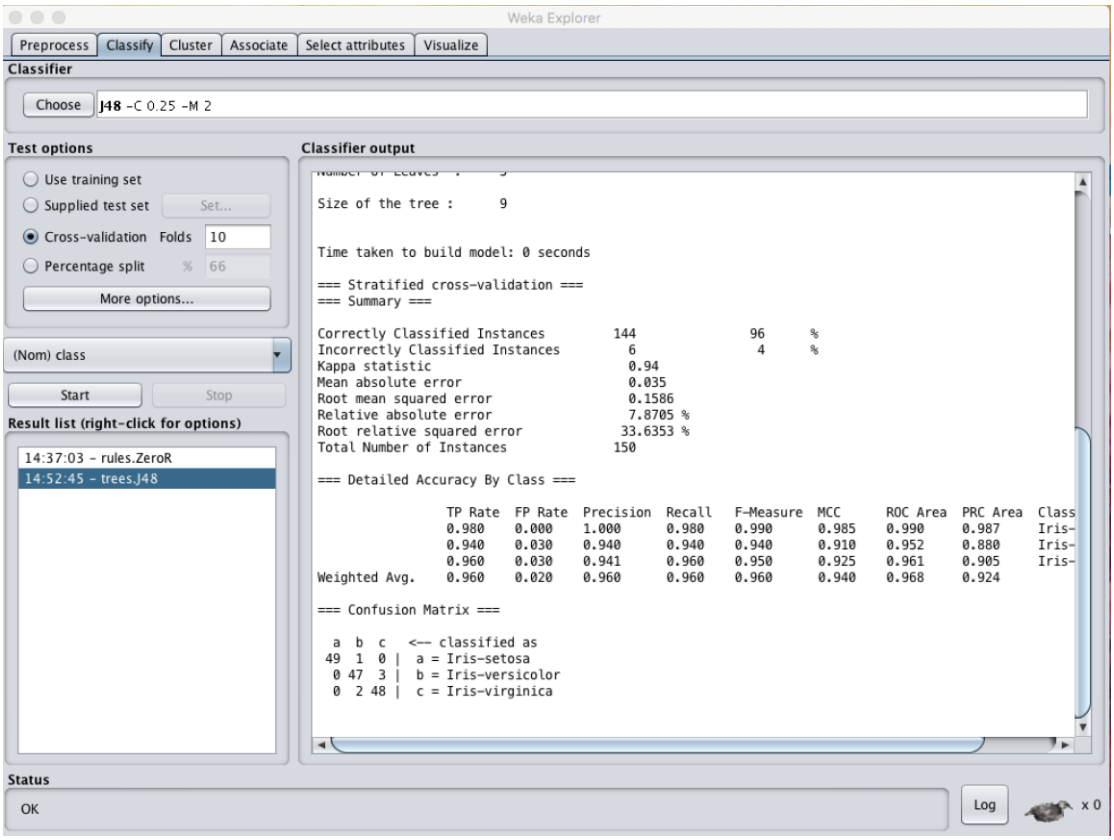

Figure 4 Run the Start, acquire the Result from the Classifier output

We can right click ,also save the model in local path.

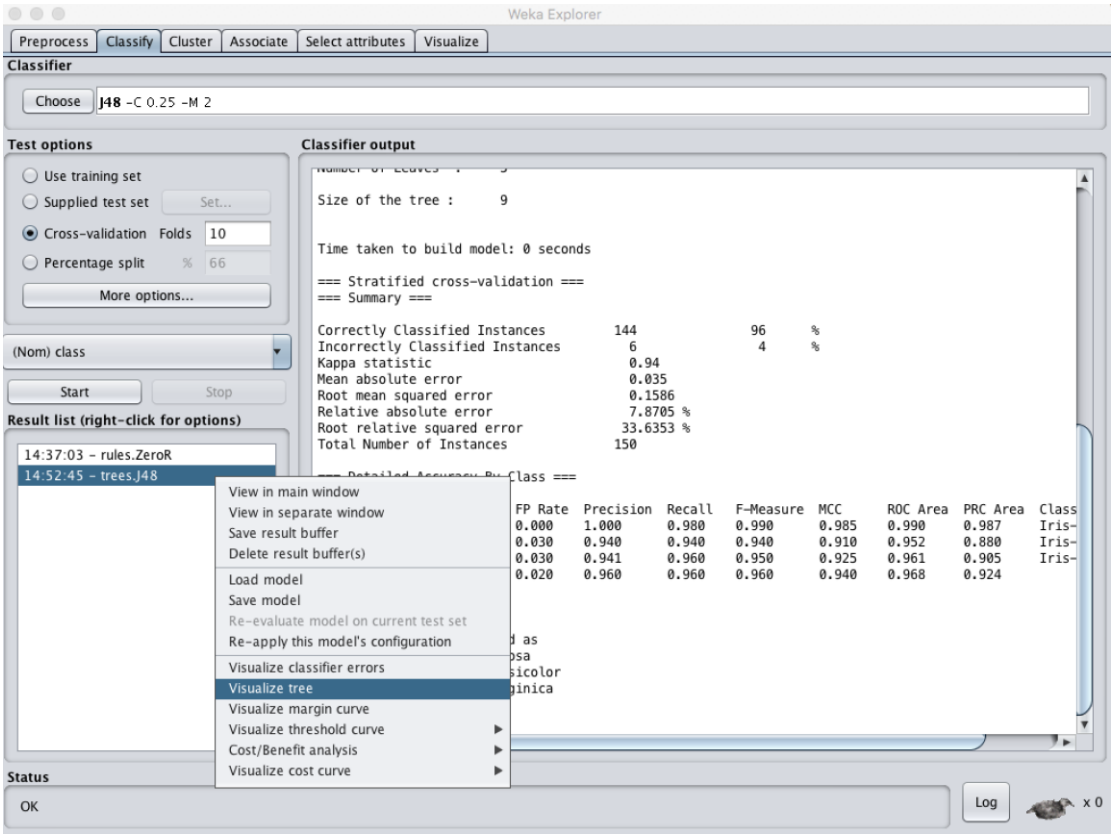

Figure 5 Right click can save the model in local path

Cluster: Select ：

Run a clustering algorithm on the data set.

Associate:

Run correlation algorithms to extract insights from data.

Select attributes:

Run an attribute selection algorithm on the data to select those attributes related to the features to be predicted.

At the same time, each label can use all the functions. For the time being, I only use some of the functions of preprocessing and classification. The other functions will be studied in depth in future research work. At the bottom of each panel there is a status bar (Status) and a log button (Log). The user can know what WEKA is doing now with the information displayed in the status bar. Click the Log button to open a text log,

which records all the activities that WEKA performed during the run and the time of each activity.

Visualize

Through the visualization panel, you can view the data set and select different attributes for the x-axis and y-axis. The instances are displayed as dots, with different colors for different categories. [14]

Experimenter

The Experimenter interface is designed to deal with classification and regression problems in some specific fields. These problems are usually deterministic, that is, to find out some parameter values that can obtain accurate results to feedback to the user. Of course we cannot answer this question solely by speculation. The user interface experimenter data processing can be automated, so that the data mining work simplification, this interface is also described with a certain degree of interactivity. It is also an important user interface in the WEKA platform. [15]

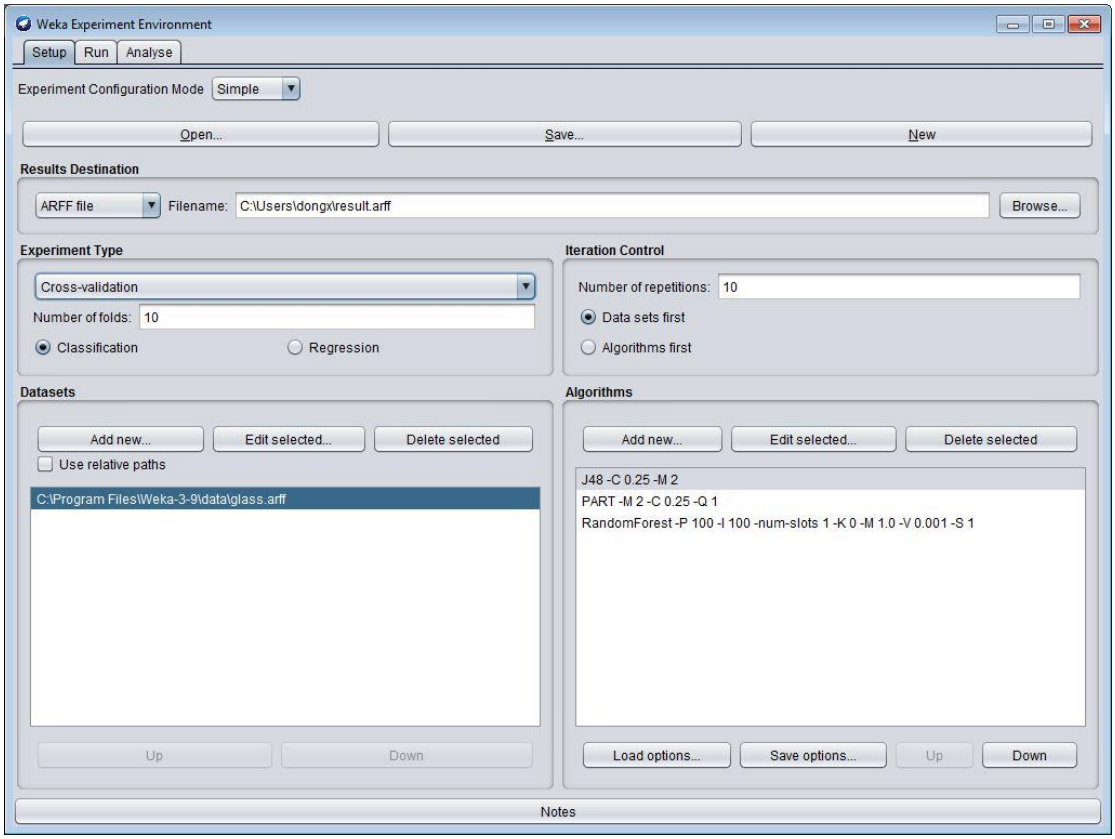

Figure 6 Experimenter interface

Knowledge Flow

The knowledge flow graphical user interface can set the flow data to be processed. The biggest difference is that large data sets that can handle large data sets, as well as data sources, learning algorithms, and evaluation methods, form a knowledge flow. The knowledge flow graphical user interface is also an important user interface in the WEKA platform.

Applied machine learning is a process and the Knowledge Flow interface allows you to graphically design that process and run the designs that you create. This includes the loading and transforming of input data, running of algorithms and the presentation of results.

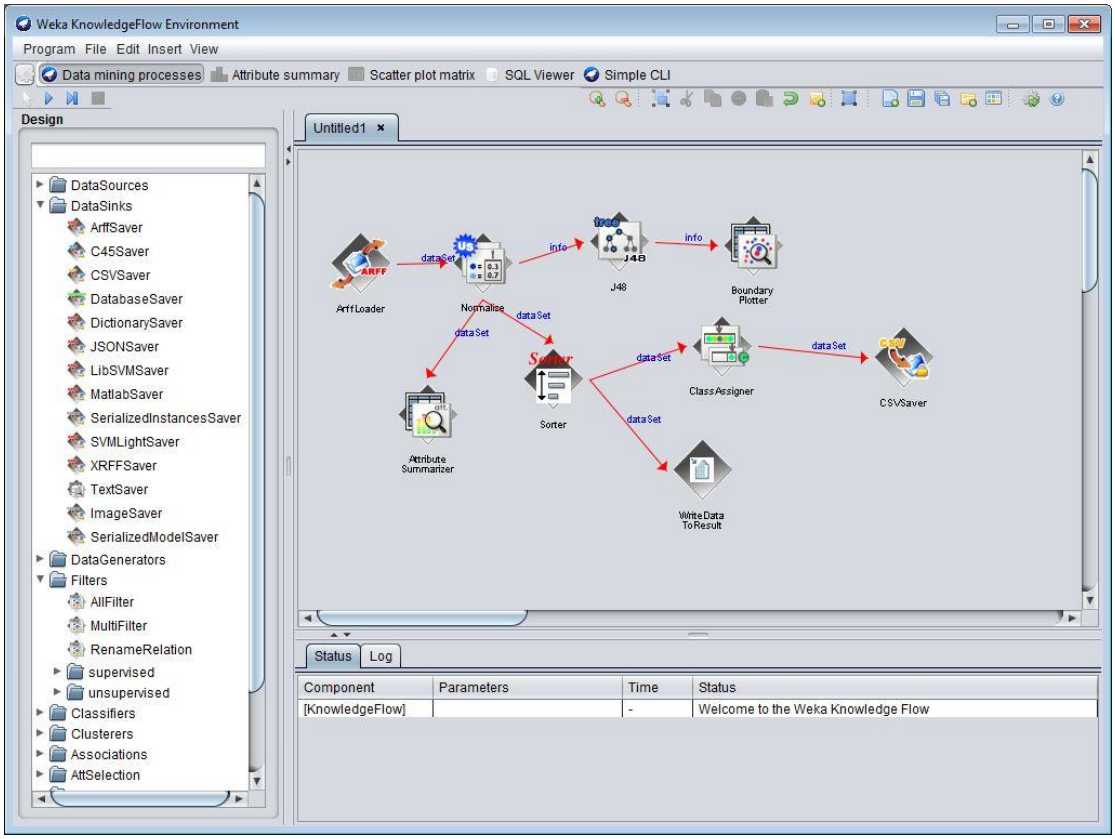

Figure 7 Knowledge Flow interface

Workbeach

Workbench provides a unified operation interface for other interfaces.

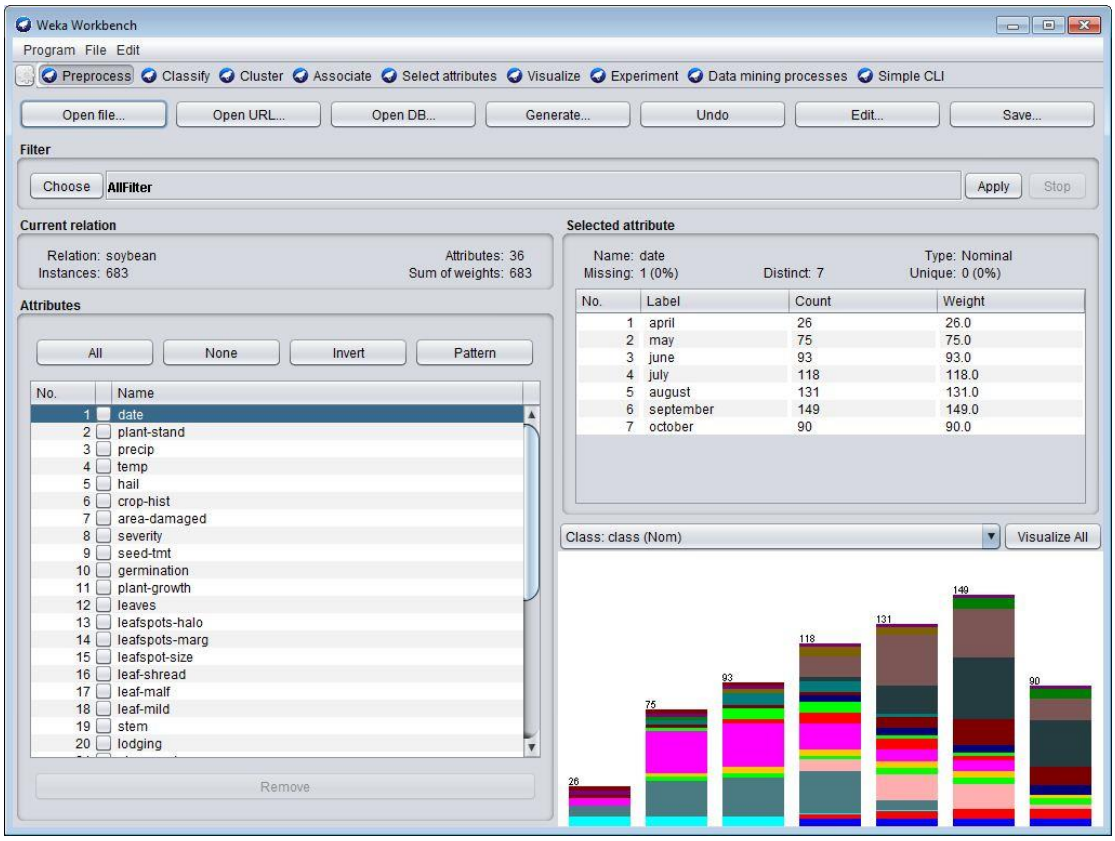

Figure 8 Workbeach interface

Simple CLI

SimpleCLI provides a simple command line interface that can call all WEKA classes.

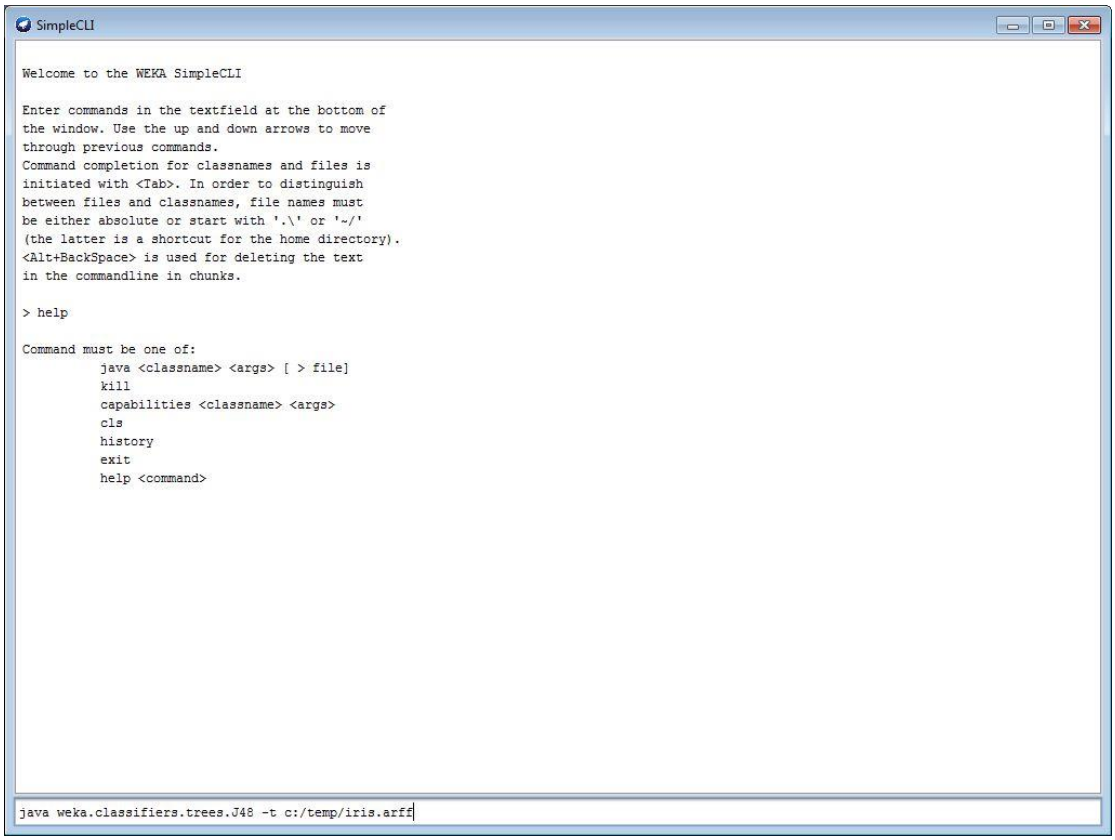

Figure 9 Simple CLI interface

2.4.3 ARFF file structure in WEKA

The format for storing data in WEKA is ARFF format. Before starting to study data mining problems, you first need to collect all the data into a set of examples. Centralize, integrate and clean up different data.

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the WEKA machine learning software. [16]

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

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

```
@RELATION iris
```

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

The Data of the ARFF file looks like the following:

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

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

@RELATION is a relation name that declares a data, the format is @relation<relation name>, relation name is a string, when it contains spaces, it must be quoted.

@ATTRIBUTE declares an attribute in the text, and these declared attributes are arranged in order, in the format @attribute<attribute name>, for example, @attribute today date declares the date attribute. In the same way, the attribute name is also a string. When it contains spaces, it must be quoted.

In the ARFF file, the line starting with @DATE is followed by instance data. One line is an instance. Each instance has several attributes, and the attributes are columns. Among them, the attributes are separated by ",".

2.4.4 How WEKA run in your java

Since WEKA is written in Java, the functions of WEKA can be done well in Java. After importing the WEKA source program into the development environment, you will find that WEKA's source code consists of many software packages, and you can view the functions of each software package according to the name of the software package. For example, the classifier is a software package related to classification, and the GUI is a software package related to a graphical interface. Here are some important software packages.

WEKA.core

The Core package is the core of WEKA's entire system. The classes in it can be read by almost all other classes. The most critical classes in the Core package mainly include Instance, Attribute, Instances and Utils. The following is an introduction to the important classes.

1. Instance.java
   Define classes, very important here, Used to handle instances. It has different storage methods for different types of attribute values. One of its objects contains attribute values contained in a specific instance.
2. Attribute.java
   Define the class, a very important class. An object represents an attribute and handles the attributes of an instance. Contains the attribute name, attribute type, and possible values of the attribute if it is a nominal or string attribute.
3. Instances.java
   Define class, very important class, store weighted instance set. One of its objects contains some instance sets arranged in order, which is a data set.
4. Utils.java
   Define the class and implement some static tool functions.
5. Version.java
   Define the class, store WEKA's version information, and implement the comparable interface, which contains some static variables. Used to compare the order of versions.
6. Queue.java
   The definition class, file input and output queue, can be serialized, there is an important internal class queuenode, which is the basic component of the queue, and can also be serialized.
7. Matrix.java
   Defining classes, manipulating floating-point matrices, and using confusion matrices when evaluating learning results are an important application of this class.

WEKA.core.converters

The main class used in this package is ConverterUtils.java, which converts the file to the format of the retrieved file. The function getLoaderForFile(File) is used to get the file through the absolute path of the loaded file. There is also a class AbstractFileLoader.java, which uses the makeOptionStr (AbstractFileLoader) function to select files and import abstract files, and uses runFileLoader (AbstractFileLoader, String[]) to obtain the data set in the file

WEKA.classifiers

The classification package contains implementation packages or classes of integrated classification and regression algorithms in WEKA. It contains not only classification algorithms, but also many regression algorithms that implement numerical predictions. It is used as a predictive interpretation of classes with continuous values. .
The most important class in this package is Classifier.java. The pass structure it defines is applicable to any learning scheme for number prediction or classification. The classifier contains three methods, buildClassifier(), classifyInstance() and distributionForInstance(). In the proper noun of object-oriented programming, the learning algorithm is represented by the subclass of the classifier, so these three methods are automatically inherited . Each method differs according to the way it builds the classifier and the specific way it classifies the instance. Inheriting these three methods provides a unified interface for building and using classifiers in Java code. Therefore, the same evaluation module can be used to evaluate the performance of any classifier in WEKA.

The CheckClassifier.java class queries the available classification algorithms and displays them in the list, which is implemented by the setOptions (String []) function and the setClassifier (Classifier) function.
Class Evaluation.java evaluates the classification and regression results of the algorithm in the classifier, judges its accuracy, and evaluates the effect of the algorithm through other parameters.

WEKA.clusterers

Which contains an implementation of various unsupervised learning methods, included in the category Clusterer contains buildCluster (), clusterInstance () and distribution () methods, these three methods are all clustering algorithms have inherited method.

WEKA.gui

The GUIChooser.java class is the initial package of the entire GUI interface. The entire WEKA function starts here. After running, it will display the main functional interface of the WEKA program. The entire program package is the basis of all WEKA graphical interfaces.

WEKA.gui.explorer

The setInstancesFromFile (AbstractFileLoader) function in the PreprocessPanel.java class creates a thread to obtain the data collection in the file and preprocess the data. The ExplorerDefaults.java class displays the content of the default Explorer interface by reading the content in the configuration file, and obtains the content of each tab

through the getTabs() function. Before selecting a file, other labels are not available except for the preprocessed data labels. After selecting the file, convert it to a usable state accordingly.Class Explorer.java uses the function explorer() in the preprocessing function to determine whether the data can be used and whether to restore other tags to a usable state.All visual interface portion display content classification function section is displayed in the class ClassifierPanel.java. The most important thing to note is that the area represented by m_OutText displays the final output result, which can be viewed in the final visual display interface.

2.4.5 Common classification algorithms in WEKA

Classification is a very important and widely used algorithm in the field of machine learning. The purpose of classification is learning classifier that obtains classification function or classification model, mapped to a category classified by the classification function or model can be classified data. According to classification purposes, you can use the automatic classification derived from historical data to a promotional description given data to predict future data. It should be noted that regression can also be used for prediction. The difference is that the output of classification is a discrete category value, while the output of regression is a continuous or ordered value. Since the algorithms we use are of the class, so here only discuss the classification.
On the "Classification" tab panel introduced above, users can select the classifier algorithm embedded in WEKA to train by "Select", and then click "Select" to select the desired algorithm. In WEKA, all classifier algorithm that can be used is divided into Bayesian classifier, tree, rules, functions, lazy classifier and a variety of other classifiers. Because the project is based on the decision tree, so we focus on tree algorithm. Click on the tree, there will be the following options: [17]

1. DecisionStump
   Class for building and using a decision stump.
2. HoeffdingTree
   A Hoeffding tree (VFDT) is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time.
3. J48
   Class for generating a pruned or unpruned C4.5 decision tree.
4. LMT
   Classifier for building 'logistic model trees', which are classification trees with logistic regression functions at the leaves.
5. M5P
   M5Base.
6. RandomForest
   Class for constructing a forest of random trees.
7. RandomTree
   Class for constructing a tree that considers K randomly chosen attributes at each node.

8. REPTree
   Fast decision tree learner.

The C4.5 algorithm for building a decision tree is implemented as a classifier in WEKA, named J48.
In subsequent chapters, we will focus on how to use J48 in the project.

2.5 Data compression for big data

Compressing big data can help address these demands by reducing the amount of storage and bandwidth required for data sets. Compression can also remove irrelevant or redundant data, making analysis and processing easier and faster
There are many ways of data compression, and there are different data compression methods (that is, encoding methods) for data with different characteristics. The following are classified from several aspects:

1. Instant compression and non-instant compression
   For example, to make an IP phone call is to convert the voice signal into a digital signal, compress it at the same time, and then transmit it through the Internet. This data compression process is carried out in real time. Real-time compression is generally used in the transmission of video and audio data. Instant compression is commonly used in specialized hardware devices, such as compression cards.
   Non-instant compression is often used by computer users. This compression is performed when needed, and has no instantaneity. For example, compress a picture, an article, a piece of music, etc. Non-immediate compression generally does not require special equipment, just install and use the corresponding compression software directly in the computer.

2. Data compression and file compression
   In fact, data compression includes file compression. Data originally refers to any digitized information, including various files used in computers, but sometimes data refers to time-based data. These data are often collected and instantaneously. Processed or transmitted. File compression refers to the compression of data that will be stored on physical media such as disks, such as the compression of an article data, a piece of music data, and a piece of program coded data.

3. Lossless compression and lossy compression
   Lossless compression uses statistical redundancy of data for compression. The theoretical limit of data statistical redundancy is 2:1 to 5:1, so the compression ratio of lossless compression is generally lower. This kind of method is widely used in the compression of text data, programs and image data in special applications that require accurate storage of data. The lossy compression method takes advantage of the insensitivity of human vision and hearing to certain frequency components in images and sounds, allowing certain information to be lost during the compression process. Although the original data cannot be completely restored, the lost part has

less impact on the understanding of the original image, but in exchange for a larger compression ratio. Lossy compression is widely used in the compression of voice, image and video data.

Compression of big data is becoming key to maintaining costs and productivity for many businesses. Thankfully, new technologies and algorithms are being researched and created to address this need.

3
Implementation

In this project, we mainly rely on the open source of WEKA and WEKA API to complete data processing. A good data mining concept should start from the definition of the problem, the collection and preprocessing of data, the process of data mining, and the evaluation of the mining results. Due to the diversity and complexity of the data. Choosing the right algorithm and data model will determine how to classify the data and the accuracy of the classification. Before classification, reasonable preprocessing of the data can improve the accuracy and efficiency of the classification.

Reasonably we select and divide the original data into training data and test data through JAVA language which defined by ourselves. WEKA plays an important role here, because each classification algorithms and operation basically rely on it. Obtain the desired data set and original training model after each classification, also at the same time, build compressed files and decision tree path (decision tree rule) that conform to structured data through JAVA language. Obviously, the compact representation of the data set must be in the same format as the original data, which helps to improve the efficiency of the classifier. In this project, what we want most is two sets of models, they correspond to the model created by the original data and the model created based on the compressed file. After that, we can easily get two sets of model classification accuracy of test data sets, as well as various parameters we want. Relying on these data, we can more easily analyze the classification impact of lossy compression in the structured data.

Since many data sets have been selected, in this chapter, we will extend the project with data sets such as Car-Evaluation in the UCI data set as an example.

Below, let us expand our specific experimental procedures.

3.1. Preparation of data

In order to get the experimental results closer to real life, we search and collect data sets through UCI data sets.
The UCI database is a machine learning database proposed by the University of California Irvine. This database currently has 488 data sets, the number of which is still increasing. The UCI data set is a commonly used standard test data sets. [18]
When searching a data set, we need to pay attention to the attributes of the data. Attributes are data fields that represent characteristics of data objects. An attribute vector (or feature vector) is a set of attributes used to describe a given object. There are different types of attributes: nominal attributes, binary attributes, ordinal attributes, numerical attributes, discrete attributes and continuous attributes.
Since this experiment is based on the analysis of decision trees, we choose a dataset with classification attributes from the UCI dataset. At the same time, in order to improve the accuracy of classification and reduce the possibility of overfitting, in the classification data set, we only select the data sets with categorial attributes . If there is data with numerical attributes in the data set, the data should be discretized first.

3.1.1 Convert data to ARFF format

Each data file (*.data) of UCI data contains records of many individual samples described in the form of "attribute-value" pairs. The corresponding *.info file contains a large amount of documentation. As a supplement to the data set and domain knowledge, some useful information when using this data set is included in the utilities directory.

Since the data sets downloaded from UCI are all in (.data) format, but the use of WEKA requires (.ARFF) files, so the data set must be converted into the (.ARFF) format before apply to WEKA.

In order to better understand the data and make better planning for the subsequent classification, on the homepage of Car-Evaluation, we can also find the Data Set Information.

Specific steps are as follows, we use Car-Evaluation data set as a example：

1. Download the target data set in UCI
2. Click the data folder and download the the data file with .data format, also including the file car.c45-names.
3. Rename both suffix of two files to format(.ARFF).
4. Open both of them, through the original file car.c45-names, we can easily point out the attributes of this car data set. This data is a crucial part of building the (.ARFF) file. But there is an easy way to find these data via data set page on UCI website.

The follow shown the class values of CAR dataset：

```
Class Values:
unacc, acc, good, vgood
Attributes:
buying: vhigh, high, med, low.
maint: vhigh, high, med, low.
doors: 2, 3, 4, 5more.
persons: 2, 4, more.
lug_boot: small, med, big.
safety: low, med, high.
```

5. Open the file which store the data and rename as car.ARFF.
6. Rebuild the car.ARFF file, copy and paste the attributes obtained in step 4. In the above, we also mentioned the format of the ARFF file. According to the correct format, reconstruct the car data. Figure 3.4 shown the correctly car data set finally.

3.2. Split the data

Since the model must be trained and tested through the data set, the original data set must be divided into a training data set and a test data set. The split training set can represent the entire data set, and the features of the selected test set should be the same as those of the training set.
The purpose of the training set is to train the model, and the test set is used to test the trained model. Both are essential. It is worth noting that the test set is only suitable for testing training models.
To improve the accuracy of training and testing, divide as much data as possible into training data sets. Here, through the JAVA environment and language, the data in the .ARFF file is divided into 66% training data set and 33% test data set. In addition, both the training data set and the test data set should retain the format of the original file, in other words, they must contain the same attribute definitions as the original data file.

In the java environment, there are two simple ways to make the WEKA API.。

1.3. Use Maven to import the WEKA package.
Create a Maven project and add the corresponding code to the Maven pom.xml file.
2.3. Download WEKA.jar and add WEKA.jar to the build path
For example, the component we are using here has an instance to load the original data set.

After importing the original data set, by calling the randomize function and the Math function, 33% and 66% of the original data are allocated to the test set and the training set, respectively. And save it for future work.

F

For example, in car data set, we have 1728 instances with the relation “car” and 7 attributes which are buying, maint, doors, persons, lug_boot, satety, and class.

Inside the class, that can be classify by unacc, acc, good, vgood. After splitting data successfully, the training set which we named as *Car-trainset.ARFF* should has 1140 instances, the test set which we named as cart-testset.ARFF should has 570 instances.Boths of them has the same format as car.ARFF. That means they all have 7 attributes which are buying, maint, doors, persons, lug_boot, satety, and class,but the relation can defined by ourself.

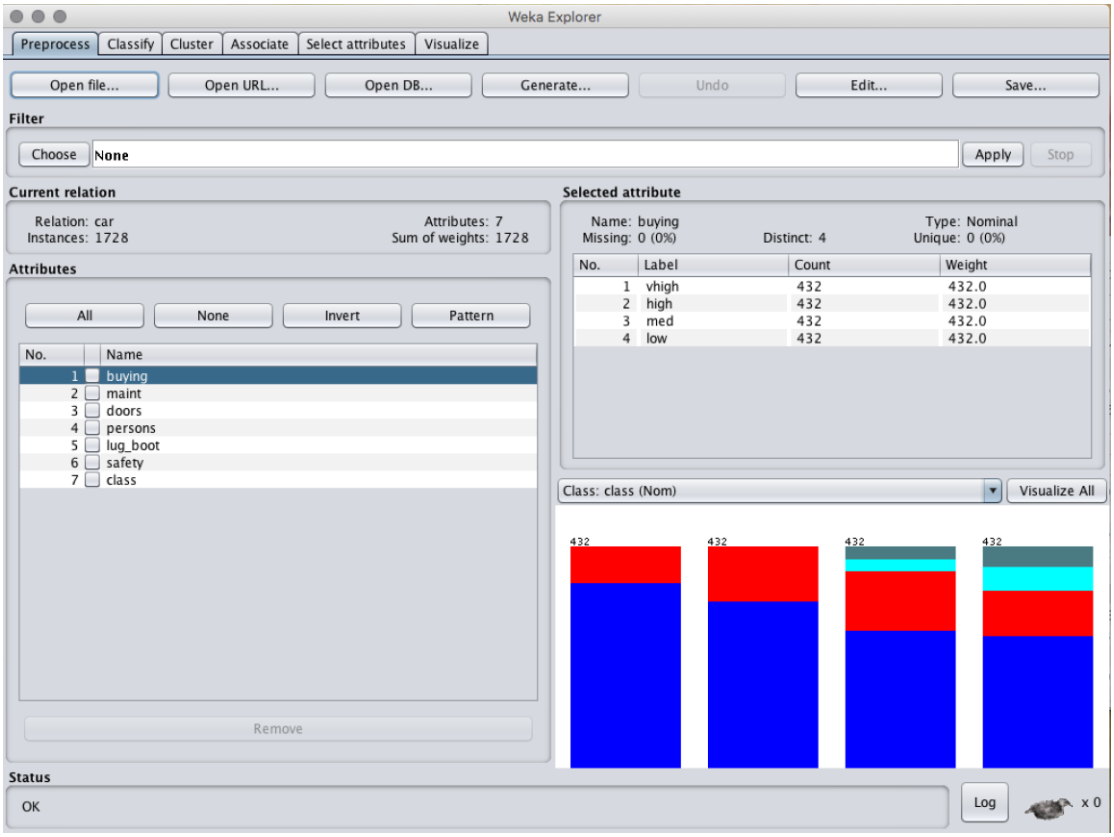

Figure 10 After open the CAR.ARFF

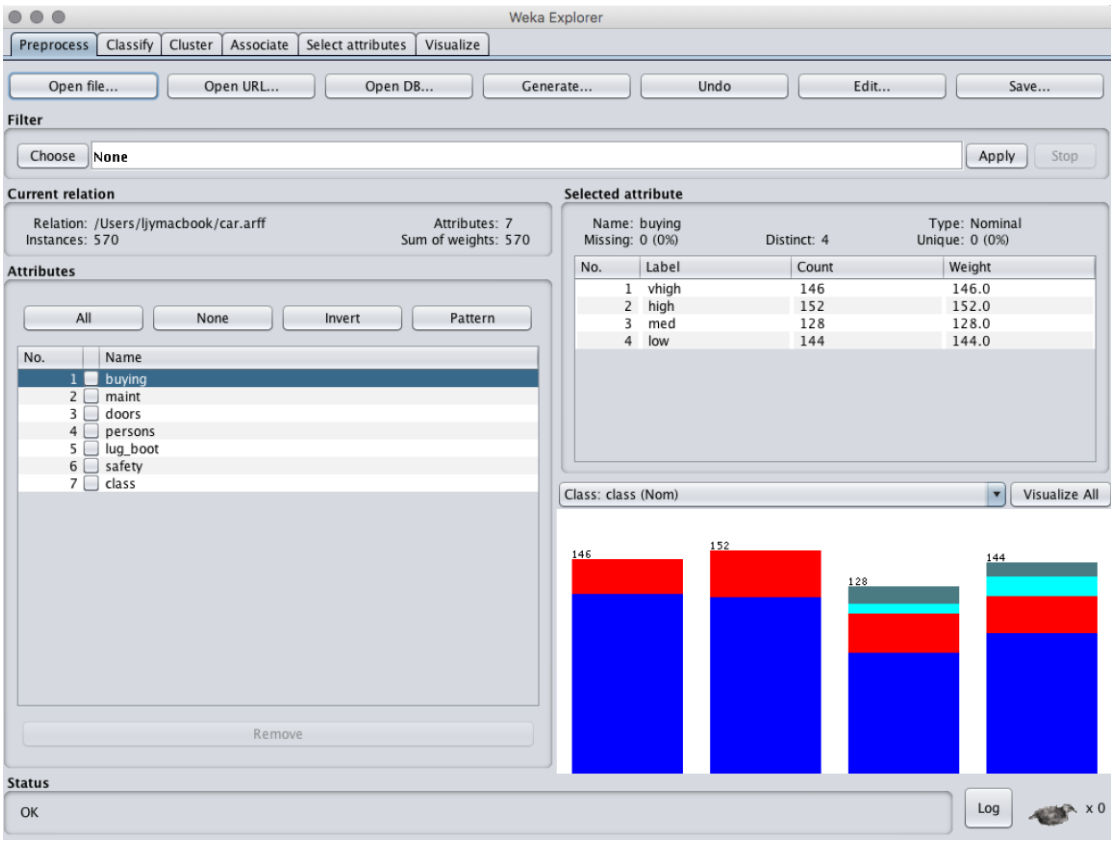

Figure 11 Open the CARt-testset.ARFF

Of course, the original data set can also be quickly segmented through WEKA, just by operating on the WEKA interface. After opening the data set, the only need is to use the RemovePercentage filter.

3.3. Training the original data model (first model)

This step is used to create a decision tree model of the training data set, here, the data we input is *Car-trainset.ARFF* we got from the chapter 3.2.

Open the training dataset by WEKA

Click Open file to open a dialog box that allows to browse the data files on the local file system and find the saved training data set car-trainset.ARFF.

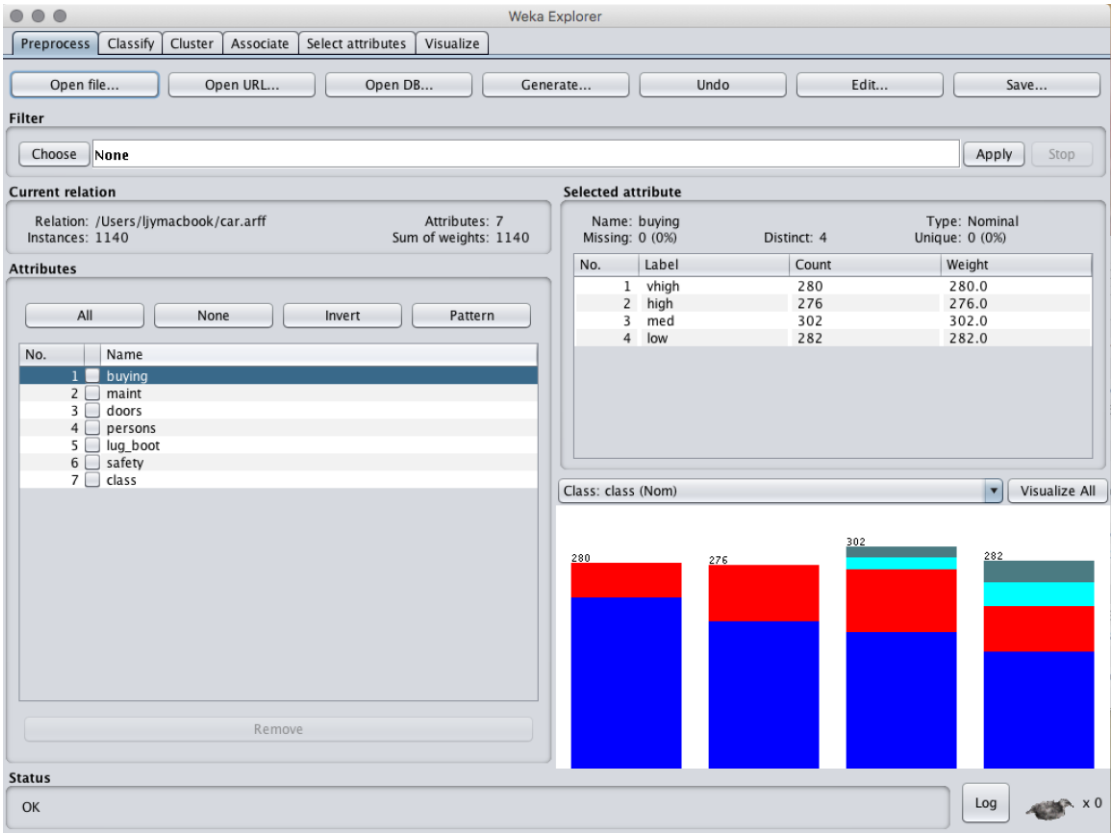

Figure 12 Open the car-trainset.ARFF in weka

Select classifier

Now that we loaded a dataset which names car-trainset.ARFF, it's time to choose a machine learning algorithm to model the problem and make predictions.
At the top of the classify page is the Classifier column. Click the Choose button to select the classifiers available in WEKA.
We will note that the "ZeroR" algorithm is selected by default.
Since here need to implement decision tree classification, click Trees here.

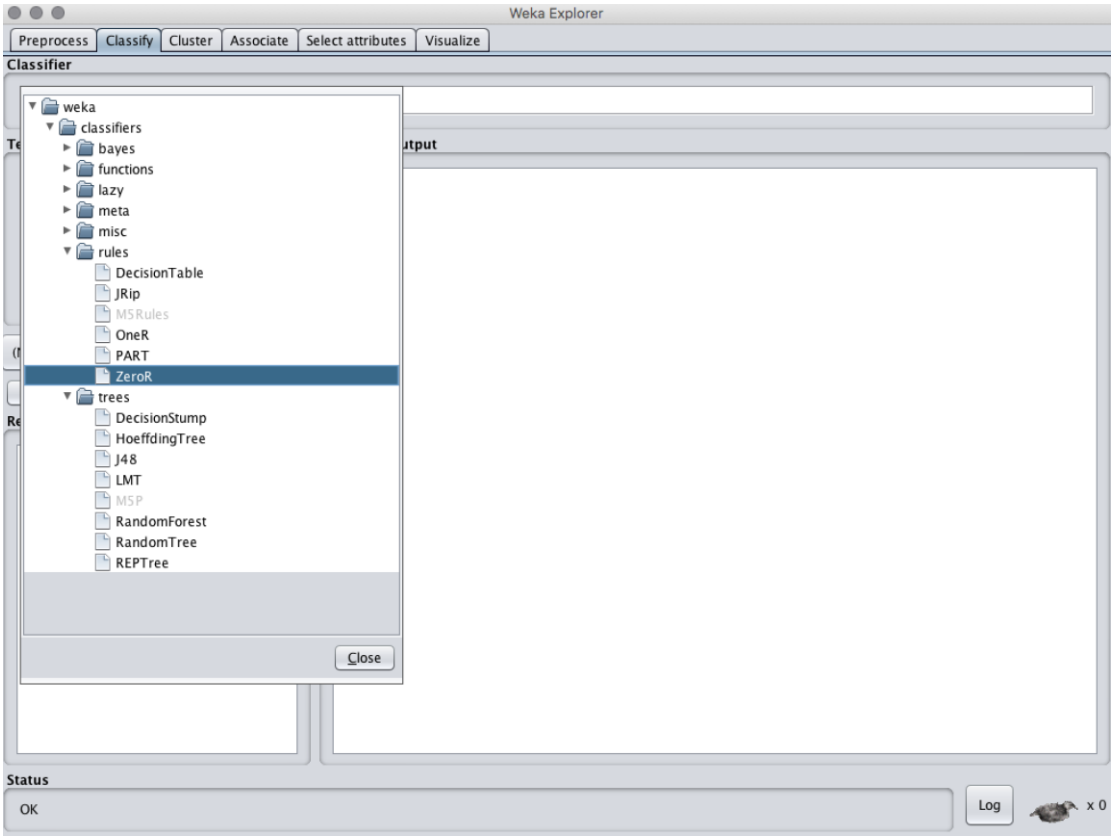

Figure 13 Choose the Classify

Select algorithm

In the algorithm part, we want use all attributes to predicate the class label.The basically problem is Nonlinear algorithms do not make strong assumptions about the relationship between the input attributes and the output attribute being predicted.
The preferred algorithms are:

1. Naive Bayes: bayes.NaiveBayes
2. Decision Tree (specifically the C4.5 variety): trees.J48
3. k-Nearest Neighbors (also called KNN: lazy.IBk
4. Support Vector Machines (also called SVM): functions.SMO
5. Neural Network: functions.MultilayerPerceptron

we choose the classic algorithm C4.5, the corresponding J48 algorithm in WEKA firstly.

Set the Options

Since we mentioned the first two options before, we generally don't choose them, so we choose from cross-validation and percentage split. Because of the limited test data, where we choose the default cross-validation, Folds 10. Of course, we can also set some additional parameters, such as when clicking on more options, you can set Output model, Output evaluation measures, Cost-sensitive evaluation, etc.

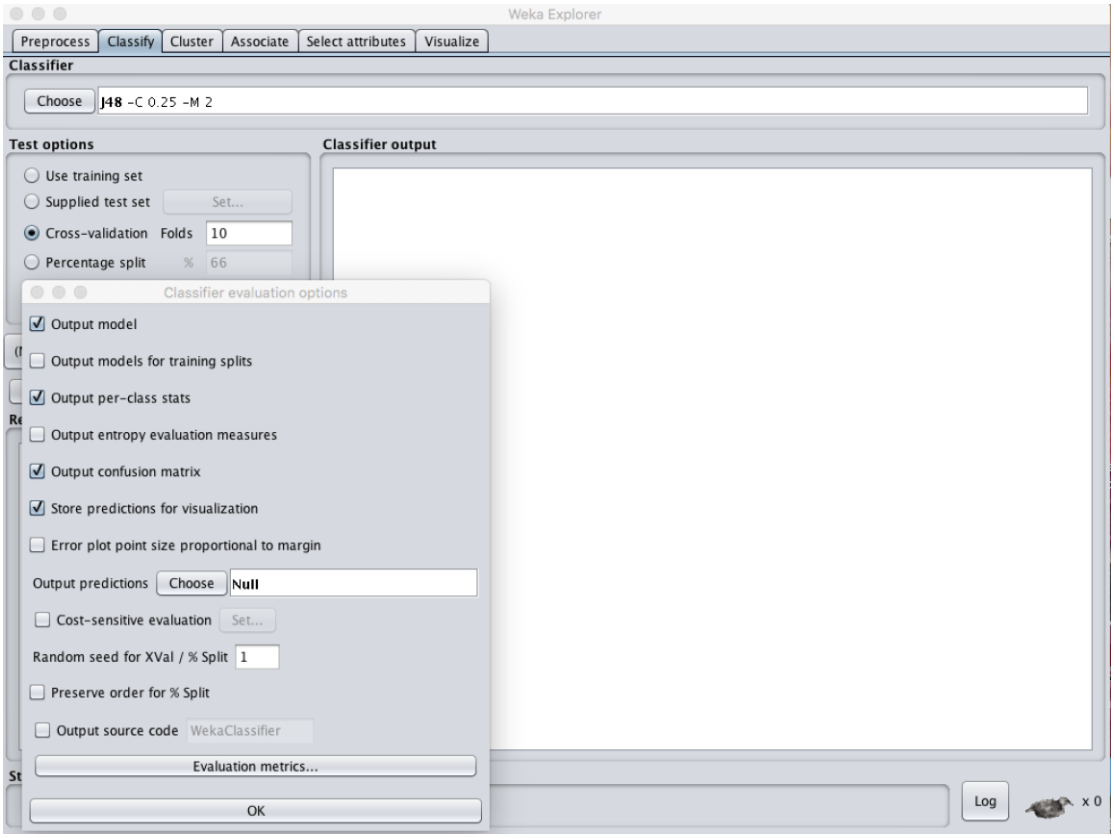

Figure 14 choose options

Set the classes attribute

The classifier in WEKA is designed to be trained to predict a class attribute, which is the target of prediction. Some classifiers can only be used to learn class attributes of sub-types; some can only be used to learn numerical class attributes (regression problems); and others can learn both.
By default, the last attribute in the data set is regarded as the class attribute. If you want to train a classifier to predict a different attribute, you can set it by clicking Class. Since the class has been set in the original data set, there is no need to set it here.

Start training the model

After the classifier we choose J48, test options we set as cross-validation with 10 folds and class attributes are set, click the Start button to start the learning process.
When the classifier is busy training, the bird below will move around. We can stop the training process at any time by clicking the Stop button.
After the training is complete, several things will happen. The Classifier output area on the right will be filled with some text describing the results of training and testing. A new entry will appear in the Result list column. Next we will observe this list of results, but let's study the output text first.

Output

The text in the Classifier output area has a scroll bar to browse the results. You can save the output results if necessary. Of course, you can enlarge the Explorer window to get a larger display area. The output result can be divided into several parts:

1. Run information.

Gives a list of options for the learning algorithm. Including the relationship name, attribute, instance and test mode involved in the learning process.

2. Classifier model.

A classification model based on the entire training set expressed in text.

The results of the selected test mode can be broken down into the following parts:

3. Summary.

A list of statistics describing how accurately the classifier predicts the class attribute under the specified test mode.

4. Detailed Accuracy By Class.

gives a more detailed description of the prediction accuracy of each class.

5. Confusion Matrix.

Show the number of instances of each class in the prediction result. The rows of the matrix are the actual classes, the columns of the matrix are the predicted classes, and the matrix elements are the number of corresponding test samples.

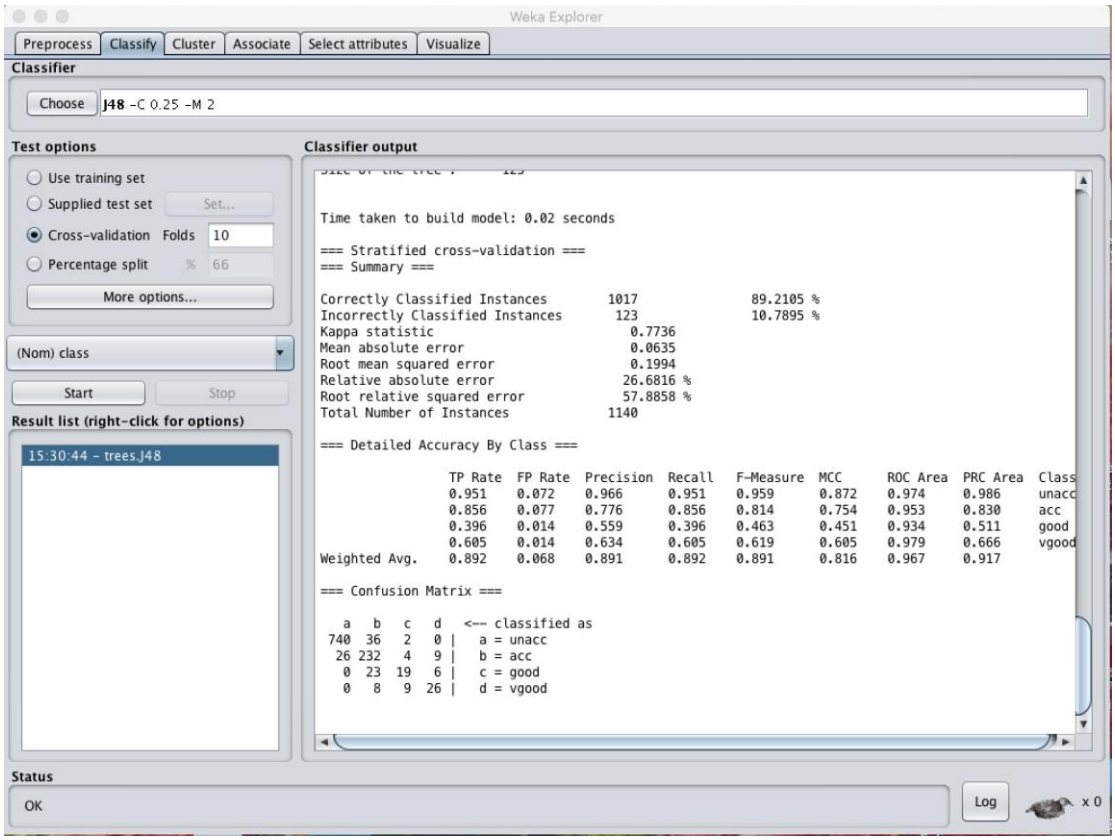

Figure 15 shown the resukt of first model of CAR dataset

In this picture, we can see the information we listed above. Since we are most concerned about the accuracy of the model classification, we found 1017 correctly classified examples here, and 123 correctly classified examples. The accuracy It is 89.2105%.
After we tried different classification tree algorithms and set different options, we compared the accuracy of each model to find the best one as the final model.
For recording clarity, we create abbreviations for the two options.
For example: cross-validation with 10 folds we write as Cv10.c
Percentage split with number 10 we write as Ps10.

Table 1 different Algorithms with different Options in training first model for the CAR dataset

ID	*Algorithms*	*Options*	*Accuracy*
1	*Hoeffding tree*	*Cv10*	*84.4737 %*
2	*Hoeffding tree*	*Cv30*	*84.6491 %*
3	*Hoeffding tree*	*Ps10*	*73.8791 %*
4	*Hoeffding tree*	*Ps30*	*82.7068 %*
5	*J48*	*Cv10*	*89.2105 %*
6	*J48*	*Cv20*	*89.6491 %*
8	*J48*	*Ps10*	*72.9045 %*
9	*J48*	*Ps30*	*82.0802 %*
10	*LMT*	*Cv10*	*95.1754 %*
11	*LMT*	*Cv20*	*95.4386 %*
12	*LMT*	*Ps10*	*76.9006 %*
13	*LMT*	*Ps30*	*89.599 %*

In these simple examples, we can see that LMT with the highest accuracy when cross-validation set folds in 20, which means that this is the most optimized model we have found so far.
Let's check the decision tree generated by this algorithm.

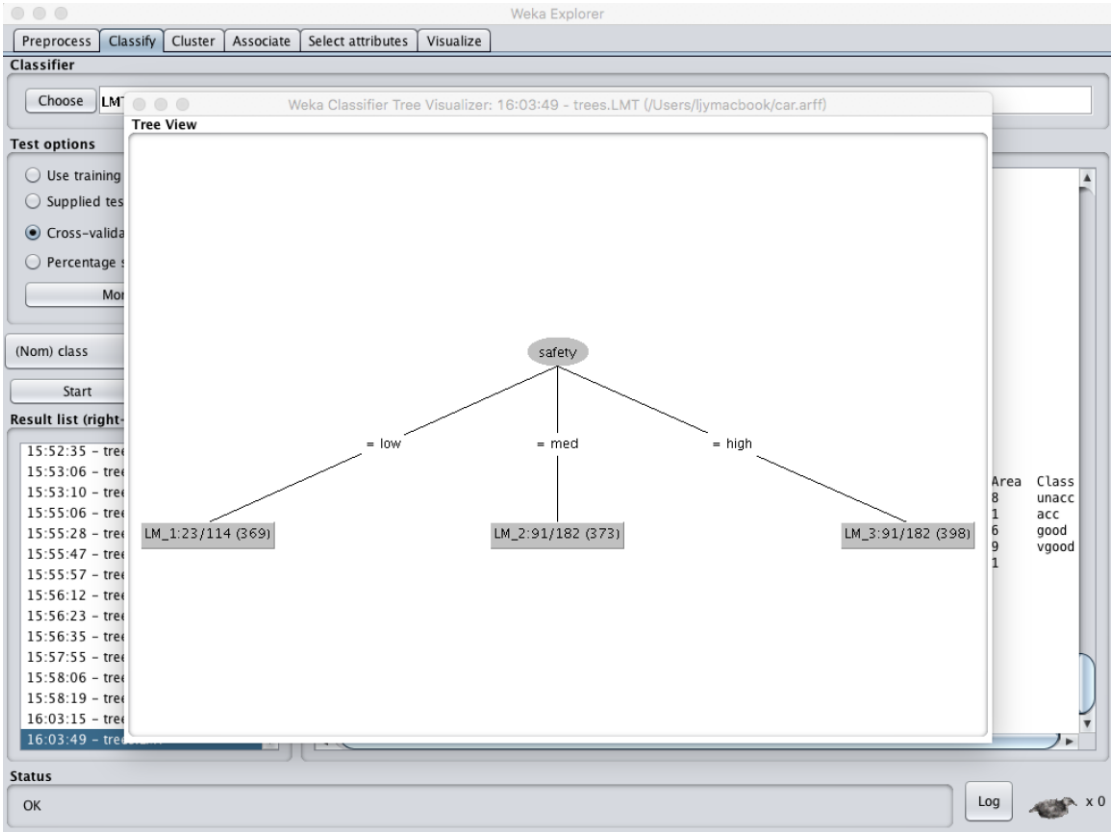

Figure 16 The Tree of Algorithm LMT

Unfortunately, it is not in line with our goal, because the output of this decision tree cannot provide useful support for the follow-up, so we can only give up it and choice J48 with cross-validation with 20 folds.

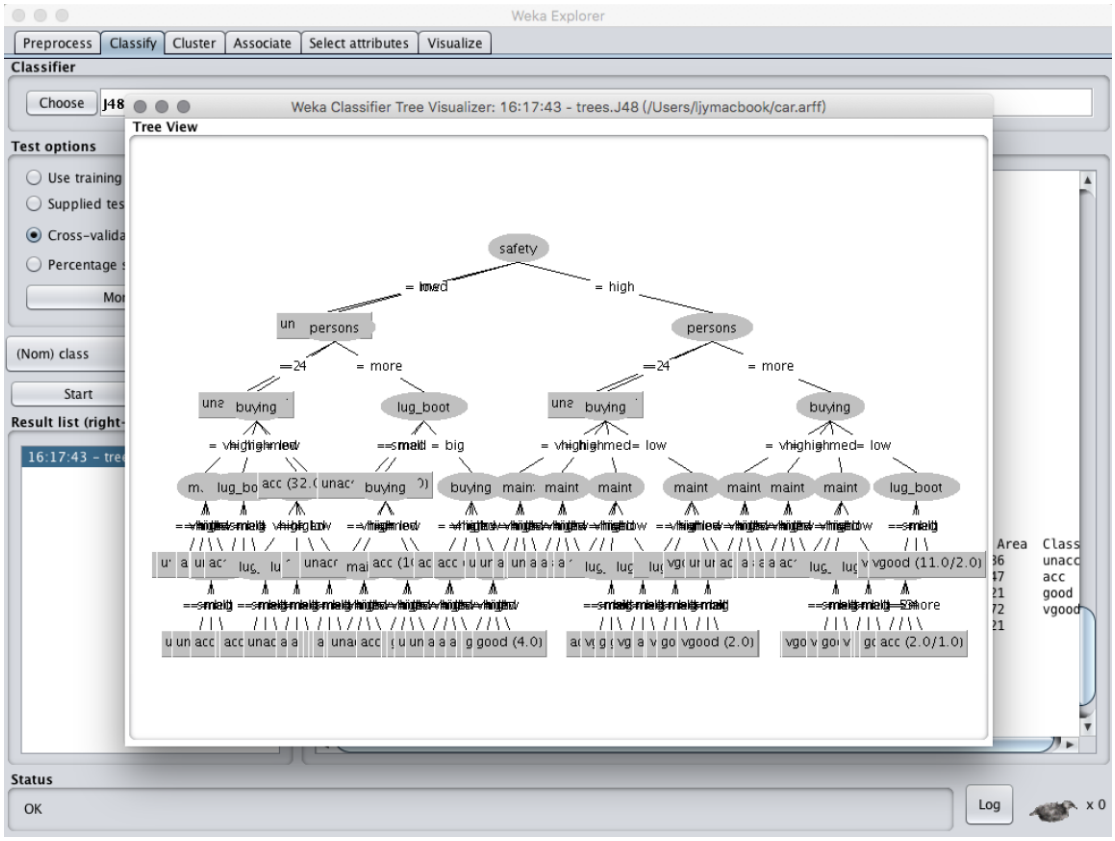

Figure 17 The Tree of J48

Save the training model

After training the classifier, the corresponding running items will also be displayed in the result list. Left-click on these items to switch between the generated results. Right-clicking an item will pop up a menu with the following options: view in the main window, view in a separate window, save the result buffer, load the model to save the model, and re-evaluate the model on the current test set. and many more. You can save the training model by clicking the "Save Model" button. Then saved training model as *car model1st.model*. At the same time, we also noticed that there is a "load model" button. With the load model, you can directly call any previously stored training model in the corresponding data set format during the next training.

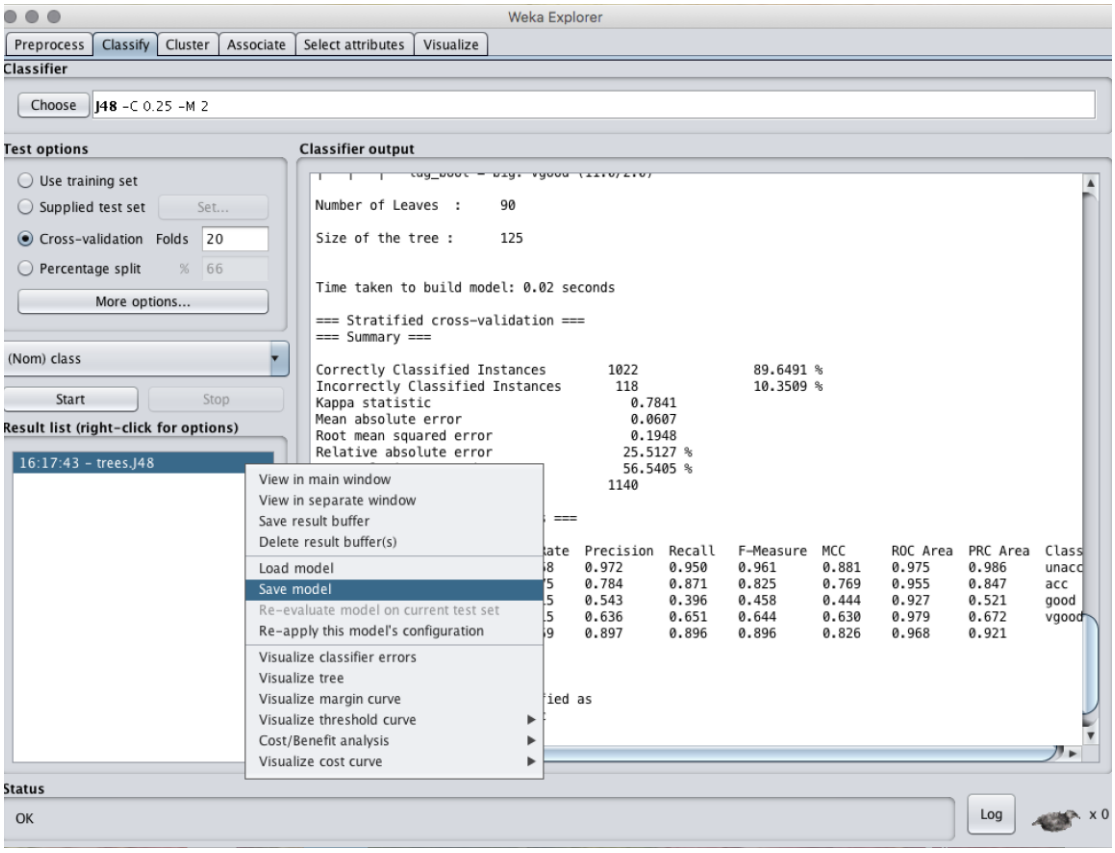

Figure 18 Save the model from J48 Algorithm in local path

3.4. Get classified files and create compressed file sets

After obtaining the training model, download it and use it in the Java environment.
Here we still use the training model data as input data, but this time we use the JAVA environment to complete the prediction and classification.
Our goal here is get two output files which *Car_incorrectly.txt* store the incorrectly predicated data set, also in *Car_correctly.txt* we store correctly predicate data set. Then we process the obtained *Car_correctly.txt*, combine the Java language with the model obtained in WEKA, and extract the decision tree path of the model through which each row of data in *Car_correctly.txt* passes. Then find the nodes through which the data passes according to these decision tree paths and store the nodes. For compressed files, it is actually to combine the data set obtained in *Car_incorrectly.tx*t with those data sets that only retain nodes passing through the decision tree path after processing in *Car_correctly.txt.*
It is undeniable that compressed file here are actually part of the principle of lossy data compression. Only the useful data is retained, and the unused or partially used data is discarded.
After we get the *Car-trainset.ARFF* dataset and the *Car model1st.model* we got in step 3.3, we can start our experiment in JAVA. Of course, The important point is that if you

want to use the WEKA API in a JAVA environment, you need to create an environment, such as configuring maven or jar packages.

Import the data set in the JAVA environment

Here we need to use the WEKA.core mentioned in the second chapter, which mainly uses Instances, attributes, etc. to obtain the data set through the local path.

```
DATA
Create BufferReader,
Create Instances,
Initialize BufferReader as breader.
Initialize Instances as ins.
If path of data in breaderis empty then
  Return ERROR
Else find data file from breader,put inside in ins;
```

Load the training set model in the JAVA environment

Mostly we did like the first step1,obtain the model which we stored through its path and WEKA.core.

```
Model
Create Classifier,
Initialize Classifier as cf.
If path of model read from WEKA.core.SerializationHelper is empty then
  Return ERROR
Else give cf as the model which read out;
```

Predictive classification of the imported data set

After obtaining the model and data, you can proceed to the classification step. All data will be classified according to the predictions in the training model. There are two situations. The first one is in line with the model prediction also store in *Car_correctly.txt*, and the second is the one that does not meet the model prediction also tore in *Car_incorrectly.txt*.In other words, if the class is good in the original data, but after the data passes the training model, the prediction is displayed as vgood, then this situation is not in line with the prediction.

```
Classification
Create Car_incorrectly.txt,Car_correctly.txt,
If    cf.classifyInstance(ins(each row))==ins(each row).classvalue
```

```
        Write in Car_correctly.txt
Else Write in Car_correctly.txt;
```

The 5th lines in *Car_incorrectly.txt* as blow:

```
predict result:1.0 predict value:acc
actual result:0.0 features:vhigh,med,2,more,small,high,unacc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:1.0 features:vhigh,med,3,more,med,med,acc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:1.0 features:vhigh,med,4,4,med,med,acc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:1.0 features:vhigh,med,4,more,med,med,acc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:1.0 features:vhigh,med,5more,4,med,med,acc
-------------------------------------------------------------
```

The 5th lines in *Car_correctly.txt* as blow:

```
safety = low: unacc (369.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,low,unacc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,med,unacc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,high,unacc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,med,low,unacc
-------------------------------------------------------------
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,med,med,unacc
-------------------------------------------------------------
```

2 Compressed Data Set

Here is the core of our lossy compression Technique, for structured data, the prediction of the training model can be correctly predicted during the classification process, indicating that the training model can be found regularly. This rule is the decision tree model, and each rule is the path of the decision tree. In order to quickly classify the same structured data (with different data content) into decision trees, it is essential to study the classified data.

In image processing, we use compression algorithms to propose feature values, in text data, we can also try feature value extraction in similarly.

Corresponding to each row of data, there is a set of eigenvalues, and these eigenvalues are the nodes where the data passes through the decision tree path.

Therefore, the first step in creating a compressed file is to find the decision tree path through which each row of data in the file passes, and then save nodes through the path and extract feature values. Finally, the extracted nodes are stored locally in the same format as the original data.

The next txt is the decision path where we got from 5th lines in *Car_correctly.txt*

```
safety = low: unacc (369.0)
------------------------------------------------------------
safety = med
|   persons = 2: unacc (112.0)
------------------------------------------------------------
safety = high
|   persons = 2: unacc (129.0)
------------------------------------------------------------
safety = low: unacc (369.0)
------------------------------------------------------------
safety = med
|   persons = 2: unacc (112.0)
------------------------------------------------------------
```

After we apply the compression algorithm, we achieve the next txt show the compressed data set in *Car_correctly.txt*, use 5th lines as examples always.

```
?,?,?,?,?,low,unacc
safety = low: unacc (369.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,low,unacc
------------------------------------------------------------
?,?,?,2,?,med,unacc
safety = med
|   persons = 2: unacc (112.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,med,unacc
------------------------------------------------------------
?,?,?,2,?,high,unacc
```

```
safety = high
|   persons = 2: unacc (129.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,small,high,unacc
------------------------------------------------------------
?,?,?,?,?,low,unacc
safety = low: unacc (369.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,med,low,unacc
------------------------------------------------------------
?,?,?,2,?,med,unacc
safety = med
|   persons = 2: unacc (112.0)
predict result:0.0 predict value:unacc
actual result:0.0 features:vhigh,vhigh,2,2,med,med,unacc
------------------------------------------------------------
```

The pseudocode to complete the compressed file 2 is as follows:

```
Create Car_correctly.txtcompress
Create array treetext
Create array tree
Create array tmptree

Create int tree_cnt, maxlayer

Set treetext=cf.toString()
Set tree= new String[treetext.length]
for i init to limit by treetext.length
    IF treetext.length=0
    Break;
    Else tree_cnt++,maxlayer
End
Copy tree into treetmp
Copy allinstances into result
If prediction = class
    Create array double dt =MembershipValues in instance
    Create array string dtv=all dt.length
    Create cnt=0
    For ly init to limit by maxlayer+1
      For m init to limit by tree_cnt
       If tem_tree[m]=ly
         dtv[cnt]=tmp_tree[m]
         cnt+=1
```

```
        For j init to limit by dt
          If dt[j]>0
            Create array string tmp1 from dtv[j-1]
              If tmp1.length>0
                Create array string tmp2 from tmp1[1]
          Int pos=-1
          For k init to limit by result
            If tmp contains instance.attribute[k].name
              pos=k;
              Break;
            Create string content=tmp1[0]
              If tmp1.length bigger than 1 also pos is bigger or equal 0
                result[pos] =tmp1[1]
          //only strore the passed node
          content =" "
          If result.length>0
            For kk init to limit by result.length
              If kk=0
                If content=result[0]
                  If content=" "
                    content=?
              Else
              If result[k]=" "
                result[k]="?"
            content = content + "," + result[k];
Output conent
```

The final compressed file named car-compress.ARFF is composed of the compressed *Car_correctly.txt* combined with the classified *Car_incorrectly.txt.*
Similarly, they are the same as the car.arff, test data set, and training data set. That means they all have 7 attributes and the ARFF format.

3.5. Training compressed data model (second model)

We use WEKA to train the compressed data set. The training data set is opened through WEKA. Also here, we use WEKA and the data set we need to use to operate. These steps are actually similar with the steps in 3.3.

Open the dataset by WEKA

Same as what we did in 3.3 .
Click Open file to open a dialog box that allows you to browse the data files on the local file system and find the saved training data set car-compress.ARFF.

Select classifier

Same as what we did in 3.3

Select algorithm

Same as what we did in 3.3

Set the Options

Same as what we did in 3.3

Set the classes attribute

Same as what we did in 3.3 .
We choose the classic algorithm C4.5, the corresponding J48 algorithm in WEKA firstly.

Start training the model

Same as what we did in 3.3

Output

After we tried different classification tree algorithms and set different options, we compared the accuracy of each model to find the best one as the final model.
For recording clarity, we create abbreviations for the two options.
For example: cross-validation with 10 folds we write as Cv10.c
Percentage split with number 10 we write as Ps10.

Table 2 different Algorithms with different Options in training second model for the CAR dataset

ID	*Algorithms*	*Options*	*Accuracy*
1	*Hoeffding tree*	*Cv10*	*70.0231 %*
2	*Hoeffding tree*	*Cv20*	*70.0231 %*
3	*Hoeffding tree*	*Ps10*	*69.3891 %*
4	*Hoeffding tree*	*Ps30*	*70.9091 %*
5	*J48*	*Cv10*	*72.4537 %*
6	*J48*	*Cv20*	*72.0486 %*
8	*J48*	*Ps10*	*69.5177 %*
9	*J48*	*Ps30*	*70.0826 %*

10	*LMT*	*Cv10*	*73.7269 %*
11	*LMT*	*Cv20*	*74.3634 %*
12	*LMT*	*Ps10*	*69.1318 %*
13	*LMT*	*Ps30*	*70.9917 %*

In this data set, we can see that when the cross-validation set is folded, LMT has the highest accuracy, which is 20. Similar to the results we trained in Chapter 3.3, after checking the decision tree generated by the algorithm, it is still the same as before, we have to abandon this model.

Finally, we choose J48 combine with cross-validation, the folds is 10.

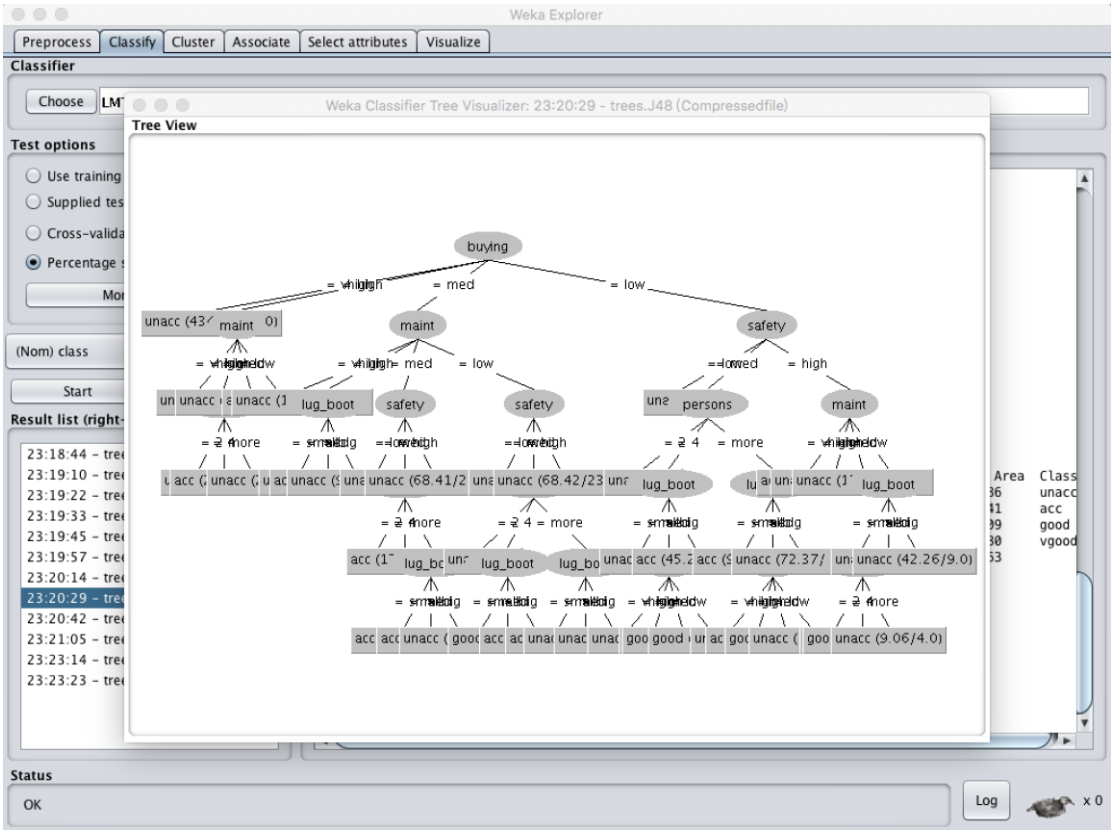

Figure 19 The Tree of J48 combine with cross-validation, the folds is 10.

Save the training model

We still save the training model by clicking the "Save Model" button. Then save the training model as *car model2nd.model*

3.6. Compare the difference between the two models

Before this chapter, we already have two training models,*car model1st.model* and *car-model2nd.model.* And the test data set for testing both of them .

What we want get are, calculate the overall accuracy, compute also precision and recall for each class. In this step, we rely on JAVA, use the java language to load the model and data, and evaluate by calling the evaluation function WEKA.classifiers.evaluation based on the WEKA API.
The simple steps implement in JAVA as follows:
1. Use the test dataset for the initial model (first model)
2. Use the test dataset to train the compressed data set model (the second model)
3. Output the accuracy of the two models and recall for each class to the table
Similarly, we can use pseudo code for deduction

```
safety = low: unacc (369.0)
Create BufferedReader1, BufferedReader2
Initialize BufferedReader1 as b1,BufferedReader as b2
Create Instances1, Instances2,
Initialize Instances1 as ins1, Instances2 as ins2
Create Classifier1, Classifier2
Initialize Classifier1as cf1, Classifier2as cf2
Copy path of training set in b1
Copy path of compressed set in b2
Copy path of car model1st.model in cf1
Copy path of car-j48-2.model in cf2
ins1 read from b1,ins2 read from b2,ins3 read from b3
if b1,b2,cf1,cf2 which one is empty
  ERROR
Else create Evaluation e1,e2
Set e1.evaluationmodel as (cf1,test data set)
Set e2.evaluationmodel as (cf2,test data set)
Print evaluation from e1,e2,e3
```

In this car data set, we can obtained the evaluation as follows:

Table 3 shown the results of evalution from two models

File name	*Overall accuracy*	*unacc*	*acc*	*good*	*vgood*
precision based on Car Model1st	*88.94736842*	*0.95962*	*0.726496*	*0.222222*	*0.695652*
precision based on Car Model2nd	*73.68421053*	*0.736842*	*#NUM!*	*#NUM!*	*#NUM!*
recall based on Car Model1st	*88.94736842*	*0.961905*	*0.787037*	*0.1*	*0.727273*
recall based on Car Model2nd	*73.68421053*	*1*	*0*	*0*	*0*

In addition, in order to improve the accuracy of subsequent analysis, we need to continuously expand the input data set. Therefore, in the uci data set, we also selected other data sets for experiments.
The steps of these data sets are the same as those described in this chapter. After two appropriate decision tree models are selected, Finally, the test data set is used for evaluation.

This table shows the other data we found in the UCI data set, their names after being converted into ARFF format files, and the number of their instances.

Table 4 All Data sest we used ,with their names and number of instance

Data set	*Data name*	*Number of instances*
Balance Scale	*Balance-Scale.arff*	*625*
Breast Cancer	*Breast-Cancer.arff*	*286*
Mushroom	*Mushroom.arff*	*8124*
Lymphography	*Lymphography.arff*	*148*
Congressional Voting Records	*Voting-Records.arff*	*435*
Balloons	*Balloons.arff*	*16*
Lenses	*Lenses.arff*	*24*
Nursery	*Nursery.arff*	*12960*
Shuttle Landing Control	*Shuttle-Control.arff*	*15*

4

Result

This chapter first explains how to evaluate the method based on overall accuracy, regression rate and prediction rate for each corresponding category. The purpose is to analyze the reasons for the changes in these variables, what methods should be used to effectively improve the accuracy of classification, and how to establish an effective data model.

The algorithm will be tested against multiple data sets, and the experimental results will be summarized and analyzed.

More specifically, we are interested in the following:

How to choose the right data model

How to properly handle missing data values

How to correctly view the data in the output of the classifier, including regression rate, classification accuracy rate, etc.

4.1. What is Accuracy,recall,Precision

Here we first introduce a few common model evaluation terms. Now suppose that our classification target has only two categories, which are counted as positive and negative respectively:

1. True positives (TP): The number of positive examples that are correctly classified, that is, the number of instances (number of samples) that are actually positive and classified as positive by the classifier;

2. False positives (FP): The number of false positives, that is, the number of instances that are actually negative but classified as positive by the classifier;

3. False negatives (FN): the number of false negatives, that is, the number of instances that are actually positive but classified as negative by the classifier;

4. True negatives (TN): The number of negative examples that are correctly classified, that is, the number of cases that are actually negative and are classified as negative by the classifier.

Table 5 The table shown the confusion matrix of these four terms.

	Preditecd			
		Yes	***No***	***Total***
Actual	***Yes***	***TP***	***FN***	***P（Actually is Yes）***
	No	***FP***	***TN***	***N(Actually is No）***
	Total	***P'（divided into Yes）***	***N'（ divided into No）***	***P+N***

Note that P=TP+FN represents the number of samples that are actually positive examples. Remember that True and False describe whether the classifier is judged correctly. Positive and Negative are the classification of the classifier result.
If a positive example is counted as 1, and a negative example is counted as -1, that is, positive=1, negative=-1, 1 means True, -1 means False, then the actual class label=TF*PN, TF is true or false, PN is positive or negative. For example, the actual category label of True positives (TP)=1*1=1 is a positive example, the actual category label of False positives (FP)=(-1)*1=-1 is a negative example, and the actual category label of False negatives (FN) The category label=(-1)*(-1)=1 is a positive example. The actual category label of True negatives(TN)=1*(-1)=-1 is a negative example.Accuracy: The accuracy rate is our most common evaluation index, accuracy = (TP+TN)/(P+N). This is easy to understand, that is, the number of samples to be matched divided by the number of all samples. Generally speaking, the more accurate High, the better the classifier.

Error rate:
The error rate is the opposite of the correct rate. It describes the proportion of misclassification by the classifier, error rate = (FP+FN)/(P+N). For a certain instance, right and wrong are mutually exclusive events, so accuracy =1-error rate;

Sensitive:
Sensitive = TP/P, which represents the proportion of all positive examples that are matched, and measures the classifier's ability to recognize positive examples;

Specificity:
Specificity = TN/N, which represents the proportion of all negative examples that are matched, and measures the classifier's ability to recognize negative examples;

Precision:
Accuracy is a measure of accuracy, representing the proportion of positive examples (actually positive examples) classified as positive.
precision=TP/(TP+FP);

Recall:
Recall rate is a measure of coverage. There are multiple positive examples of the measure, which are classified as positive.
Recall=TP/(TP+FN)=TP/P=sensitive. It can be seen that the recall rate and sensitivity are the same.

Other evaluation indicators
Calculation speed: the time required for classifier training and prediction;
Robustness: the ability to deal with missing values and outliers;
Scalability: the ability to handle large data sets;

Interpretability: The comprehensibility of the prediction criteria of the classifier, like the rules generated by the decision tree, is easy to understand, and a bunch of parameters of the neural network are not easy to understand, so we have to regard it as a black box.

4.2. Model comparison of each data set

Before displaying the results of each set of data, we must know what each parameter in these structured data represents and the meaning of the classification. In other words, we need to know what the attributes in the data are, and what the classified class is.
Then, we will adopt a plurality of models in the analysis result table and the result is stored a plurality of data sets.

4.2.1 balance-scale

Table 6 he data set which names Balance scale was store as the arff suffix file Balance-Scale.arff.

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Balance Scale	*Balance-Scale.arff*	*5*	*3*	*625*

The class names as L,B,R.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as Balance-Scale Model1st.model, the second model which names as Balance-Scale Model2nd.model represents the model based on compressed data set.

Table 7 the table records recall, precision of class L,B,R and the overall accuracy in the two models

File name	*Overall accuracy*	*L*	*B*	*R*
precision based on Balance-Scale Model1st	*79.61165049*	*0.828571*	*0*	*0.875*
precision based on Balance-Scale Model2nd	*33.98058252*	*0.630435*	*0.036697*	*0.725490196*
recall based on Balance-Scale Model1st	*79.61165049*	*0.84466*	*0*	*0.836956522*
recall based on Balance-Scale Model2nd	*33.98058252*	*0.281553*	*0.363636*	*0.402173913*

Overall accuracy:

We can clearly see that in the *Balance-Scale Model1st.model*, the accuracy rate is higher than that in the *Balance-Scale Model2nd.model*.
Evidently see the accuracy of the *Balance-Scale Model1st.model*, is 79.61%, while in the *Balance-Scale Model2nd.model*,is only 33.98 percent. In terms of the accuracy of the model, it seems that the first one is much better.

Recall for each class

Corresponding to the recall rate of class L, the *Balance-Scale Model2nd.model,* still shows a downward trend, 0.28 is far less than 0.84. For B, recall has increased from 0 to 0.36. Let's look at R again, which is still the same as L Shows a reducing trend.

Precision for each class:

Compared with the trend of precision rate, it is consistent with recall rate on the whole, because the calculation methods of precision rate and precision rate are similar. We will discuss and analyze this in the next chapter. Similarly, the accuracy of L in the *Balance-Scale Model2nd.model* l is a downward trend, while B is an upward trend, but it only rises from 0 to 0.036. In the *Balance-Scale Model2nd.model*, R dropped from 0.857 to 0.725.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters：

Table 8 this table shown the number of incorrectly and correctly instances, the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Balance-Scale Model1st.model,	*164*	*42*	*24*
Balance-Scale Model2nd.model	*70*	*136*	*7*

The number of correct instances and incorrect instances can be tested from another angle to get the overall accuracy rate. For different paths, the total number of paths also shows a downward trend. As the accuracy decreases, the number of different decision tree paths is also greatly reduced. It shows that the branches of the decision tree are also reduced.

4.2.2 breast-cancer

Table 9 the data set which names Breast Cancer was store as the arff suffix file Breast-Cancer.arff.

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Breast Cancer	*Breast Cancer.arff*	*10*	*2*	*286*

The two classes are no-recurrence-events, recurrence-events

Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Breast Cancer Model1st.model,* the second model which names as *Breast Cancer Model2nd.model* represents the model based on compressed data set.

Table 10 the table records recall, precision of class no-recurrence-events， recurrence-events and the overall accuracy in the two models

File name	*Overall accuracy*	*no-recurrence-events*	*recurrence-events*
precision based on Breast Cancer Model1st	*75.53191489*	*0.764045*	*0.6*
precision based on Breast Cancer Model2nd	*27.65957447*	*0.583333*	*0.231707317*
recall based on Breast Cancer Model1st	*75.53191489*	*0.971429*	*0.125*
recall based on Breast Cancer Model2nd	*27.65957447*	*0.1*	*0.791666667*

Overall accuracy:

We can clearly see that in the *Breast Cancer Model1st.model*, the accuracy is much higher than the s *Breast Cancer Model2nd.model*. Obviously, the accuracy of the *Breast Cancer Model1st.model* is 75.53%, but another model is only 26.65%.

Recall for each class

Corresponding to the recall rate of non-repetitive events, the *Breast Cancer Model2nd.model* still shows a downward trend, where 0.1 is much smaller than 0.971. For the second repetitive event, the recall rate in the *Breast Cancer Model2nd.model* is 0.79. The recall rate of the first one is only 0.125

Precision for each class:

Corresponding to the precision of no-recurrence-events, the *Breast Cancer Model2nd.model* still shows a downward trend, 0.58 is less than 0.76. For the second class recurrence-events, the precision in the *Breast Cancer Model2nd.model* is 0.23, But in the first one it was 0.6. Both classes are due to a downward trend

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters:

Table 11 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Breast Cancer Model1st	*71*	*23*	*6*
Breast Cancer Model2nd	*26*	*68*	*5*

The number of correct instances and incorrect instances can be tested from another angle to obtain overall accuracy. For different paths, the total number of paths also declined. Here it changes from 6 to 5.

4.2.3 Car

Table 12 the data set which names Car was store as the arff suffix file Car.arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Car	*Car.arff*	*7*	*4*	*15*

The four classes are unacc,acc,good,vgood.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Car Model1st.model,* the second model which names as *Car Model2nd.model* represents the model based on compressed data set.

Table 13 the table records recall, precision of class unacc,acc,good and vgood in the two models

File name	*Overall accuracy*	*unacc*	*acc*	*good*	*vgood*
precision based on Car Model1st	*88.94737*	*0.95962*	*0.726496*	*0.222222*	*0.695652*
precision based on Car Model2nd	*72.80702*	*0.795652*	*0.425743*	*0.666667*	*#NUM!*
recall based on Car Model1st	*88.94737*	*0.961905*	*0.787037*	*0.1*	*0.727273*
recall based on Car Model2nd	*72.80702*	*0.871429*	*0.398148*	*0.3*	*0*

Overall accuracy:

In the above table, we can see that first one model which refer to *Car Model1st.model* is still better than *Car Model2nd.model* in general because the accuracy of *Car Model2nd.model* is reduced from 88.94 to 72.80. But the overall data is no different from the previous data set.

Recall for each class

Throughout the recall table, unacc and acc are showing a downward trend, good upward trend, but never 0.727 vgood dropped to 0, which is the mean forecast vgood data in

the raw data, through the decision-making *Car Model2nd.model* analysis after, and not return

Precision for each class:

For the class unacc, precision has dropped from 0.959 to 0.795, and the same situation is true for the acc class. Then the precision of class good is actually on the rise, and the ratio rises to 0.66 in *Car Model1st.model*. Unfortunately for class vgood, in *Car Model2nd.model* is no output precision, which is not predicted.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters：

Table 14 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Car Model1st	*507*	*63*	*65*
Car Model2nd	*415*	*155*	*34*

The number of correct instances and incorrect instances can be tested from another angle to obtain overall accuracy. For different paths, the total number of paths also shows a downward trend. Obviously, in the *Car Model2nd.model*, the number of non-repeating path tree has been reduced.

4.2.4 Lenses

Table 15 the data set which names Lenses was store as the arff suffix file Lenses.arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Lenses	*Lenses.arff*	*5*	*3*	*24*

The three classes are soft,hard,none
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Lenses Model1st.model,* the second model which names as *Lenses Model2nd.model* represents the model based on compressed data set.

Table 16 the table records recall, precision of class soft,hard,none in the two models

File name	Overall accuracy	soft	hard	none
precision based on Lenses Model1st	75	0.5	0.666667	1
precision based on Lenses Model2nd	62.5	#NUM!	#NUM!	0.625
recall based on Lenses Model1st	75	1	1	0.6
recall based on Lenses Model2nd	62.5	0	0	1

Overall accuracy:

In the above table, we can see that *Lenses Model1st.model* is still better than *Lenses Model2ndt.model* overall, because the accuracy rate in second one is reduced from 75 to 62.5.

Recall for each class

In the entire recall table, the recall of soft and hard is 0, which means that in the test set, all data predicted as soft and hard are not predicted. For the class none, the recall increased from 0.6 to 1.

Precision for each class:

Throughout the precision table, the soft and hard precision are empty, indicating that the test set, all data for the prediction of soft and hard as in the recall, or not used to predict. Similarly, the precision of the none class is not the same as in the class, but has dropped.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters：

Table 17 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	Correctly number of instances	Incorrectly number of instances	Different Path
Lenses Model1st	6	2	3
Lenses Model2nd	5	3	2

In this table, not difficult to find data in each row are very similar, because our data sample is limited, it can be said that our data is too small, although the size of digital or less, but to do the calculation, then the whole There is still a difference. In general, the same as some previous data, paths are also in a downward trend.

4.2.5 Lymphography

Table 18 the data set which names Lymphography was store as the arff suffix file Lymphography.arff

Data set	*Data name*	*Number of attributes*	*Number of classed*	*Number of instances*
Lymphography	*Lymphography.arff*	*18*	*4*	*148*

The four classes are normal find, metastases, malign lymph, fibrosis.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Lymphography Model1st.model,* the second model which names as *Lymphography Model2nd.model* represents the model based on compressed data set.

Table 19 the table records recall, precision of class normal,metastases,malign_lymph,filrosis in the two models

File name	*Overall accuracy*	*normal*	*metastases*	*malign_lymph*	*fibrosis*
precision based on Lymphography Model1st	*69.38776*	*#NUM!*	*0.758621*	*0.6*	*#NUM!*
precision based on Lymphography Model2nd	*61.22449*	*#NUM!*	*0.612245*	*#NUM!*	*#NUM!*
recall based on Lymphography Model1st	*69.38776*	*0*	*0.733333*	*0.705882*	*0*
recall based on Lymphography Model2nd	*61.22449*	*0*	*1*	*0*	*0*

Overall accuracy:

In this set of data, the two models do not seem to be very different, the only difference is only reduced from 69.38 to 61.22 given by the *Lymphography Model2nd.model* intelligence.

Recall for each class

In the two models of normal and fibrosis, the recall rate was 0, the transfer rate increased from 0.73 to 1, and malign_lymph decreased from 0.70 to 0.

Precision for each class:x

In both the normal and fibrotic models, the exact null transfer decreased from 0.75 to 0.61, while malign_lymph changed from 0.76 to null.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters,

Table 20 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Lymphography Model1st	*34*	*15*	*5*
Lymphography Model2nd	*30*	*19*	*3*

And some of the data overall still the same as before, paths also a downward trend.

4.2.6 Mushroom

Table 21 form4.16 the data set which names Mushroom was store as the arff suffix file mushroom.arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Mushroom	*Mushroom.arff*	*22*	2	*8124*

The two classes names are edible and poisonous, we name them as e and p for simple. Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Mushroom Model1st.model,* the second model which names as *Mushroom Model2nd.model* which refer to the model based on compressed file.

Table 22 the table records recall, precision of class e,p in the two models

File name	*Overall accuracy*	*e*	*p*
precision based on Mushroom Model1st	*100*	*1*	*1*
precision based on Mushroom Model2nd	*52.07012*	*0.520701*	*#NUM!*
recall based on Mushroom Model1st	*100*	*1*	*1*
recall based on Mushroom Model2nd	*52.07012*	*1*	*0*

Overall accuracy:

Through the *Mushroom Model2nd.model*, the accuracy is significantly reduced, from the original 100 to 52.07.

Recall for each class

The recall of Class e has not changed, it is still 1. For p, it has changed from 0.

Precision for each class:

The precision of class e is reduced from 1 in the *Mushroom Model1st.model* to 0.52, followed by class p, from 1 to null.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters:

Table 23 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Mushroom Model1st	*2681*	*0*	*19*
Mushroom Model2nd	*1396*	*1285*	*17*

In general, the same as some previous data, paths are also in a downward trend.

4.2.7 Nursery

Table 24 the data set which names Nursery was store as the arff suffix file Nursery .arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Nursery	*Nursery.arff*	*9*	*5*	*12960*

The five classes names are not_recom ,recommend,very_recom,priority ,spec_prior.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as Nursery *Model1st.model,* the second model names *Nursery Model2nd.model* which refer to the model based on compressed file.

Table 25 the table records recall, precision of not_recom,recommend,very_recom_priority,spec_prior inf the two models

File name	*Overall accuracy*	*not_recom*	*recommend*	*very_recom*	*priority*	*spec_prior*
precision based on Nursery Model1st	*96.16553659*	*1*	*#NUM!*	*0.719512*	*0.941095*	*0.958793*
precision based on Nursery Model2nd	*45.2887538*	*0.046243*	*#NUM!*	*#NUM!*	*0.463602*	*0.478632*

recall based on Nursery Model1st	*96.16553659*	*1*	*0*	*0.551402*	*0.945682*	*0.972388*
recall based on Nursery Model2nd	*45.2887538*	*0.005743*	*0*	*0*	*0.758357*	*0.626866*

Overall accuracy:

In this set of data, overall accuracy has dropped significantly, from 96.16 to 45.28

Recall for each class

In these classes, all trends are decreasing, except for recommend which is always 0

Precision for each class

Similarly, the trend of precision is also decreasing. Recommend stayed empty value, very_recom from 0.719 become empty

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters:

Table 26 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Nursery Model1st	*4113*	*164*	*239*
Nursery Model2nd	*1937*	*2340*	*78*

In general, the same as some previous data, paths are also in a downward trend

4.2.8 Congressional Voting Records

Table 27the data set which names Congressional Voting Records was store as the arff suffix file voting .arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Congressional Voting Records	*Voting.arff*	*17*	*2*	*435*

The two classes names are democrat and republican

Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Voting Model1st.model,* the second model which names as *Voting Model2nd.model* which refer to the model based on compressed file.

Table 28 the table records recall, precision of democrat ,republication the two models in two models

File name	*Overall accuracy*	*democrat*	*republican*
precision based on Voting Model1st	*97.22222*	*0.975309*	*0.968254*
precision based on VotingModel2nd	*56.25*	*0.5625*	*#NUM!*
recall based on Voting Model1st	*97.22222*	*0.975309*	*0.968254*
recall based onVoting Model2nd	*56.25*	*1*	*0*

Overall accuracy:

In this set of data, overall accuracy decreased significantly reduced from the original 97.22 to 56.25

Recall for each class

In these classes, democrat has changed from 0.97 to 1, and republication has dropped from 0.968 to 0

Precision for each class

Similarly, the trend of precision is also decreasing.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters:

Table 29 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Voting Model1st	*4113*	*164*	*239*
Voting Model2nd	*1937*	*2340*	*78*

The overall data is still the same as some of the previous data, but when the accuracy rate is reduced, it is not difficult to find that the data of different paths is also much less than before.

4.2.9 Hayes-Roth

Table 30 the data set which names hayes-Roths was store as the arff suffix file HRoth .arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Hayes-Roth	*HRoth.arff*	*6*	*3*	*160*

There are three classes which are nominal value between 1 and 3.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Hayes Model1st.model,* the second model which names as *Hayes Model2nd.model* which refer to the model based on compressed file.

Table 31 the table records recall, precision of 1,2,3the two models in two models

File name	*Overall accuracy*	*1*	*2*	*3*
precision based on Hayes Model1st	*88.67925*	*0.888889*	*0.851852*	*1*
precision based on Hayes Model2nd	*49.0566*	*0.333333*	*0.536585*	*#NUM!*
recall based on Hayes Model1st	*88.67925*	*0.8*	*0.92*	*1*
recall based on Hayes Model2nd	*49.0566*	*0.2*	*0.88*	*0*

Overall accuracy:

In this set of data, overall accuracy decreased significantly reduced from the original 88.67 to 49.05

Recall for each class

In this set of data, tagged as class 1 2 3, with respect to their recall a model for both decrease.

Precision for each class

The same recall, in this set of data, all class of precision is a downward trend.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters:

Table 32 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Hayes Model1st	*93*	*13*	*11*
Hayes Model2nd	*36*	*70*	*5*

Generally, in the case where accuracy is lowered, the number of paths is reduced accordingly.

4.2.10 Balloons

Table 33 the data set which names Balloons was store as the arff suffix file Balloons.arff

Data set	*Data name*	*Number of attributes*	*Number of classes*	*Number of instances*
Balloons	*Balloons.arff*	*5*	*2*	*16*

There are two classes which are True and False, we refer as T nad F.
Classification is done by decision tree, In order to see the recall rate and prediction rate of each class more intuitively, as well as the overall accuracy of the both model, we fill in the data in the following table.
The first model represents the model based on training data set, we also name it as *Balloons Model1st.model,* the second model which names as *Balloons Model2nd.model* which refer to the model based on compressed file.

Table 34 the table records recall, precision of true，false in two models

File name	*Overall accuracy*	*1*	*2*
precision based on Balloons Model1st	*100*	*1*	*1*
precision based on Balloons Model2nd	*57.14286*	*#NUM!*	*0.571429*
recall based on Balloons Model1st	*100*	*1*	*1*
recall based on Balloons Model2nd	*57.14286*	*0*	*1*

Overall accuracy:

In this set of data, overall accuracy decreased significantly reduced from the original 100 to 57.14

Recall for each class

In this set of data, the label of class should be true and false. For statistical convenience, in the experiment, set true to label 1, and fasle to label 2. Because it is a binary selection, when the recall of label 1 drops to 0, label 2 is 1.

Precision for each class

The precision with label 1 changed from 1 to empty, and the same on label 2, changed from 1 to 0.57.

Different paths

In order to better analyze the accuracy and various parameters later, we output the following parameters,

Table 35 this table shown the number of incorrectly and correctly instances, and the totally distinct paths in two models

Model	*Correctly number of instances*	*Incorrectly number of instances*	*Different Path*
Balloons Model1st	*93*	*13*	*11*
Balloons Model2nd	*36*	*70*	*5*

As with the previous experimental data, as the accuracy rate decreases, the decision tree path for classification also decreases.

4.2 Summary of results

From the results generated by our 10 sets of data, we can accurately see that most of the accuracy is reduced. Accuracy of many training models based on compressed data sets is only about half of that of training models based on original data sets, and for each class, Precision and recall are also decreasing. All training models based on compressed data sets have overall performance slightly lower than the initial training models.

In order to find out more, we also output the total number of distinct paths in different models. In these 10 sets of data, the result is greater than the number of paths of the compressed data set to train the model based on the number of paths of the original data set to train the model results. Although some of these differences are not significant, for example, Mushrooms data set and date set of lenses may be missing one or two paths in the training model of compressed data.

Even so, the parameters we refer to have dropped a lot. Therefore, the sum of different paths and accuracy have mutual influence. At present, the trend is proportional. When Accuracy drops, the number of paths also drops.
For precision ,recall and others , we will discuss and summarize in the next chapter.

5
Discussion

We can intuitively see from the data in the above chapter that the overall trend is showing a decline. In order to be able to analyze more accurately, we first need to understand how Accuracy, precision, and recall are calculated. Although obtaining these data is very simple, but the algorithm should also be explained in detail.

Simultaneously, we will also analyze compression techniques applied to compressed data sets.

5.1 Reasons for the decline in classification accuracy

In the following part, we will analyze from multiple aspects.The most important thing is how we got these result parameters, whether there are limitations in the data set, and what are the problems in the compression algorithm.

5.1.1 Analysis from Precision, recall, and Precision

In summary, we can clearly see that (TP+TN)/(P+N) is the formula for calculating Accuracy, the decline of Accuracy, in our experiment, we can most intuitively see that when P+N does not change The decline in the number of TP and TN determines the decline in Accuracy. To be sure, we have two sets of data used to generate the model, the number of samples is the same, so here TP + TN engagement dropped, that is a lot of data are classified are wrong.

Regarding Precision, the formula is TP / (TP + FP), due to the misclassification, leading FP rise, TP fall so that Precision will increase as the classification error rate decreases.

For Recall, recall=TP/(TP+FN)=TP/P. While the total amount of P remains the same, the number of TPs has dropped because many are misclassified. Therefore, the downward trend of Recall is also inevitable.

It can be seen that the numerator of precision and recall is the same, the denominator of precision is (FP + TP), the denominator of recall is (FN + TP), and both denominators contain TP, so precision and recall The relationship between depends on FP and FN, and FP and FN are negatively correlated, so precision and recall are also negatively correlated, that is, when the precision increases, the recall rate will increase; when the precision decreases, the recall rate increases.

5.1.2 Analysis from data set

The attributes we use are all datasets with categorical attributes and categorial attributes, which means that they have a certain structure and are also structured data. Because each row of data follows the same format and select value from range.
In the experiment, the experience of poor model effect occurs on the compressed data set. From the compressed data, we use question marks to replace the nodes that do not stay in the decision tree path, and only keep the nodes that the decision tree path passes through (specific data we can refer to the results of the third chapter car as a case of data derived).

In fact, the "?" used when replacing a node actually represents a missing value, which also makes our data dirty. The value that should be in the structured data is missing, which naturally makes the data set tainted, and the accuracy of classification is also reduced.

Another reason is that there are too few instances in our data set. Especially in data set Balloons and Lenses, both of theirs' instances are under 50. It is undeniable that when the number of instances is too small, a small classification error, even one wrong classification, will cause the accuracy to drop a lot. Scilicet, when the denominator is constant, the size of the numerator determines the overall size.

5.1.3 Analysis from compression Technique

As mentioned in 5.1.3, our compression technology happened to destroy our data. Many data have missing values during the replacement process.
We only care about the routines that the decision tree path passes through, which is the feature value that best reflects the classification, but the routines that we do not use are actually a manifestation of lossy compression technology. However, it seems that the performance of lossy compression on text data is not as good as on pictures.

5.2 how we can improve the results

According to some of the problems we have raised, some solutions that can be found are summarized. Of course, if we raise the accuracy of the classification, we have done the analysis in addition to other experimental results.

5.2.1 Through the missing value

Before dealing with missing data, it is necessary to understand the mechanism and form of missing data. The variables in the data set without missing values are called complete variables, and the variables in the data set with missing values are called incomplete variables. From the distribution of the missing, the missing can be divided into completely random missing, random missing and completely non-random missing.

missing completely at random (MCAR): It means that the missing data is completely random, does not depend on any incomplete or complete variables, and does not affect the unbiasedness of the sample. For example, the home address is missing.

missing at random (MAR): refers to that the missing data is not completely random, that is, the missing data of this type depends on other complete variables. For example, the lack of financial data is related to the size of the enterprise.
missing not at random (MNAR): refers to the lack of data related to the value of the incomplete variable itself. Such as high-income groups do not intend to provide family income.

In our experiment, although the java code is used to complete the file compression, it is practical and the three missing values mentioned above. It seems that MNAR is more in line with our data situation.There are three main types of processing methods for missing data: delete tuples, complete data, and do not process.
Deleting tuples: that is, deleting objects (tuples, records) that have missing information attribute values, so as to obtain a complete information table.The advantage is that it is simple and easy to implement, and it is very effective when the object has multiple attribute missing values, and the deleted object with missing values is very small compared to the initial data set. However, when the proportion of missing data is large, especially when the missing data is not randomly distributed, this method may cause the data to deviate and lead to wrong conclusions.

Data completion: Fill the empty value with a certain value, so as to complete the information table. Usually based on statistical principles, a missing value is filled according to the distribution of the values of the remaining objects in the initial data set. In data mining, there are several ways of complementing:

1. Filling manually
According to the actual situation to decide

2.Treating Missing Attribute values as Special values
Treat the null value as a special attribute value, which is different from any other attribute value. For example, all empty values are filled with "unknown". Generally used as a temporary filling or intermediate process.

3. Mean/Mode Completer
The attributes in the initial data set are divided into numeric attributes and non-numeric attributes to be processed separately.
If the null value is numeric, fill in the missing attribute value based on the average value of the attribute in all other objects; if the null value is non-numeric, use the mode principle in statistics The value of the attribute with the most value in all other objects (that is, the value with the highest occurrence frequency) fills in the missing attribute value.
Another method similar to it is called Conditional Mean Completer (Conditional Mean Completer). In this method, the value used for averaging is not taken from all objects in the data set, but from objects that have the same decision attribute value as the object. The basic starting points of these two data complementation methods are the same. The missing attribute values are filled with the highest possible value, but the specific methods are a little different. Compared with other methods, it uses most of the existing data to infer missing values.

4. Hot deck imputation
For an object containing a null value, the hot card filling method finds the most similar object in the complete data, and then fills it with the value of this similar object. Different problems may use different criteria to judge similarity. The method is conceptually very simple, and uses the relationship between the data to estimate the null value. The disadvantage of this method is that it is difficult to define similar standards and there are many subjective factors.

5. K-means clustering
First, determine the K samples closest to the sample with missing data according to Euclidean distance or correlation analysis, and weight the K values to estimate the missing data of the sample.

6. Assigning All Possible values of the Attribute
Fill in with all possible attribute values f the missing attribute values, which can get a better filling effect. However, when the amount of data is large or there are many missing attribute values, the calculation cost is high, and there are many possible test schemes. This method can be used if there is no variable that can be used or the effect of the reference variable is very low, which is convenient and simple.

7. Regression
Based on the complete data set, a regression equation is established. For objects that contain null values, substitute the known attribute values into the equation to estimate the unknown attribute values, and fill in with the estimated values. When the variables are not linearly related, it will lead to biased estimates. Linear regression is commonly used.

8. Expectation maximization method (Expectation maximization, EM)
The EM algorithm is an iterative algorithm that calculates maximum likelihood estimation or posterior distribution in the case of incomplete data. Two steps are performed alternately during each iteration loop: E step (Excepctaion step, expected step), given the complete data and the parameter estimates obtained in the previous iteration, the log-likelihood function corresponding to the complete data is calculated The conditional expectation of; M step (Maximzation step, maximization step), use the maximum log-likelihood function to determine the value of the parameter, and use it for the next iteration. The algorithm continues to iterate between E step and M step until it converges, that is, it ends when the parameter change between two iterations is less than a predetermined threshold. This method may fall into local extremes, the convergence speed is not very fast, and the calculation is very complicated.

9. Multiple Imputation (MI)
The multiple filling method is divided into three steps:
A set of possible filling values are generated for each null value. These values reflect the uncertainty of the unresponsive model; each value is used to fill in the missing values in the data set to generate several complete data sets.
Each filled data set is statistically analyzed using statistical methods for the complete data set.
The results from each filled data set are synthesized to produce the final statistical inference, which takes into account the uncertainty due to data filling. This method treats the vacancy value as a random sample, so the calculated statistical inference may be affected by the uncertainty of the vacancy value. The calculation of this method is also very complicated.

10. C4.5 method
Fill in missing values by looking for relationships between attributes. It looks for the two attributes with the greatest correlation between them. The one with no missing value is called the proxy attribute and the other is called the original attribute. The proxy attribute determines the missing value in the original attribute. This rule-based induction method can only deal with noun attributes with a small base.

In terms of several statistical-based methods, regression is a better method, but it is still inferior to hot deck and EM; EM lacks the uncertain components contained in MI. It is worth noting that these methods directly deal with the estimation of model parameters rather than the prediction of vacancies. They are suitable for dealing with the problem

of unsupervised learning, but for supervised learning, the situation is different. For example, you can delete objects that contain null values and use a complete data set for training, but you cannot ignore objects that contain null values when predicting. In addition, C4.5 and the use of all possible value filling methods also have better filling effects. Manual filling and special value filling are generally not recommended. [19]

No processing: does not deal with missing values, and methods of data mining directly on data containing null values include Bayesian networks and artificial neural networks. The supplementary process only supplements the unknown value with our subjective estimated value, which may not completely conform to the objective facts. While supplementing incomplete information, we changed the original information system more or less. Moreover, incorrectly filling in empty values usually introduces new noise in the data, leading to erroneous results in mining tasks. Therefore, in many cases, we still want to deal with information systems while keeping the original information unchanged.

Bayesian networks provide a natural way to express causal information between variables to discover potential relationships between data. In this network, nodes are used to represent variables, and directed edges are used to represent dependencies between variables. Bayesian networks are only suitable for situations where there is a certain understanding of domain knowledge, at least when the dependencies between variables are clear. Otherwise, learning the structure of the Bayesian network directly from the data will not only have higher complexity (exponentially increase with the increase of variables), the network maintenance cost is high, and there are many estimated parameters, which brings a lot to the system. Big difference. Affect its prediction accuracy.

Neural network can effectively deal with missing values, but the research of neural network in this field needs further development.

5.2.1 Through the data set

Try to find a data set with more Instances for analysis. When the instance is insufficient, not only does the classification accuracy rate tend to be low, but it is also prone to overfitting.
Overfitting refers to the phenomenon of matching a specific data set too closely or accurately, so that it cannot fit other data well or predict future observation results. In layman's terms, the accuracy of the trained model on the training set is very high, but the accuracy on the test set is very poor.
In our cases, to prevent overfitting, when classify small sample data ,carefully consider using cross-validation

5.2.2 Through compression Technique

The need to improve classification accuracy after compression is also related to compression technology. Generally, data compression actually uses shorter data to represent repeated data to achieve compression. Therefore, the higher the data repetition rate or the stronger the predictability, the higher the compressibility. Different data can be compressed to different degrees.

For structured data, although we can focus on lossy compression, the accuracy of failures caused by compressed data is reduced. Therefore, before using lossy compression technology, we should fully examine whether our data conforms to what we mentioned earlier For example, for image compression, Principal components analysis is a good method. At the same time, the combination of different compression algorithms is also an inevitable method for the classification algorithm.

5.3 Other suggestions

For structured data, lossy compression is actually not a particularly good idea. Even if it keeps some basic features, it has an impact on the classification accuracy. However, if lossy compression combined with deep learning, there must be more room for development.

Deep learning dating effectively solves the problems of traditional methods. Compared with traditional compression technology classification, deep learning-based methods have the following natural advantages:

Since the parameters of deep learning are pushed in based on a large amount of actual data, and traditional data compression coding methods are mainly generated manually based on prior knowledge, the excellent content adaptability of deep learning is based on signal processing models.

The deep learning method effectively uses the absorbed receptive field (receiving field), so it can not only use adjacent information but also use distant samples to improve coding efficiency, while traditional coding tools only use adjacent samples, which is difficult Use remote samples.

The compression method based on deep learning is flexible. It can further reduce the bit rate according to specific domain characteristics while achieving fast processing. The internal reFpresentation of deep learning is suitable for modern data processing.
Compared with traditional neural network compression technical indicators, the advantages of the deep learning-based method are:
(1) Based on the multi-layer network structure, the deep learning model has better nonlinear mapping capabilities, so it can help to learn data (especially images)

(2) The deep learning model improves the training speed through the process of multi-layer learning and fine-tuning of weights, to meet the requirements of large-scale data compression.
(3) A deeper learning level can more effectively remove redundant data features, thereby obtaining a higher compression ratio.
Finally, random neural networks, convolutional neural networks, recurrent neural networks, Generative Adversarial Networks, and variational automatic Encoders , etc. have been successively applied to data compression.

6
REFERENCES

[1] L. Douglas, “ "The Importance of 'Big Data': A Definition",” Gartner, 21 6 2012. .

[2] M. v. Rijmenam, "Why the 3V's Are Not Sufficient to Describe Big Data".

[3] M. M. B. Thakur, Data Mining for Big Data: A Review," International Journal of Advanced Research in Computer Science and Software Engineering, 2014.

[4] C.-Y. Z. C.L. P. Chen, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,, Information Sciences,, 2014.

[5] S.-W. M. Y.-H. L. M. Chen, Big data: A survey, Mobile Netw Appl, 2014.

[6] "the core of the big data,"Available: http://www.jyt-bj.com/wap/jishu/5505.html.

[7] J. Brownlee, "4 Types of Classification Tasks in Machine Learning," 8 4 2020 Available: https://machinelearningmastery.com/types-of-classification-in-machine-learning/.

[8] J. d. l. C. M. Argelia Berenice Urbina Nájera, "Brief Review of Educational Applications Using Data Mining and Machine Learning," redie, 24 6 2016.

[9] "Artificial neural network,"Available: https://en.wikipedia.org/wiki/Artificial_neural_network.

[10] D. H. Chowdary, "Decision Trees Explained With a Practical Example," Available: https://medium.com/towards-artificial-intelligence/decision-trees-explained-with-a-practical-example-fe47872d3b53.

[11] "C4.5 algorithm,"Available: https://en.wikipedia.org/wiki/C4.5_algorithm.

[12] "Decision tree learning," Available: https://en.wikipedia.org/wiki/Decision_tree_learning.

[13] A. V. Solank, "Data Mining Techniques Using WEKA classification for Sickle Cell Disease".

[14] J. Brownlee, "How to Run Your First Classifier in Weka," 17 2 2014. Available: https://machinelearningmastery.com/how-to-run-your-first-classifier-in-weka/.

[15] "A Machine Learning Workbench for Data Mining," Available: https://core.ac.uk/download/pdf/29195529.pdf.

[16] "Attribute-Relation File Format (ARFF)," 1 11 2008. Available: https://www.cs.waikato.ac.nz/ml/weka/arff.html.

[17] "Package weka.classifiers.trees,"Available: https://weka.sourceforge.io/doc.stable-3-8/weka/classifiers/trees/package-summary.html.

[18] S. Chang, Y. Shihong 和 L. Q, "Clustering Characteristics of UCI Dataset".IEEE.

[19] S. Fiori, "Neural Systems with Numerically Matched Input-Output Statistic: Isotonic Bivariate Statistical Modeling," Comput Intell Neurosci, 12 7 2007.*

[20] G. Piatetsky-Shapiro, "KDnuggets news on SIGKDD service award.," Available: https://www.kdnuggets.com/news/index.html.