# DATA MINING
# PROJECT REPORT

Data Set : Mushroom Data Set
Source : UCI Machine Learning Repository
URL : https://archive.ics.uci.edu/ml/datasets/Mushroom


Done By

Group 16

| | |
|---|---|
| B120458CS | Anjana Babu |
| B120239CS | Asla Aboo |
| B120086CS | Drishya Praveen C.P |
| B120668CS | Indu Sree |




Department of Computer Science and Engineering
National Institute of Technology Calicut, Kozhikode.

Date: 21st April 2015

# ABSTRACT

Our project aims at selecting a data set, cleaning it and mining it for useful patterns. The project gave us valuable insights into the field of data mining. Being beginners to this course we got a first-hand experience of what this field offers.

The data set we chose was “Mushroom dataset” and data mining task associated with it was classification. The problem was to classify a given mushroom sample as either poisonous or edible.

We got our hands dirty with data cleaning tools - Data Wrangler and Open Refine which enabled us to clean messy data efficiently and thus leaving us with more time to analyse the data and draw conclusions and identify patterns from it. The data set contained missing values and other faulty entries which was taken care of using these tools.

After cleaning data, we explored data mining tool WEKA in which we performed numerous classification methods in order to select a classifier which outperformed others and which served our purpose.

The data set which was given to us for modifications was “Don’t Get Kicked dataset” and the data mining task associated with it was clustering. The challenge of this data set was to predict if the car purchased at the Auction is a good / bad buy.

# TABLE OF CONTENTS

# INTRODUCTION

## ❖ Project Overview

Our project aims at developing the best classifier model using the data mining tool **WEKA**, *to predict whether a given mushroom is edible or not.*

Through this project, we intend to find the best classifier model by comparing some of the models offered by the tool. How do classifiers operate? What is the basis of comparison? How can we determine accuracy of each model? Can we increase accuracy? We delve into these aspects, as well as explore all other facilities provided by the tool.

The data set, **mushroom data set**, is taken from UCI Machine Learning Repository. The whole knowledge discovery process includes cleaning, integration, selection, transformation, mining, evaluation and presentation.

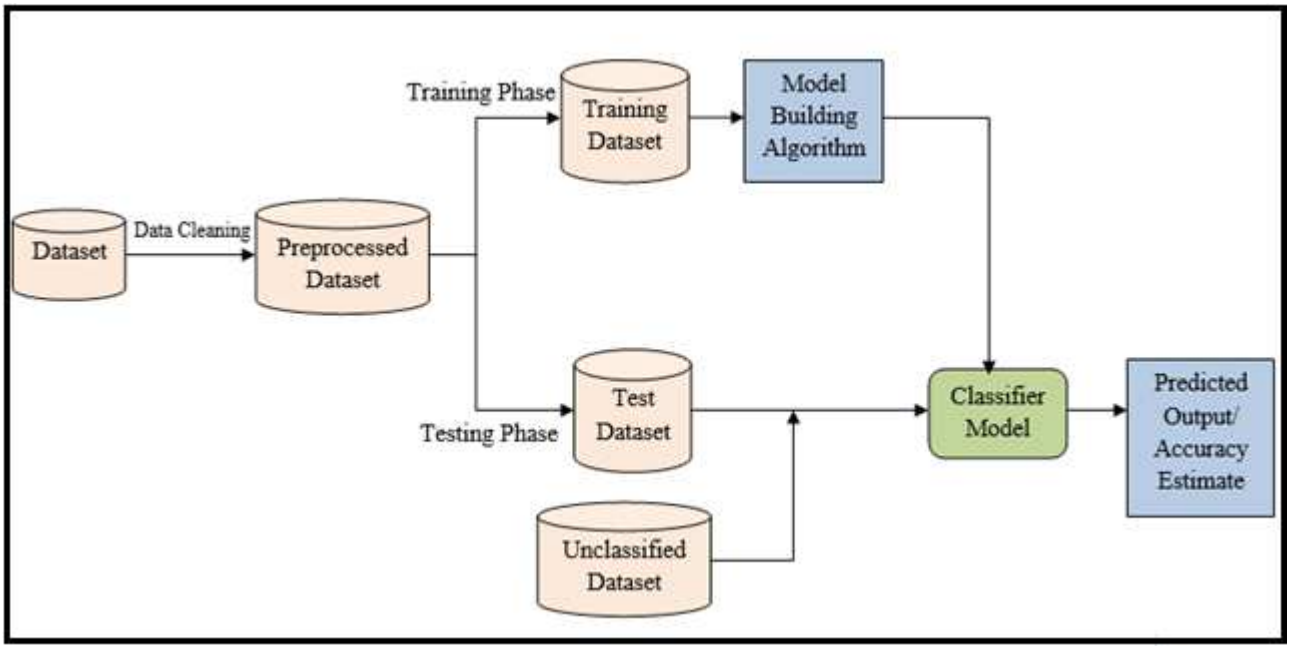
We start with cleaning the data set, using **Data Wrangler** tool, and proceed to study and implement the various techniques on the data set which includes identifying the problem domain, deciding on what you want to achieve with data mining, choosing appropriate methods and algorithms, implementing and testing your methods, evaluating your techniques on your data sets, reporting conclusions.

## ❖ Project Deliverables

- Data Set Selection
  - Mushroom Dataset was selected to work on.
- Data Pre-processing
  - Data Cleaning and Transformation: Raw dataset was cleaned and transformed to usable form.
  - Data Reduction: Attribute “veil-type” took only one distinct value, hence was removed.
  - Data Integration: The cleaned datasets were integrated to form the final cleaned data set to work with.
- Classification
  - Decision tree, Rule Based and Naïve Bayes classifier models were made and tested.
- Final Report
  - Final Report noting down our inferences and works was made and submitted.

# PROJECT ORGANISATION

- Process Model




- Role and Responsibility

| Team Members | Role & Responsibility |
|---|---|
| Anjana Babu | Leader, Integrate & Coordinate Work |
| Asla Aboo | Project Organisation |
| Drishya Praveen C.P. | Project Management |
| Indu Sree | Test Documentation |

## ❖ Tools and Techniques Used

### Data Cleaning Tool: Data Wrangler

Handles missing values by various techniques:

- Deleting tuples
- Copying the cell from the tuple above
- Copying the cell from the tuple below

### Data Mining Tool: WEKA

1. **Pre-process**
    - Filter out certain tuples or attributes alone
    - Remove attributes or tuples

2. **Classification**
   Process of creating classifier models from the training data set in the form of a decision tree or set of rules that can be used to predict the classes for test set.
    - **ZeroR Classifier**
      A primitive model based on single rule. For nominal attribute, majority class label is assigned whereas for numeric attribute the average value is assigned to the test set.
    - **Naïve Bayes Classifier**
      Predicts class membership probability for the new tuple in test set.
      $$P(X|H)\,P(H) = P(X_1|H)\,P(X_2|H)\ldots P(X_n|H)$$
    - **J48 Classifier & ID3 Classifier**
      Produces decision tree models. Provision to prune trees is also available.

3. **Testing Options Available**
    - Using Training Set
    - Using Test Set
    - Cross Validation
    - Percentage Split

4. **Visualization**
    - A Plot-Matrix and a Graph between various attributes are shown.
    - We can also select a portion from the graph which includes only some portion of the data set and analyse separately.
    - Jitter adds randomness to points. We can view the instances corresponding to a cross by clicking on it.

# PROJECT MANAGEMENT PLAN

- Tasks
  a. Cleaning
     1. Dealing with missing values
     2. Removing irrelevant attributes
  b. Mining
     1. Analysing data using data mining tool WEKA
     2. Determining the type of data mining task associated with the given data set
     3. Visualisation using WEKA
     4. Building algorithm based on result of task b.2
     5. Results validation
     6. Reporting conclusions

- Description of Plan
  a. Cleaning
     1. Missing values are dealt by replacing them with value of the attribute from the tuple above current tuple. Removing tuples with missing values is not a viable option since number of tuples with missing values is 2480 which is a quarter of the whole data set.
     2. Removing attribute veil-type as it takes only one value for all the tuples.
  b. Mining
     1. Analysing given data set using pre-processing tool of WEKA which displays histograms of attributes and class labels. This analysing part is important because often we can identify good data sets from the bad ones here.The characteristics of data set can be explored and various filters can be applied to the data.
     2. The given data set contains a class label. Therefore it is a supervised learning problem which rules out Clustering and Association. The class label is nominal and hence not Regression. Therefore our data mining task is Classification.
     3. Exploring given data set using visualization. Sometimes visualizations can reveal important patterns in data set. Scatter plot matrix of the data set is used to spot misclassified errors
     4. We have identified the data mining task at hand to be Classification. Therefore we are going to build a classifier. Classifiers can be built from existing algorithms or manually. Classifiers we are going to use include:
        i. Tree based classifiers like ID3,J48
        ii. Rule based classifiers like ZeroR , OneR
        iii. Naïve Bayes classifier
     5. To test the trained classifier on test set using validation techniques such as Cross Validation, Percentage Split or on training set itself or on a supplied test set.

## ❖ Data Set Description

**Dataset**: Mushroom Dataset

**Source**: http://archive.ics.uci.edu/ml/datasets/Mushroom

**Origin**: Mushroom records drawn from the The Audubon Society Field Guide to North American Mushrooms (1981). G H Lincoff (Pres.), New York :Alfred A Knopf.

**Donor**: Jeff Schlimmer

**Dataset Information:**

This data set includes descriptions of hypothetical sample corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflet three, let it be" for Poisonous Oak and Ivy

**Number of Instances**: 8124

**Number of Attributes**: 22 (all nominally valued)

**Class**: edible=e, Poisonous=p

**Attribute Information:**

1. Cap-shape: bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s.
2. Cap-surface: fibrous = f, grooves = g, scaly = y, smooth = s.
3. Cap-color: brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, yellow = y.
4. Bruises: bruises = t, no = f.
5. Odor: almond = a, anise = l, creosote = c, fishy = y, foul = f, musty = m, none = n, pungent = p, spicy = s.
6. Gill-attachment: attached = a, descending = d, free = f, notched = n.
7. Gill-spacing: close = c, crowded = w, distant = d.
8. Gill-size: broad = b, narrow = n.
9. Gill-color: black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y.
10. Stalk-shape: enlarging = e, tapering = t.
11. Stalk-root: bulbous = b, club = c, cup = u , equal = e, rhizomorphs = z, rooted = r, missing = ?.
12. Stalk-surface-above-ring: fibrous = f, scaly = y, silky = k, smooth = s.
13. Stalk-surface-below-ring: fibrous = f, scaly = y, silky = k, smooth = s.
14. Stalk-color-above-ring: brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y.
15. Stalk-color-below-ring: brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y.
16. Veil-type: partial = p, universal = u.
17. Veil-color: brown = n, orange = o, white = w, yellow = y.
18. Ring-number: none = n, one = o, two = t.
19. Ring-type: cobwebby = c, evanescent = e, flaring = f, large = l, none = n, pendant = p, sheathing = s, zone = z.
20. Spore-print-color: black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, purple = u, white = w, yellow = y.

21. Population: abundant = a, clustered = c, numerous = n, scattered = s, several = v, solitary = y.
22. Habitat: grasses = g, leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d.

**Missing Attribute Values**: 2480 of them (denoted by "?"), all for attribute #11.

**Class Distribution:**

Edible: 4208 (51.8%)
Poisonous: 3916 (48.2%)
Total: 8124 instances

## ❖ Deliverables and Milestones

| No. | Milestone | Deliverable | Date |
|---|---|---|---|
| 1. | Submission of dataset | Mushroom dataset selected | March 14, 2015 |
| 2. | Completion of Data Cleaning | Data Cleaning completed | March 15, 2015 |
| 3. | Completion of Data Mining | Data Mining completed | March 20, 2015 |
| 4. | Early Submission of Project | Project submitted | March 23, 2015 |
| 5. | Final submission of Project | Report submission on Mushroom dataset and modifications | April 21, 2015 |

## ❖ Dependencies and other constraints

- Dataset for data mining using WEKA requires pre-processed data from Data Wrangler.
- Data Wrangler supports only 1000 rows & 40 columns at a time.
- Data Wrangler is available only online
- Dataset had only nominal attributes, so could try out only classification methods.

# REQUIREMENT SPECIFICATION

## ❖ Hardware Requirements

- Secondary Storage: Min 80MB to install WEKA; Data Wrangler is an on-line software tool.
- Peripherals : Keyboard, Mouse
- Memory : 512MB RAM for WEKA

## ❖ Specific Requirements

### Data Cleaning Tool: Data Wrangler (Online only)

- **User Interface**
  - Data Wrangler, the web-based app that provides an interactive GUI front end that is used to define the procedures that will ultimately manipulate the data.
  - Each manipulation provides a preview of the output, and there is always a history of operations that can be deselected or deleted.
  - Filling in missing data or incorporating data from another source becomes a straight-forward process.
  - The output of this web app is either a CSV of your manipulated data, or a script that can be used to manipulate similar data.
- **Software Interface**
  - Data can be exported to use in R, tableau, protovis.
  - R functionality has been made accessible from several scripting languages such as Python. The python script output from data wrangler can be utilized in R for further analysis. Data can be imported to R in JSON, CSV formats also.
  - Protovis uses JavaScript and SVG for web-native visualizations. The JavaScript output from data wrangler can be made use in protovis.
  - There is no provision to connect to MATLAB.
- **Other details regarding the tool**
  - Data Wrangler builds on ideas published over ten years ago in the prescient Potter's Wheel system, as well as great work in the area of "programming by demonstration"

### Data Mining Tool: WEKA

- **User Interface**
  - Explorer - Widely Used
    Panels featured here include:
    - *The Pre-process panel*: has facilities for importing data from a database, a CSV file, etc., and for pre-processing this data using a so-called filtering algorithm. These filters can be used to transform the data (e.g., turning numeric attributes into discrete ones) and make it possible to delete instances and attributes according to specific criteria.
    - *The Classify panel*: enables the user to apply classification and regression algorithms to the resulting dataset, to estimate the accuracy of the resulting predictive model, and to visualize erroneous predictions, ROC curves, etc., or the model itself.

- *The Associate panel*: provides access to association rule learners that attempt to identify all important interrelationships between attributes in the data.
- *The Cluster panel*: gives access to the clustering techniques in WEKA, e.g., the simple k-means algorithm. There is also an implementation of the expectation maximization algorithm for learning a mixture of normal distributions.
- *The Select attributes panel*: provides algorithms for identifying the most predictive attributes in a dataset.
- *The Visualize panel:* shows a scatter plot matrix, where individual scatter plots can be selected and enlarged, and analysed further using various selection operators.

  - Experimenter - For systematic comparison of the predictive performance of WEKA's machine learning algorithms on a collection of datasets.
  - Knowledge Flow - Graphical Interface
  - Simple CLI - Command Line Interface

- **Software Interface**
  - WEKA's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes**.**
  - WEKA provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query.
- **Other details regarding the tool**
  - Open Source Software Tool for Data Mining

## ❖ About the Software

### Data Cleaning Tool: Data Wrangler

- Introduction

  Data Wrangler is an interactive tool for data cleaning and transformation developed by Stanford Visualization Group. Spend less time formatting and more time analysing your data. Wrangler allows interactive transformation of messy, real-world data into the data tables analysis tolls expect. Export data for use in Excel, R, and Tableau.

- Reliability

  Data Wrangler is a reliable data cleaning tool. Output of the cleaning actions is made available then and there which confirms to the change and adds to the reliability.

- Availability

  The tool is a web-app which restricts its availability to on-line which makes it inconvenient to use for offline purposes. A stand-alone desktop version is not yet available.

- Security

  As far as the security is concerned, being a web-based app it sends your data off to an external site which means it not an option for sensitive internal information and hence not secure.

- Maintainability
  The Wrangler research project is complete, and the software is no longer actively supported. The team behind Wrangler has moved on to work on a commercial venture, Trifacta. Therefore there is no full time team dedicated to fix bugs and repairs.

- Portability
  Since Data Wrangler is a web-app portability of it only requires a good internet connection. It is supported in most of the browsers like Google Chrome, Mozilla Firefox etc.

- Performance
  Performance for a data set of small size if really remarkable, but Data Wrangler allow to import data sets with maximum 1000 tuples with 40 attributes at a time. Evaluating performance with small data sets, text editing is easy and provides us with suggestions for actions to be performed on selected tuples. Changes are made very fast.

## Data Mining Tool: WEKA

- Introduction
  WEKA (Waikato Environment for Knowledge Analysis) is a data mining tool developed at the University of Waikato, New Zealand. WEKA is written in Java and is a comprehensive collection of data pre-processing and modelling techniques. WEKA supports several standard data mining tasks like data pre-processing, clustering, classification, regression, visualisation, and feature selection.

- Reliability
  WEKA provides us with much reliable results since the algorithms are well studied and built in.

- Availability
  It is freely available under the GNU General Public License which makes it more popular and useful.

- Security
  Being a stand-alone application, WEKA is secure.

- Maintainability
  Releasing WEKA as open source software and implementing it in Java ensure that it remains maintainable and modifiable irrespective of the commitment or health of any particular institution or company.

- Portability
  Since it is fully implemented in the Java Programming Language and thus runs on almost any modern computing platform, hence is portable.

- Performance
  Coming to performance, WEKA computes results fast and displays in an organised fashion. It is easy to use because of its graphical user interface.

# DATA BASE REQUIREMENTS

One exceptional feature of WEKA is the database connection using JDBC with any RDBMS package. If the data collected is in database formats like MySQL, MS Access, DB2 or any other, then there is no need to worry about whether we will have to convert the millions of datasets in the WEKA-specific file formats. We can simply connect WEKA with any database management software using JDBC, and can import any number of datasets of that database server. JDBC is a java database connectivity technology that can access any type of tabular data, especially data stored in RDBMS.

- ## Importing dataset to a database

(Ref: http://cran.r-project.org/doc/contrib/Zhao_R_and_data_mining.pdf)

The example below creates a data frame df1 and save it as a .CSV file with write.csv () and then, the data frame is loaded from file to df2 with read.csv ().

```
> var1 <- 1:5
> var2 <- (1:5) / 10
> var3 <- c("R", "and", "Data Mining", "Examples", "Case Studies")
> df1 <- data.frame(var1, var2, var3)
> names(df1) <- c("VariableInt", "VariableReal", "VariableChar")
> write.csv(df1, "./data/dummmyData.csv", row.names = FALSE)
> df2 <- read.csv("./data/dummmyData.csv")
> print(df2)
  VariableInt VariableReal VariableChar
1           1          0.1            R
2           2          0.2          and
3           3          0.3  Data Mining
4           4          0.4     Examples
5           5          0.5 Case Studies
```

- ## Connecting the database to WEKA

1. **Create a ODBC (Open Database Connectivity) and DSN data source**
   - ODBC is a standard programming language middleware API for accessing database management systems
   - DSN data source data is a structure that contains the information about a specific database that an ODBC driver needs in order to connect to it.
2. **Create DSN**
   - Open ODBC sources, give the data source a name and a description and select that database.
3. **Configure WEKA**
   - To support SQL by opening the file DatabaseUtils.props.msmsqlserver2005" which is present in weka.jar file
   - Edit the file by adding some conversions including bit=1 means 'convert SQL Server bit data types to WEKA Boolean' and save it.
4. **Connect with WEKA**
   - Open WEKA Explorer and go to Open DB button, we will get a window named SQL-viewer. Press the "User…" button and specify the database URL: "jdbc: odbc: User DSN Name". Then press "OK".
   - Press "Connect" and WEKA will connect to the database server.
   - Now type in your SQL-statement in the query textbox and press "Execute". Data is now fetched from the SQL Server and the result is shown in the result window. If you are satisfied with the result then press "OK".
   - Data is now loaded into the WEKA Explorer and you can start doing your data analysis and mining.

# DESIGN SPECIFICATIONS

## ❖ Design Overview

1. Data Cleaning: The unclean dataset is cleaned with data cleaning tool Data Wrangler

2. Data Mining: The clean data set is fed to data mining tool WEKA where the given data set is subjected to various classification methods.
   There are basically three types of classifiers on which we tried our data set on.
   - Decision Tree Classifier: J48, ID3
   - Naïve Bayes Classifier
   - Rule based: ZeroR, OneR

   The evaluation of classifiers is done using four evaluation methods.
   - Against Training Set
   - Against Test Set
   - Percentage Split
   - Cross Validation

3. Data Visualisation: Done using Scatter Plot Matrix

## ❖ Work Done

### 1. Chosen Tool Set

- Data Cleaning Tool: Data Wrangler
  - **Pros**
    - Online web-app, therefore portable
    - Intuitive
    - Preview of output available, hence reliable
    - Output : CSV/Script
    - Maintains record of manipulations on dataset
      - Helpful to delete/deselect manipulations
      - This record can be applied to similar datasets
      - This record is a script(Python/JavaScript)

  - **Cons**
    - Data Wrangler supports only 1000 rows & 40 columns at a time, so difficult to cleaning for large datasets.
    - Not secure, since we have to uploading our dataset to the data wrangler server.

Due to the cons of Data Wrangler and since our data set was of large size we shifted to Open Refine, a data cleaning tool by Google for wrangling of new dataset.

- Data Cleaning Tool: Open Refine
  - **Pros**
    - Ability to extend data set with external sources
    - Can handle large data sets easily
    - Robust
  - **Cons**
    - Non intuitive
- Data Mining Tool: WEKA
  - **Pros**
    - 4 GUI interfaces to choose from
    - Provides access to SQL databases
    - Easily available
    - Secure
    - Portable
  - **Cons**
    - Memory bound
    - Incapable of multi-relational data mining
    - Algorithms provided by WEKA doesn't cover sequence modelling

## 2. Classification Models

- Decision Tree Classifiers
  - ID3

    The ID3 algorithm begins with the original set $S$ as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set $S$ and calculates information gain of that attribute. It then selects the attribute which has the maximum information gain value. The set $S$ is then split by the selected attribute to produce subsets of the data. The algorithm continues to recur on each subset, considering only attributes never selected before.
  - J48 Tree Classifier – extended C4.8

    Follows same algorithm as ID3 except splitting criterion is Gain Ratio

- Naïve Bayes Classifier

  Based on Bayes theorem. This classifier assumes that an attribute's value is independent of values of other attributes (class conditional independence). The Zero Frequency Problem: Add 1 to the count for every attribute value-class combination (*Laplace estimator*) when an attribute value doesn't occur with every class value.

- Rule Based Classifier
  - ZeroR

    ZeroR classifier simply predicts the majority class ignoring all other attributes. Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods (baseline accuracy).
  - OneR

    This algorithm generates a rule for each attribute and selects the rule with minimum error as its one rule.

## 3. Evaluation Methods

- Testing against Training set
  Classifier build on training set i.e. our data set is tested against the same training set.

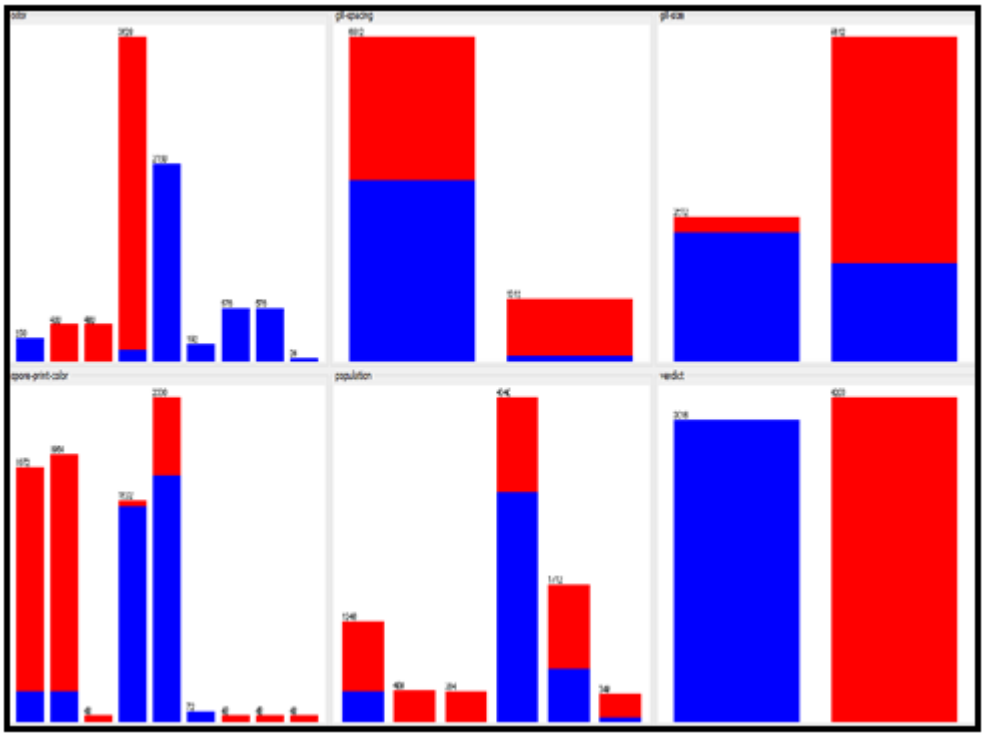
- Testing against Test set
  Classifier build on training set i.e. our data set is tested against an independently supplied test set.

- Percentage Split
  Dataset is divided it into two parts one for training and the other for testing -- perhaps 2/3$^{rd}$ of it for training and 1/3$^{rd}$ of it for testing. It's really important that the training data is different from the test data.

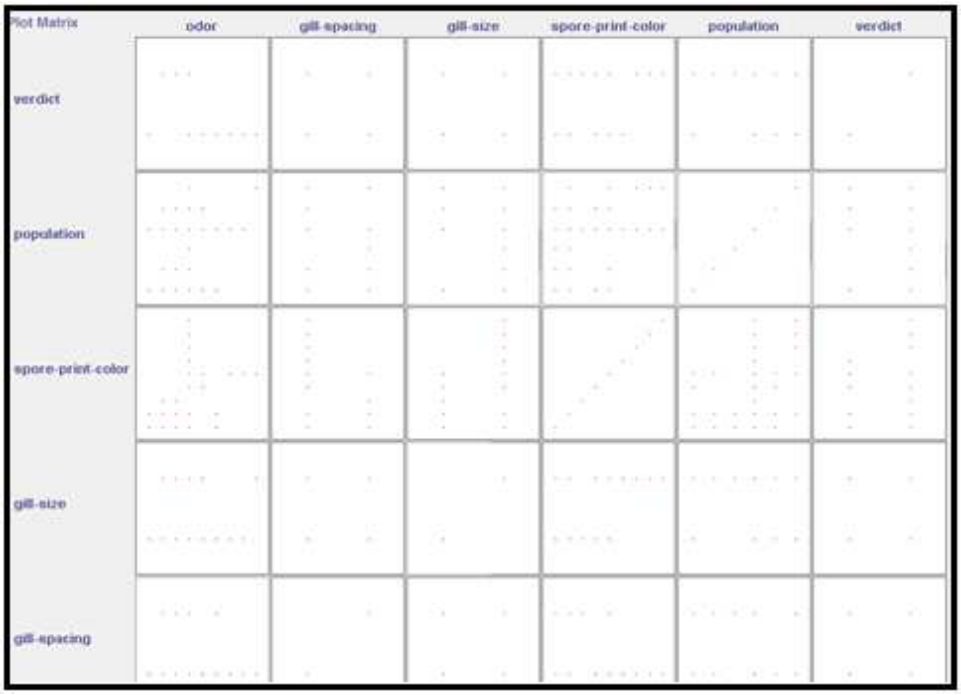
- Cross Validation
  In k- fold cross validation, the initial data is randomly partitioned into k- mutually exclusive subsets/folds, each of approximately equal size. Training and testing is performed k times. Each sample is used the same number of times for training and once for testing.

## 4. Data Visualization

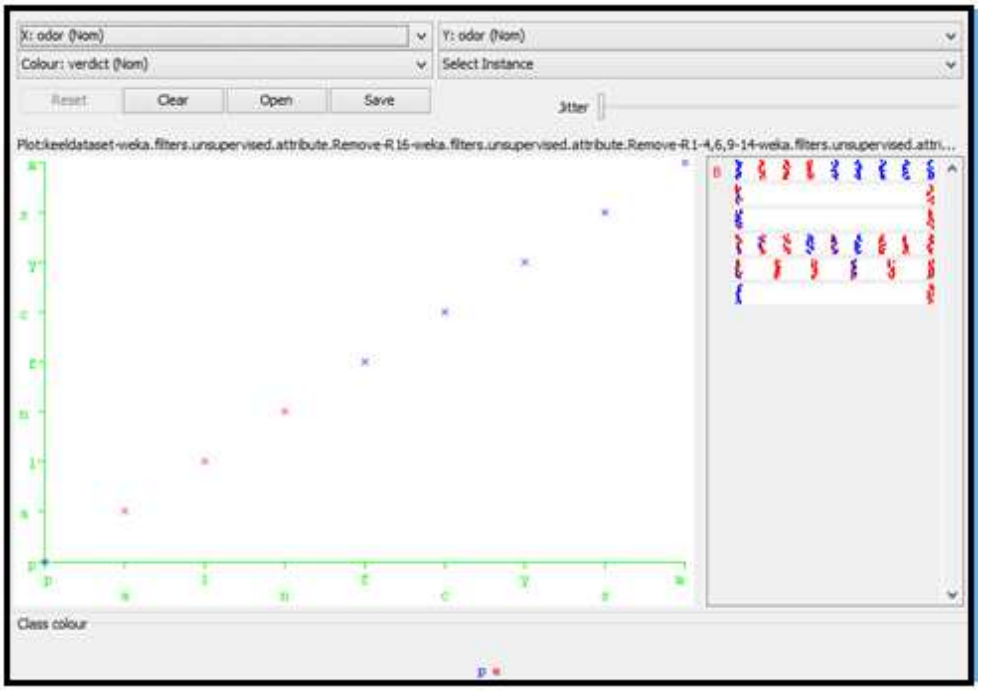
- Graph representing each attribute value and class distribution is obtained.



- A plot matrix (i.e $n_x n$ matrix where n denotes number of attributes) is obtained




- Graph between various attributes are also shown to get a better understand ability of data set. We can also select a portion from the graph which includes only some portion of the data set and analyse separately

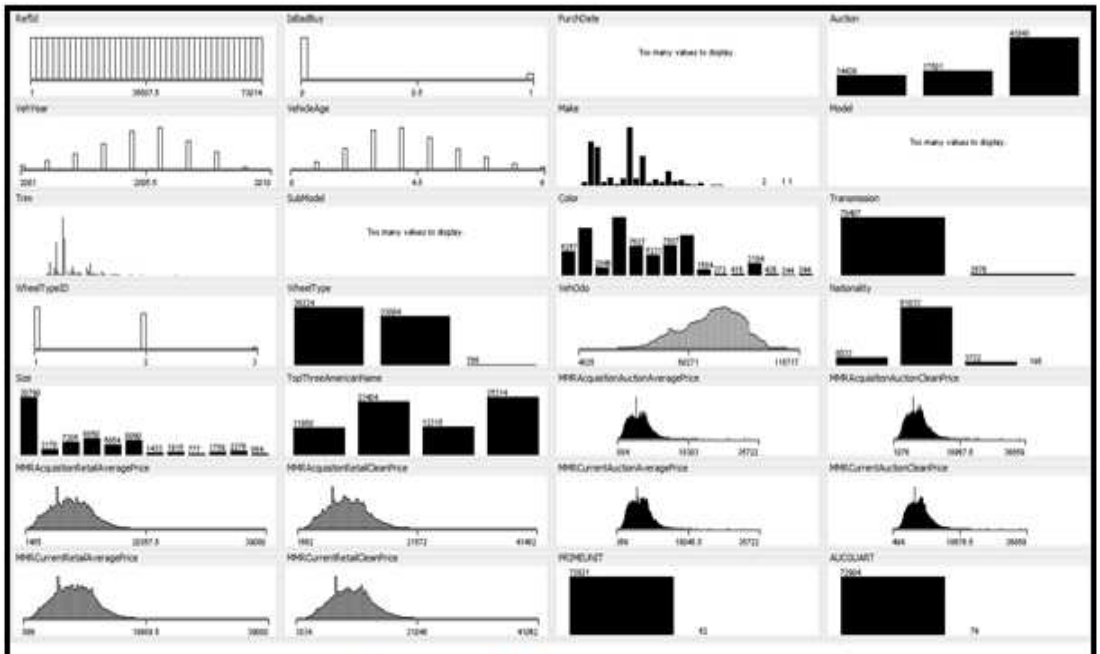


# MODIFICATIONS DONE

## ❖ Data Cleaning

- The new dataset was cleaned using Google Open Refine on 18th April 2015.
- Each attribute value was studied and cleaned properly.
- Null & blank values in the attributes were replaced with the most frequent attribute value
- Date was formatted to DD-MM-YY
- The class label was transformed from 0/1 to yes/no since WEKA was taking the attribute as numeric attribute.

## ❖ Data Visualisation



*Visualisation of Data Set - Attribute Value v/s #of Instances*

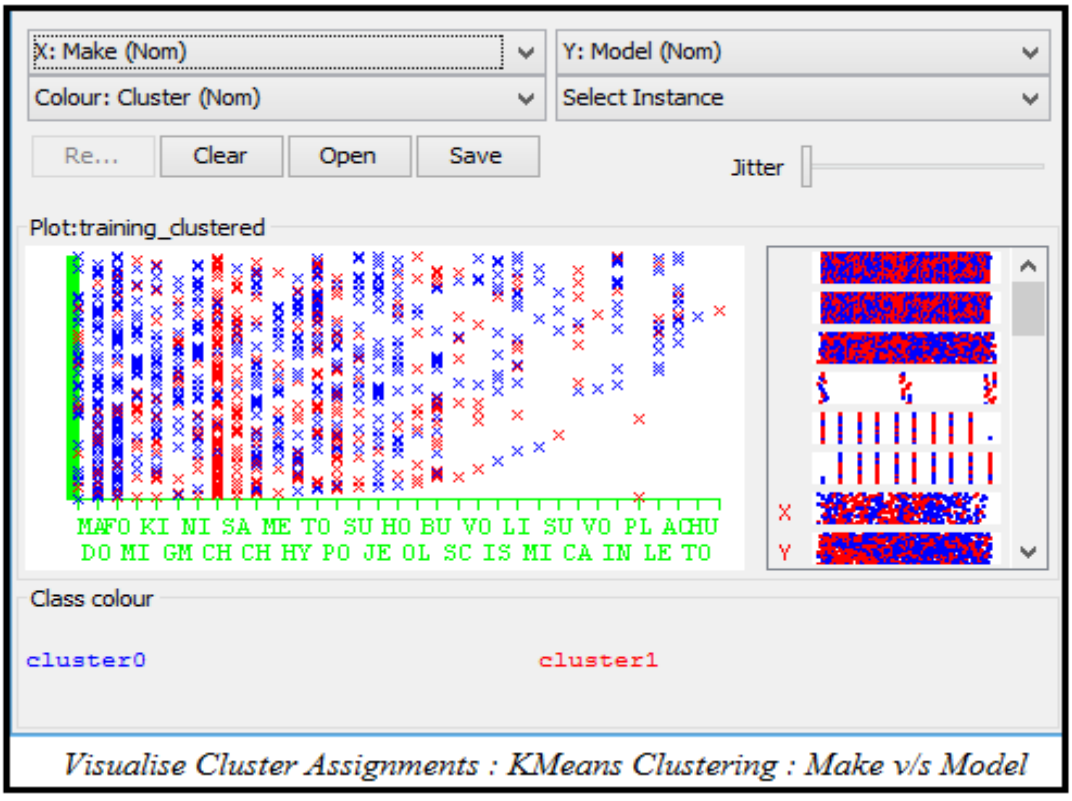
## ❖ Clustering

### 1. K-Means Clustering

- K-means clustering is a method of cluster analysis which aims to partition n observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.
- Specify *k* which is the number of clusters and choose *k* random points as centre of these clusters. Assign each instance to its nearest cluster. Calculate new centre of the cluster by taking mean of clustered instances. With the new centres repeat the steps until there is no change in cluster centres.
- We need to minimize sum of squared error within clusters.

2 Cluster – 10 Seed

| **Cluster Mode** | **Clustered Instances** | | | | **Sum of Squared Errors within Clusters** |
|---|---|---|---|---|---|
| *Using Training Set* | Cluster 0 | | 36774(50%) | | 585677.7 |
| | Cluster 1 | | 36209(50%) | | |
| *Percentage Split (66%)* | Cluster 0 | | 13202(53%) | | 392526.7 |
| | Cluster 1 | | 11613(47%) | | |
| *Classes to Cluster Evaluation* | Correctly clustered instances 38361 (52.561%) | | Cluster | | 585677.7 |
| | | | 0 (Yes) | 1 (No) | |
| | IsBadBuy | Yes | 5564 | 3412 | |
| | | No | 31210 | 32797 | |




*Visualise Cluster Assignments : KMeans Clustering : Make v/s Model*

## 2. Density Based Clustering

DBSCAN-Density Based Clustering Spatial Clustering of Applications with Noise

- Density Based Clustering clusters objects based on density. It either grows clusters according to the density of neighbourhood objects or according to some density functions.
- To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions. This is the main strategy behind density-based clustering methods, which can discover clusters of non-spherical shape.
- DBSCAN finds core objects, i.e., objects that have dense neighbourhoods. It connects core objects and their neighbourhoods to form dense regions as clusters.

| **Clustered Instances** | | | |
|---|---|---|---|
| 0 | 13 (9%) | 9 | 7 (5%) |
| 1 | 7 (5%) | 10 | 8 (6%) |
| 2 | 6 (4%) | 11 | 6 (4%) |
| 3 | 7 (5%) | 12 | 10 (7%) |
| 4 | 6 (4%) | 13 | 9 (6%) |
| 5 | 9 (6%) | 14 | 12 (9%) |
| 6 | 6 (4%) | 15 | 12 (9%) |
| 7 | 9 (6%) | 16 | 7 (9%) |
| 8 | 7 (5%) | | |

# of attributes = 33
Epsilon(r) = 0.9
MinPoints = 6
(Min # of data objects required within the given *r*)

Number of clustered instances : 17
Un-clustered instances : 19859
(Are indicated as noise)

| **Epsilon(r)** | **MinPoints** | **# of clusters** | **# of un-clustered instances** |
|---|---|---|---|
| 0.5 | 6 | 17 | 19860 |
| 1.1 | 8 | 42 | 19156 |

**Inference:** To reduce the number of outliers it is advisable to increase the radius value and the number of minimum points to come within that range.

# ❖ Correlation & Attribute Evaluator

## 1. Attribute Evaluator: $\chi^2$

- Evaluates the worth of an attribute by computing the value of $\chi^2$ statistic with respect to the class. A good feature set contains attributes that are highly correlated with the class, yet uncorrelated with each other.
- Search Method : Ranker (Ranks attributes by their individual evaluations of $\chi^2$)
- Selected Attributes

| Attribute | Chi Square Value | Attribute | Chi Square Value |
|---|---|---|---|
| Model | 3551.93503 | Trim | 805.1472 |
| SubModel | 2711.4782 | VNZIP1 | 669.1753 |
| VehicleAge | 2081.18939 | BYRNO | 624.86217 |
| VehYear | 1904.49756 | VehOdo | 590.30151 |
| MMRAcquisitionAuctionAveragePrice | 1649.35319 | Make | 466.21404 |
| VehBCost | 1585.03199 | RefId | 316.0051 |
| MMRAcquisitionAuctionCleanPrice | 1583.54698 | Size | 309.01163 |
| MMRCurrentAuctionAveragePrice | 1559.15313 | VNST | 224.99625 |
| MMRCurrentRetailAveragePrice | 1280.05123 | TopThreeAmericanName | 199.92448 |
| MMRCurrentRetailCleanPrice | 1266.97833 | Auction | 140.53951 |
| PurchDate | 1123.96315 | Color | 54.063 |
| WarrantyCost | 1070.05242 | Nationality | 13.40941 |
| MMRAcquisitionRetailAveragePrice | 1029.94943 | Transmission | 1.18416 |
| WheelType | 1016.16957 | AUCGUART | 0.34595 |
| WheelTypeID | 1016.16957 | PRIMEUNIT | 0.0585 |
| MMRAcquisitonRetailCleanPrice | 946.90275 | IsOnlineSale | 0 |

## 2. Correlation based Feature Subset (CFS) Evaluator

- Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature (correlation between attribute and class) along with the degree of redundancy between them.
- Search Method : Best First (Searches the space of attribute subsets by greedy hill climbing, augmented with a backtracking facility)
- Selected Attributes
  - VehYear
  - VehicleAge
  - WheelTypeID
  - WheelType
  - VehOdo
  - MMRAcquisitionAuctionCleanPrice
  - MMRCurrentAuctionAveragePrice
  - BYRNO
  - VehBCost
  - WarrantyCost

# TEST DOCUMENTATION

## ❖ Features to be Tested

To estimate how accurate the built classifier models are and to compare the accuracy of one classifier with another, the methods of testing that we adopted include:

- Testing against Training Data
- Testing against Test Data
- Percentage Splitting/Holdout Method
- Cross Validation/Repeated Holdout Method

## ❖ Test Cases

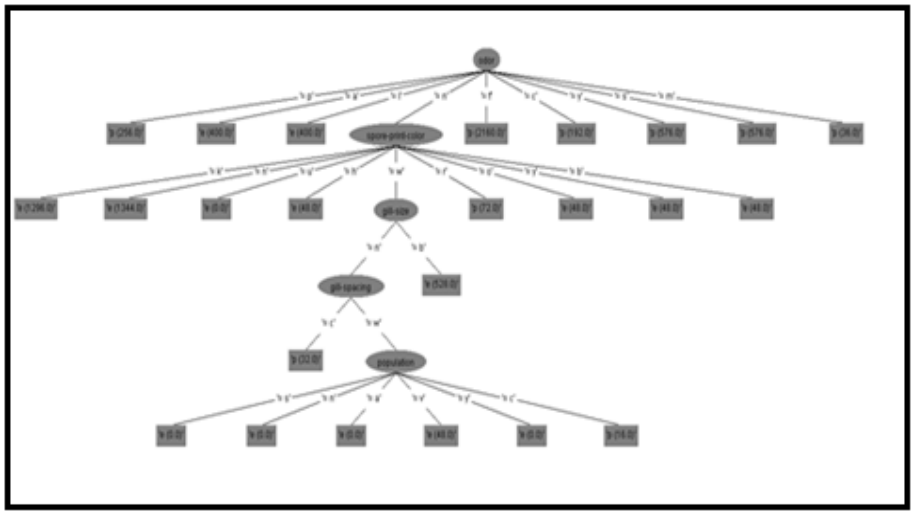
### 1. Testing against Training Set

- **Purpose**
  To evaluate classifier's accuracy.
- **Input**
  For model: Training set
  For testing: Training set
- **Expected Output**
  As is provided with the training set. Too large to list
- **Test Procedure**
  Training set is used for both building the classifier and for testing the model.
  Root : Odour
  #leaves : 24
  #nodes : 24+5
  Accuracy: 100% (as is tested against training set=>misleading results) \

| **=== Detailed Accuracy By Class ===** | | | | | | |
|---|---|---|---|---|---|---|
| TP Rate | FP Rate | Precision | Recall | F-Measure | ROC | Area Class |
| 1 | 0 | 1 | 1 | 1 | 1 | P |
| 1 | 0 | 1 | 1 | 1 | 1 | E |

| **=== Confusion Matrix ===** | | |
|---|---|---|
| A | B | Classified as |
| 3916 | 0 | a = p |
| 0 | 4208 | b = e |

Actual and Predicted results are same.
Diagonal: Correctly classified
Off-Diagonal: Incorrectly classified



## 2. Testing against Test Set

- **Purpose**
  To evaluate classifier's accuracy.
- **Input**
  For Model: Training Set
  For Testing: Test Set
  i. p, w, b, u, n, ?
  ii. y, w, n, u, v, p
  iii. a, c, n, k, s, p
  iv. f, w, b, h, v, e
  v. m, w, b, b, c, e
- **Expected Output**
  i. p or e
  ii. p
  iii. p
  iv. e
  v. e
- **Test Procedure**
  We estimate classifier's accuracy on a supervised test set that were not used to train the model. Training set and the test set should be of the same format and should be independent for reliable results .For each tuple we compare classifier's class label prediction with the tuple's known class label.

| **=== Predictions on test split ===** | | | | |
|---|---|---|---|---|
| Inst# | Actual | Predicted | Error | Probability Distribution |
| 1 | ? | 1:p | + *1 | 0 |
| 2 | 1:p | 1:p | *1 | 0 |
| 3 | 1:p | 2:e | +0 | *1 |
| 4 | 2:e | 1:p | +*1 | 0 |
| 5 | 2:e | 1:p | +*1 | 0 |
| + indicates error i.e. actual and predicted doesn’t match | | | | |

Accuracy = 25%

| **=== Confusion Matrix ===** | | |
|---|---|---|
| A | B | Classified as |
| 3916 | 0 | a = p |
| 0 | 4208 | b = e |

| **=== Detailed Accuracy By Class ===** | | | | | | |
|---|---|---|---|---|---|---|
| TP Rate | FP Rate | Precision | Recall | F-Measure | ROC | Area Class |
| 0.5 | 1 | 0.33 | 0.5 | 0.4 | 0.25 | P |
| 0 | 0.5 | 0 | 0 | 0 | 0.333 | E |
| 0.25 | 0.75 | 0.167 | 0.25 | 0.2 | 0.292 | Weighed Avg |

## 3. Percentage Splitting/Holdout Method

- **Purpose**
  To evaluate classifier's accuracy.
- **Input**
  For Model: 2/3 of the data is Training set
  For Testing: 1/3 of the data
- **Expected Output**
  As is provided with the test set. Too large to list
- **Test Procedure**
  66% of the training set is used to derive the model and the model's accuracy is tested with the test set (remaining 34% of the training set).The estimate is pessimistic because only a portion of the initial data set is used to derive the model.
    - At 66%
        - Test set is 2762 tuples
        - Accuracy = 100%
    - At 20%
        - Test set is 6492 tuples
        - Accuracy = 99.8923%
    - At 10%
        - Test set is 7304 tuples
        - Accuracy = 99.8906%

  For increased performance and reliability of the estimate, the training data set should be of large size. So that you can create a better classifier. As % split increases from 10% to 20% and so on, general trend is increased accuracy.

## 4. Cross Validation/Repeated Holdout Method

- **Purpose**
  To evaluate classifier's accuracy.
- **Input**
  Initial data set is randomly partitioned into ‘k’ mutually exclusive subsets or folds of equal size. Cross validation folds = 10.
  For Model: k-1 partitions
  For Testing: 1 partition
- **Expected Output**
  As is provided with the test set. Too large to list
- **Test Procedure**
  Training and testing is performed k times. In iteration i, $D_i$ is reserved as the test set and the remaining partitions are used to train the model. Each sample is used same number of times for training and once for testing. In Stratified Cross Validation (each portion contains equal proportion of class labels) reduces variations in estimate even further (default in WEKA). In WEKA, 10 times 90% of data goes into training & $11^{th}$ time 100% goes into training.

  Our data set gave 100% accuracy for 10, 5 folds and with different random seeds. Reason for this is because our test data set & training data set was not independent. If you have large data set we need to use % split else cross validation with 10 folds. 20 fold takes more time.

| **=== Confusion Matrix ===** | | |
|---|---|---|
| A | B | Classified as |
| 3916 | 0 | a = p |
| 0 | 4208 | b = e |

## ❖ Test Logs

| Odor | Gill-spacing | Gill-size | Spor-prnt-color | Popu lation | Actual | Predicted Label | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | J48 | ID3 | Naïve | OneR | ZeroR |
| y | W | n | U | V | p | p | p | P | p | E |
| a | C | n | K | S | p | e | e | E | e | e |
| f | W | b | H | V | e | p | p | P | p | e |
| m | W | b | B | C | p | p | p | P | p | e |
| m | C | n | O | A | p | p | p | P | p | e |
| m | W | n | K | N | e | p | p | P | p | e |
| l | C | b | K | N | p | e | e | E | e | e |
| f | W | n | u | V | p | p | p | P | p | e |
| P | C | b | k | S | e | p | p | P | p | e |
| s | C | n | u | V | p | p | p | P | p | e |

Comparative Study of Classifiers

| **Testing** | **Decision Tree** | | **Naïve Bayes** | **Rule Based** | |
|---|---|---|---|---|---|
| | **J48** | **ID3** | | **OneR** | **ZeroR** |
| Against Training Set | 100% | 100% | 99.2122% | 98.53% | 51.79% |
| Against Test Set | 50% | 50% | 40% | 50% | 30% |
| Percentage Split (66%) | 100% | 100% | 99.2397% | 98.62% | 51.05% |
| Cross Validation(10) | 100% | 100% | 99.2122% | 98.53% | 51.79% |