PREDICTIVE ANALYSIS PROJECT REPORT

(Project Semester August-December 2022)

Comparative Analysis on Wine Dataset

Submitted by

Shruti Tandon

Registration No. 11902188

Section: KM048

Course Code: INT-234

Under the Guidance of

Baljinder Kaur (27952)

Integrated B. Tech CSE-MBA

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

CERTIFICATE

This is to certify that Shruti Tandon bearing Registration no. 11902188 has completed INT-234 project titled, **Comparative Analysis on Wine Dataset** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort, and study.

Signature and Name of the Supervisor: Baljinder Kaur
Designation of the Supervisor
School of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab.

Date: 10.11.2022

DECLARATION

I, Shruti Tandon, student of Integrated B. Tech CSE-MBA under CSE Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 10.11.2022

Signature: 

Registration No. 11902188

Name of the student: Shruti Tandon

ACKNOWLEDGEMENT

Apart from the efforts of ours, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would also like to thank Baljinder Kaur ma'am for sharing her knowledge and guiding me throughout this project. Without her encouragement and guidance this report would not have been materialized. I take here a great opportunity to express my sincere and deep sense of gratitude to Lovely Professional University for giving me an opportunity to work on this project. The guidance and support received from all the members who contributed like my parents and classmates and the knowledge that I had gained from the Internet and the books was vital for the success of the project. I am grateful for their constant support and help.

TABLE OF CONTENTS

INTRODUCTION

For this project, we will be working on RStudio. R is a popular programming language used for statistical computing and graphical presentation. Its most common use is to analyze and visualize data. R is a programming language and software environment for statistical analysis and graphics representation. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand.

Following are some of the features of R:

- R is a well-developed, simple, and effective programming language which includes conditionals, loops, user defined functions and input and output facilities.
- R has an effective data handling and storage facility.
- R provides a suite of operators for calculations on arrays, lists, vectors, and matrices.
- R provides a large and integrated collection of tools for data analysis.

What are Variables in R?

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

To create a variable in R,

Follow this example:

var1 = c(1,2,3)

What are Data Types in R?

A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical ,relational or logical operations can be applied to it without causing an error.

Different data types in R can be character, integer, floating point, double floating point, Boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

In contrast to other programming languages like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.

In R, there are many types of R-objects. The frequently used ones are:

- Vectors: Vector is a sequence of data elements of the same data type.
- Lists: Lists are the R objects which contain elements of different types like − numbers, strings, vectors, and another list inside it.
- Matrices: Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types.
- Arrays: Arrays are the R data objects which can store data in more than two dimensions.
- Factors: Factors are the data objects which are used to categorize the data and store it as levels.
- Data Frames: A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

What is Predictive Analytics?

The term predictive analytics refers to the use of statistics and modeling techniques to make predictions about future outcomes and performance. Predictive analytics looks at current and historical data patterns to determine if those patterns are likely to emerge again.

This allows businesses and investors to adjust where they use their resources to take advantage of possible future events. It can also be used to improve efficiencies and reduce risk.

Predictive Analytics helps connect data to effective action by drawing reliable conclusions about current conditions and future events. It is the practice of extracting information from existing data sets in order to determine patterns and predict future outcomes and trends. Predictive analytics does not tell you what will happen in the future.

Predictive analytics encompasses a variety of statistical techniques from data mining, Artificial intelligence, and machine learning, that analyze current and historical facts to make predictions about future or otherwise unknown events.

Predictive models are used for all kinds of applications, including:

- Weather forecasts
- Creating video games
- Translating voice to text for mobile phone messaging
- Customer service
- Investment portfolio development

What is Machine Learning?

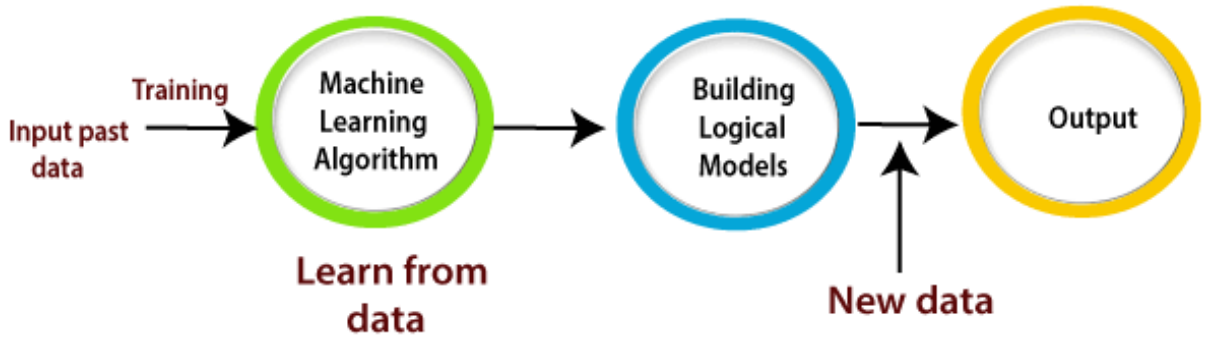
In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959.

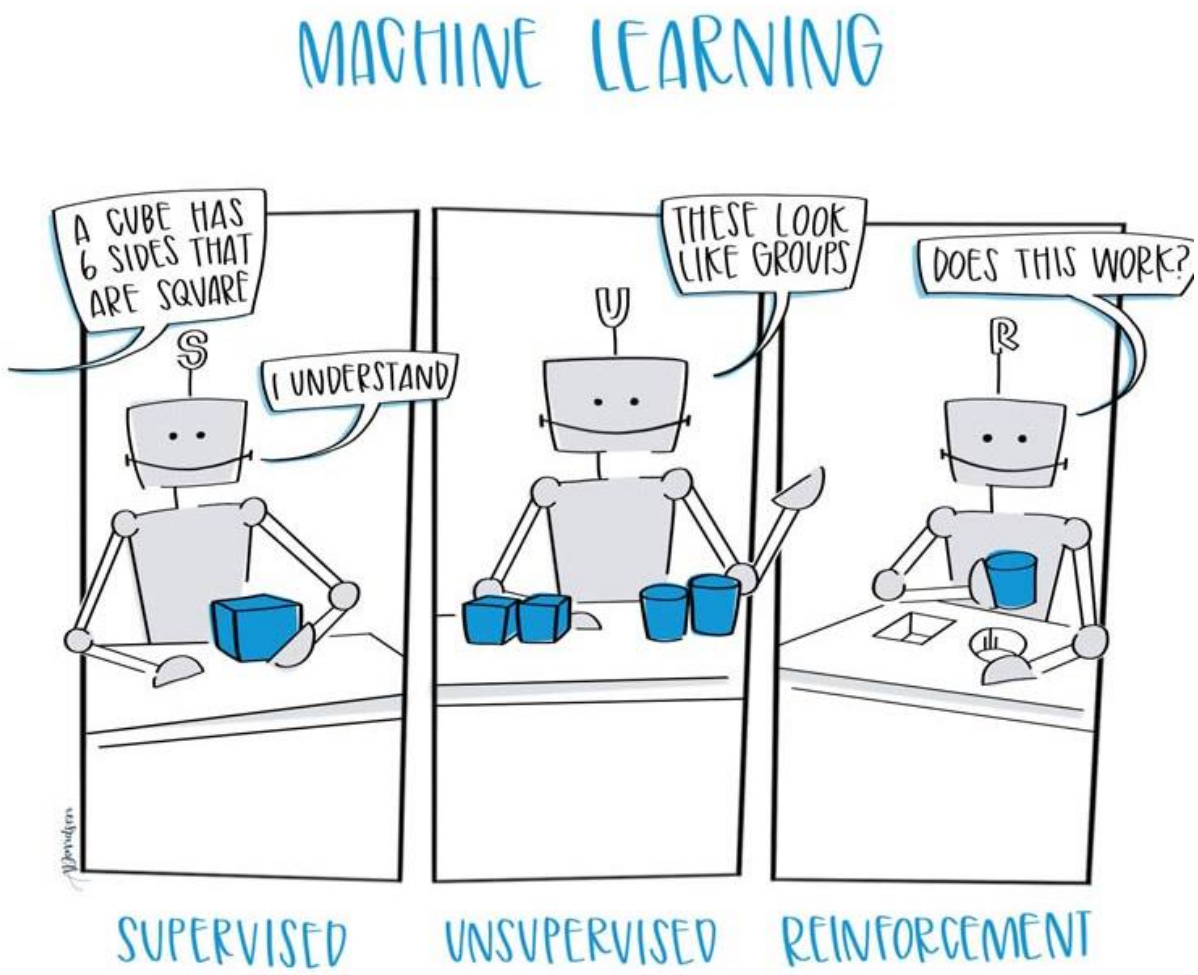
With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed.

How does Machine Learning work?

- A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.
- The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.
- Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem.



Machine Learning Techniques



- Supervised Learning

  Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

  The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

  The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher.

- Unsupervised Learning

  Unsupervised learning is a learning method in which a machine learns without any supervision.

  The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

- Reinforcement Learning

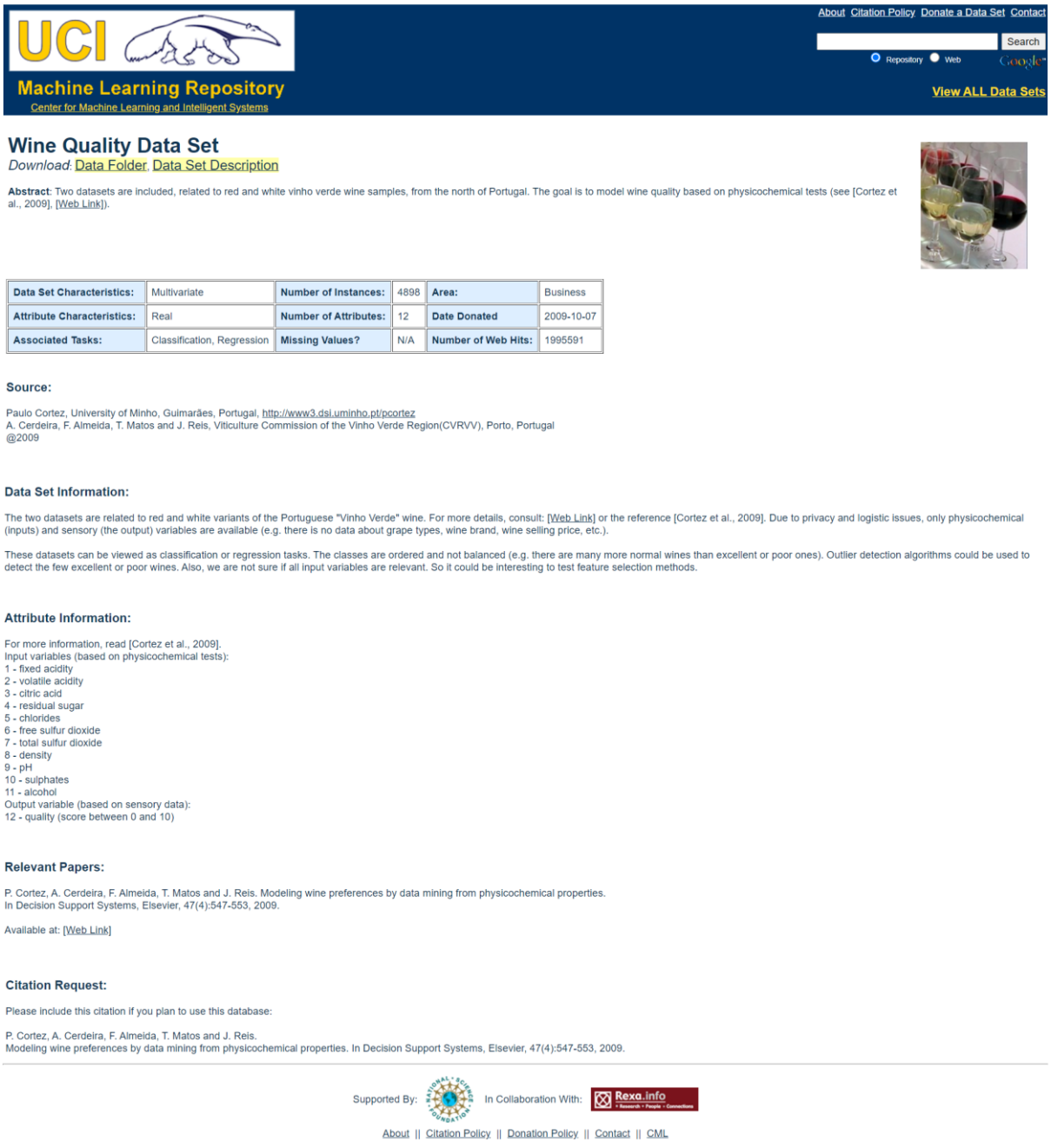
  Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

OBJECTIVES / SCOPE OF THE ANALYSIS

- Application of KNN model.
- Application of Decision Tree.
- Application of SVM model
- Application of ANN model.
- Comparative analysis of the above-mentioned Classification models.
- Application of MLR.
- Application of Random Forest.
- Comparative Analysis of the above-mentioned Regression models.

SOURCE OF DATASET

- **UCI Machine Learning Repository:** https://archive.ics.uci.edu/ml/datasets.php?format=&task=reg&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact
Search
Repository Web
View ALL Data Sets

Wine Quality Data Set

Download: Data Folder, Data Set Description

Abstract: Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], [Web Link]).

Data Set Characteristics:	Multivariate	**Number of Instances:**	4898	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	12	**Date Donated**	2009-10-07
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	1995591

Source:

Paulo Cortez, University of Minho, Guimarães, Portugal, http://www3.dsi.uminho.pt/pcortez
A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal
@2009

Data Set Information:

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult: [Web Link] or the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

Attribute Information:

For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

Relevant Papers:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Available at: [Web Link]

Citation Request:

Please include this citation if you plan to use this database:

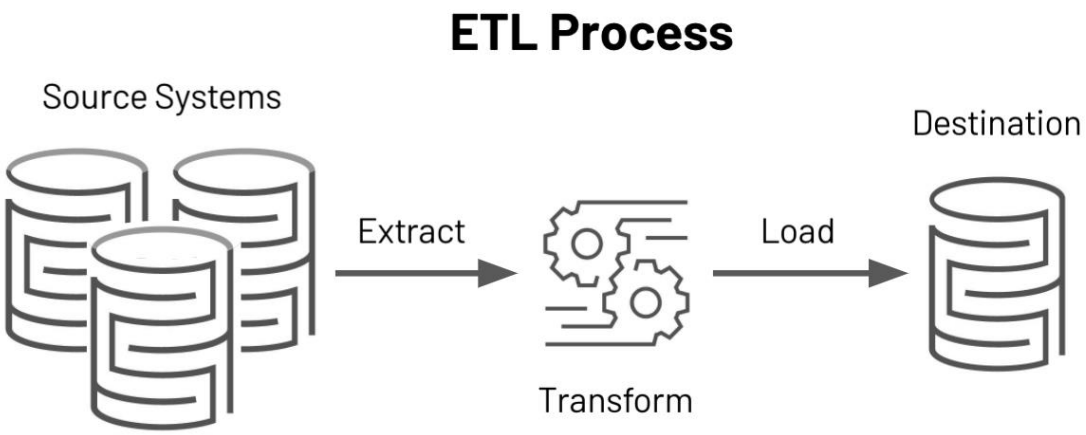
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Supported By: In Collaboration With: Rexa.info

About || Citation Policy || Donation Policy || Contact || CML

ETL PROCESS

- The mechanism of extracting information from source systems and bringing it into the data warehouse is commonly called ETL, which stands for Extraction, Transformation and Loading.
- The ETL process requires active inputs from various stakeholders, including developers, analysts, testers, top executives and is technically challenging.
- Three steps make up the ETL process and enable data to be integrated from source to destination. These are data extraction, data transformation, and data loading.



- **Extraction**

  In the first step, extracted data sets come from a source into a staging area. Structured and unstructured data is imported and consolidated into a single repository. Raw data can be extracted from a wide range of sources.

  Here input of dataset i.e., Extraction is taken from UCI Machine Learning Repository called Wine Quality Dataset which include raw data of , winequality-white and winequality-red. Among these I'll be working on winequality-white dataset.

- **Transformation**

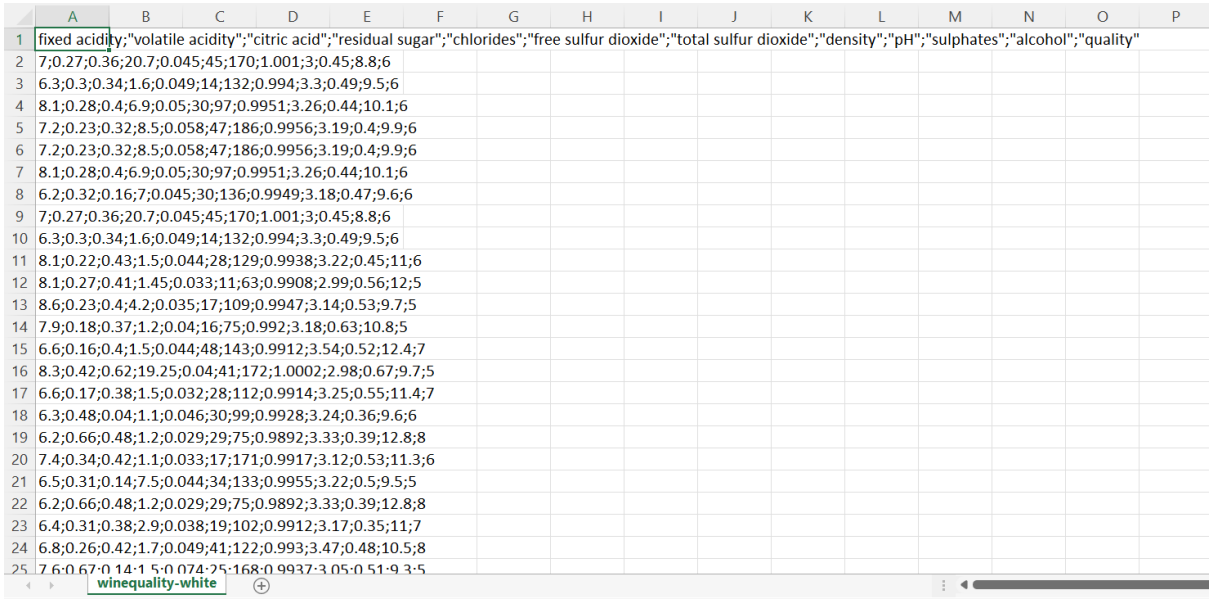
The data cleaning and organization stage is the transformation stage. All that data from multiple source systems will be normalized and converted to a single system format — improving data quality and compliance.

ETL yields transformed data through these methods:

- Cleansing — inconsistencies and missing values in the data are resolved.
- Standardization — formatting rule are applied to the data set.
- Deduplication — redundant data is excluded or discarded.
- Verification — unusable data is removed, and anomalies are flagged.
- Sorting — data is organized according to type.
- Other tasks — any additional/optional rules can be applied to improve data quality.

Transformation is generally considered to be the most important part of the ETL process. Data transformation improves data integrity and helps ensure that data arrives at its new destination fully compatible and ready to use.

The dataset extracted was a CSV (comma separated values) file and it was not in a ready to use form, the dataset was in form of plaintext, not readily readable as columns.

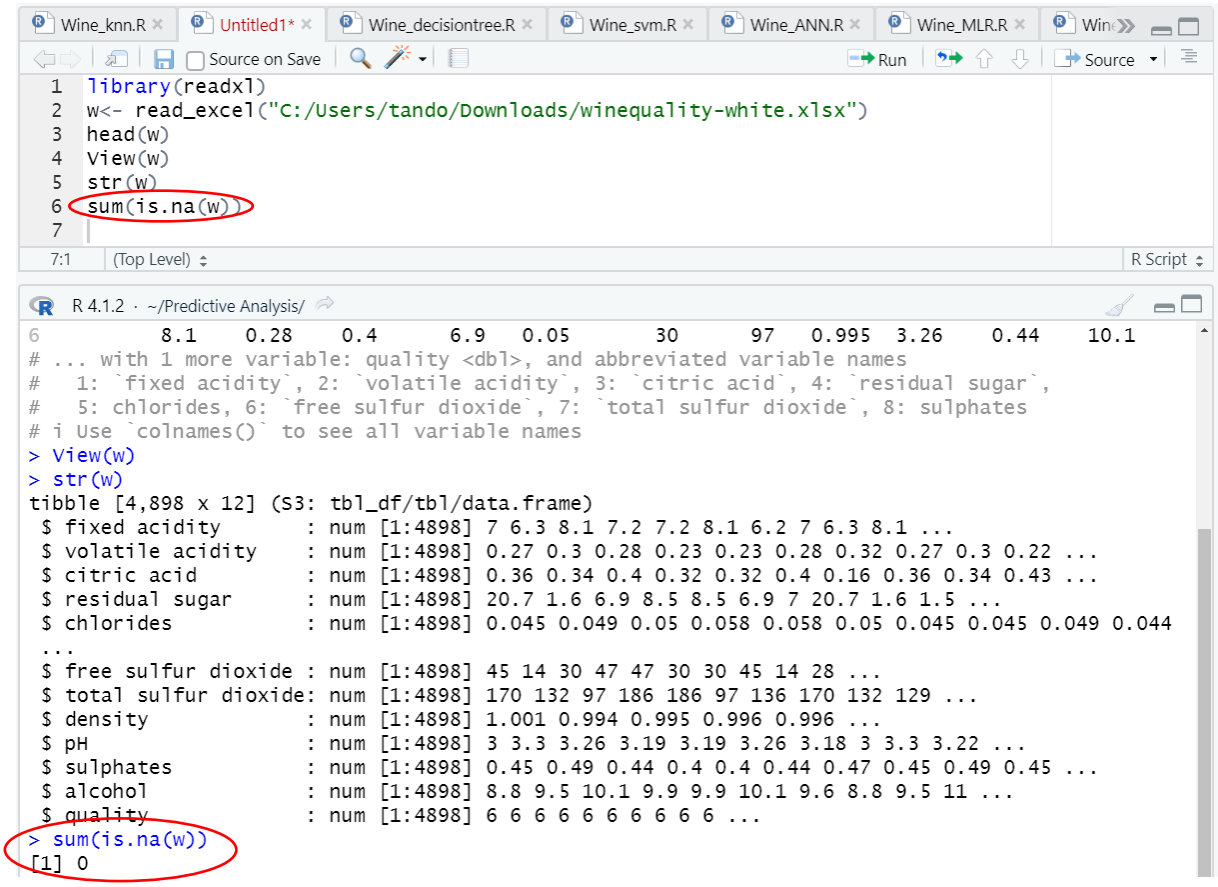


I converted this CSV file into an Excel file (.xlxs) using online converter.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	
2	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6	
3	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6	
4	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6	
5	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6	
6	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6	
7	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6	
8	6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6	
9	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6	
10	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6	
11	8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6	
12	8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5	
13	8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5	
14	7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5	
15	6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7	
16	8.3	0.42	0.62	19.25	0.04	41	172	1.0002	2.98	0.67	9.7	5	
17	6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7	
18	6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6	
19	6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8	
20	7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6	
21	6.5	0.31	0.14	7.5	0.044	34	133	0.9955	3.22	0.5	9.5	5	
22	6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8	
23	6.4	0.31	0.38	2.9	0.038	19	102	0.9912	3.17	0.35	11	7	
24	6.8	0.26	0.42	1.7	0.049	41	122	0.993	3.47	0.48	10.5	8	
25	7.6	0.67	0.14	1.5	0.074	25	168	0.9937	3.05	0.51	9.3	5	
26	6.6	0.27	0.41	1.3	0.052	16	142	0.9951	3.42	0.47	10	6	
27	7	0.25	0.32	9	0.046	56	245	0.9955	3.25	0.5	10.4	6	
28	6.9	0.24	0.35	1	0.052	35	146	0.993	3.45	0.44	10	6	

Worksheet

In this White Wine Quality dataset, the dataset was not having any null or missing values.



Formatting was standardized throughout the worksheet. Also, there were no duplicate values. Some columns are changed into factor datatype, which is

according to the need of the model and the details are mentioned ahead with the respective models.

- **Loading**

The final step in the ETL process is to load the newly transformed data into a new destination. Data that has been extracted to a staging area and transformed is loaded into your data warehouse.
Depending upon your business needs, data can be loaded in batches or all at once. Data can be loaded all at once (full load) or at scheduled intervals (incremental load). The exact nature of the loading will depend upon the data source, ETL tools, and various other factors.

Here data was not to be loaded in any data warehouse, it was simply transformed with all the required modifications and saved in work area/staging area in RStudio.

ANALYSIS ON DATASET

Introduction about the Dataset

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine.

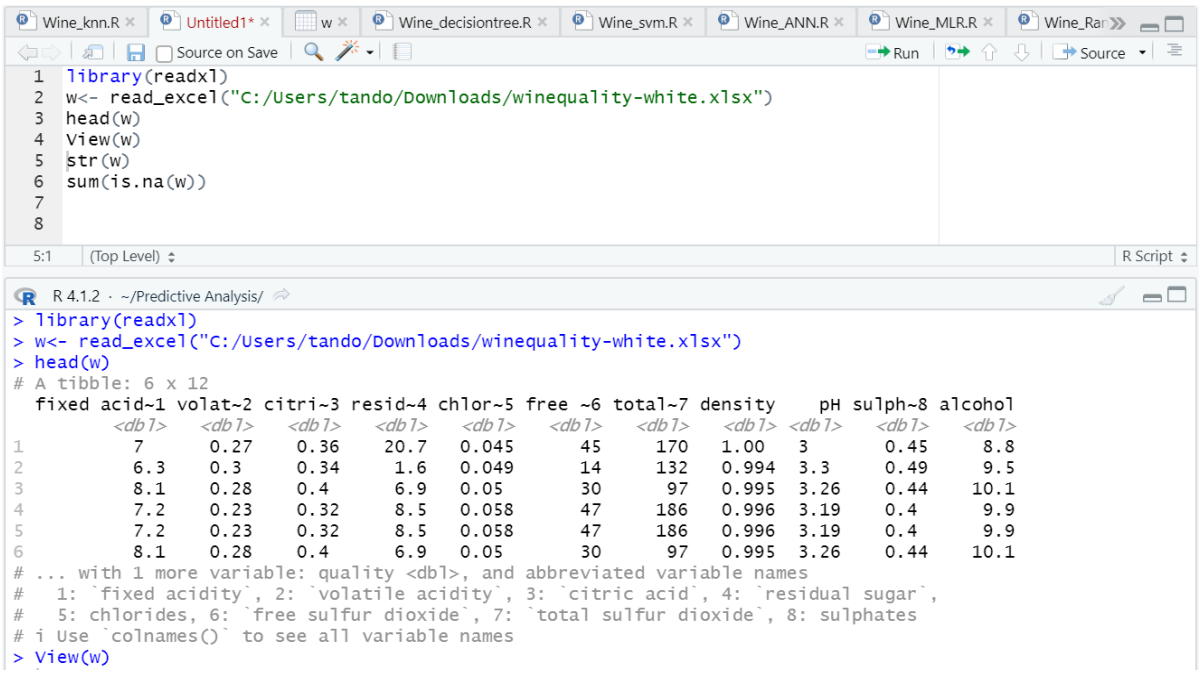
From these I have selected White Wine Quality dataset for the completion of this project.

In this dataset there are a total of 12 columns and 4898 rows.

The different columns are:

- fixed acidity
- volatile acidity
- citric acid
- residual sugar chlorides
- free sulfur dioxide
- total sulfur dioxide
- density pH
- sulphates
- alcohol
- quality

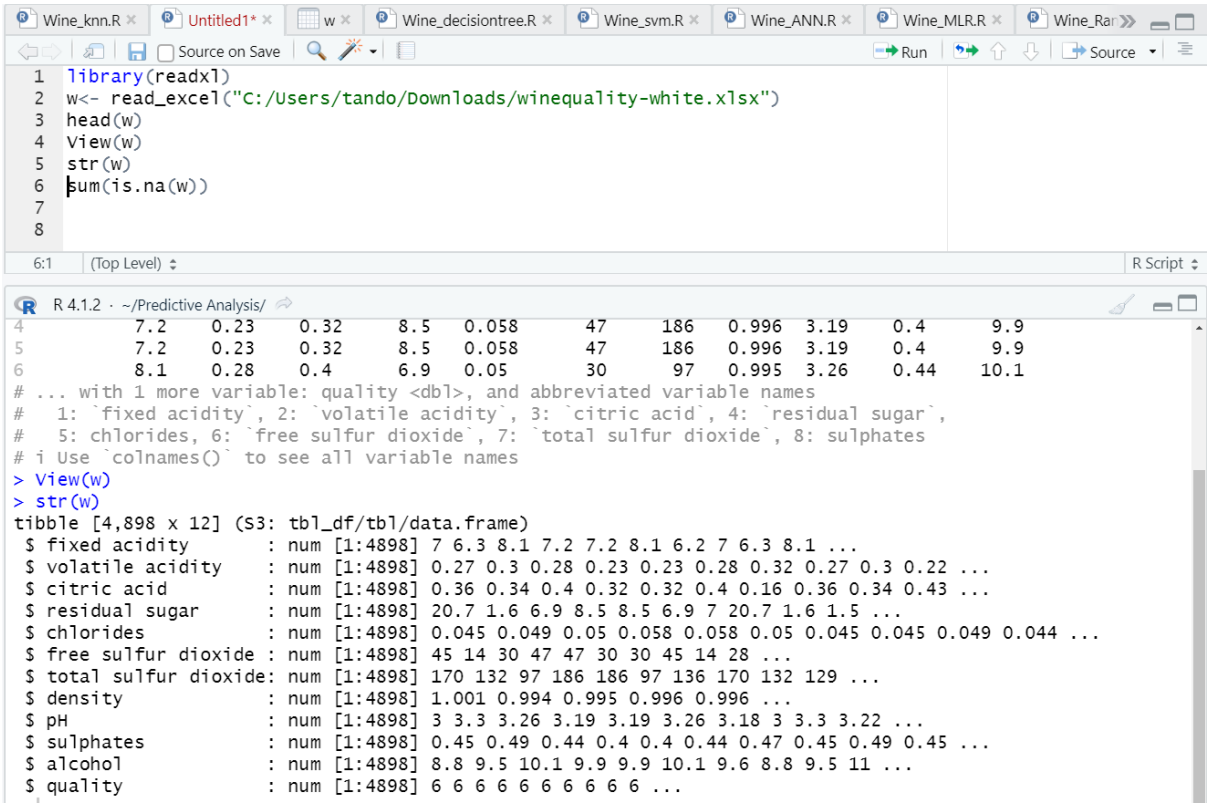
The dataset in R can be viewed using the View function:



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
2	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
3	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
4	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
6	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
7	6.2	0.320	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6
8	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
9	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
10	8.1	0.220	0.43	1.50	0.044	28.0	129.0	0.9938	3.22	0.45	11.0	6
11	8.1	0.270	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56	12.0	5
12	8.6	0.230	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53	9.7	5
13	7.9	0.180	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63	10.8	5
14	6.6	0.160	0.40	1.50	0.044	48.0	143.0	0.9912	3.54	0.52	12.4	7
15	8.3	0.420	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	9.7	5
16	6.6	0.170	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55	11.4	7
17	6.3	0.480	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36	9.6	6
18	6.2	0.660	0.48	1.20	0.029	29.0	75.0	0.9892	3.33	0.39	12.8	8
19	7.4	0.340	0.42	1.10	0.033	17.0	171.0	0.9917	3.12	0.53	11.3	6

Showing 1 to 19 of 4,898 entries, 12 total columns

Structure of the dataset:



All the columns are having “num” datatype.

OBJECTIVE DETAILS

Objective 1: Application of KNN model.

KNN Model:

Nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of similar labeled examples.

The nearest neighbor's approach to classification is exemplified by the k-nearest neighbors' algorithm (k-NN).

The k-NN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples.

The letter k is a variable term implying that any number of nearest neighbors could be used.

The Strengths and Weaknesses of this algorithm are as follows:

Strengths	Weaknesses
• Simple and effective • Makes no assumptions about the underlying data distribution • Fast training phase	• Does not produce a model, limiting the ability to understand how the features are related to the class • Requires selection of an appropriate *k* • Slow classification phase • Nominal features and missing data require additional processing

K-NN is a lazy learner. A lazy learner is not really learning anything. An eager learner has a model fitting or training step. A lazy learner does not have a training phase. There is very less training time in K-NN. This allows the training phase, which is not actually training anything, to occur very rapidly.

Application of KNN Model:

- **Libraries used:**
  - readxl: To import/load the dataset into R we used the “readxl” library.
  - caTools: It was used for sample splitting.
  - class: It was used for KNN classifier.
  - gmodels: It was used for implementing the CrossTable.
  - caret: It was used for extracting Confusion Matrix with Accuracy.

- **Steps involved in the KNN Model:**
  - Using the readxl library the dataset was imported.
  - Then data exploration was carried out.
  - We checked for the null values and in my dataset, there were none.
  - And then “quality” column was converted into factor for classification.
  - There are 7 distinct levels/classes.
  - Using the table function we determined the frequency of the different levels present in quality column.
  - Data was then then normalized/scaled as there was a significant difference in the range of the values.
  - The data was then split into Training and Testing data. Also labels for the same are also stored into train test label variables.
  - Then the KNN model was applied on the training dataset, keeping the value of k as 62. The value of k was selected as the square root of the no. of rows of the data fed to the model, which was training data having 3918 observations.
  - The model then made predictions based on the data with respect to the class/labels.
  - Finally, a CrossTable and Confusion Matrix with Accuracy were calculated.

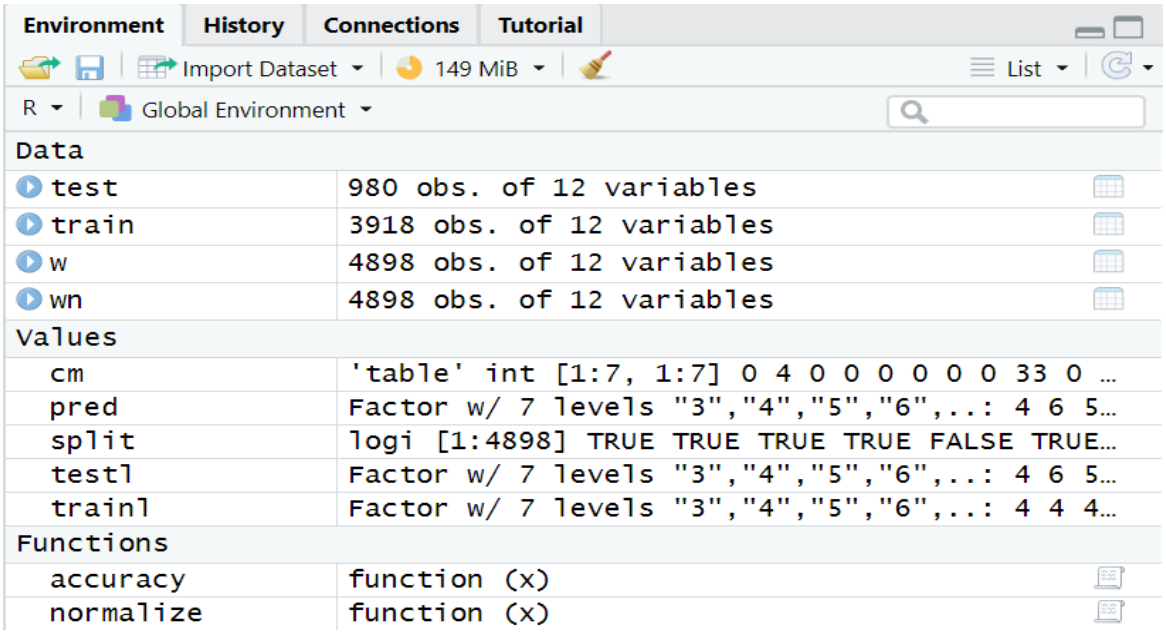
Code for KNN on winequality-white dataset:

Wine_knn.R | Wine_decisiontree.R | Wine_svm.R | Wine_ANN.R | Wine_MLR.R | Wine_F

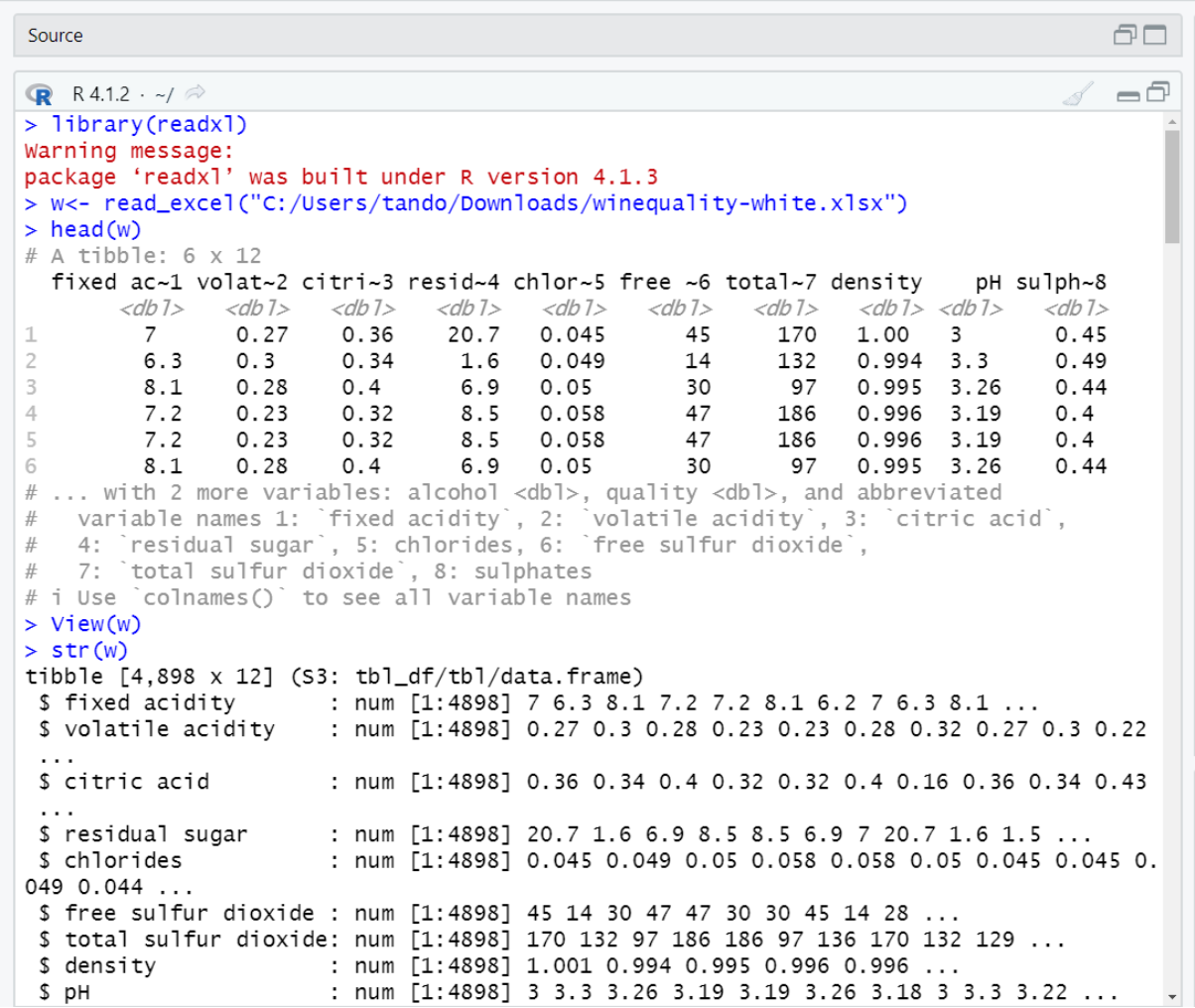
Source on Save | Run | Source

```
1  library(readxl)
2  w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
3  head(w)
4  View(w)
5  str(w)
6  sum(is.na(w))
7
8  w$quality <- factor(w$quality)
9  str(w)
10 table(w$quality)
11
12 normalize <- function(x){
13   return((x-min(x))/(max(x)-min(x)))
14 }
15 wn <- as.data.frame(lapply(w[-12], normalize))
16 head(wn)
17
18 wn$quality <- w$quality
19 View(wn)
20 str(wn)
21
22 library("caTools")
23 set.seed(1234)
24
25 split <- sample.split(wn$quality, SplitRatio=0.8)
26
27 train <- subset(wn, split==TRUE)
28 test <- subset(wn, split==FALSE)
29
30 trainl <- subset(wn$quality, split==TRUE)
31 testl <- subset(wn$quality, split==FALSE)
32
33 library(class)
34
35 pred <- knn(train = train, test = test, cl = trainl, k = 62)
36 pred
37
38 table(pred)
39
40 cm <- table(pred, test$quality)
41 cm
42
43 library("gmodels")
44 CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)
45
46 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
47 accuracy(cm)
48
49 install.packages("caret")
50 library("caret")
51 confusionMatrix(pred,test$quality)
52
```

Environment Variables:



Console:



```
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
> w$quality <- factor(w$quality)
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22
 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43
 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.
049 0.044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> table(w$quality)

   3    4    5    6    7    8    9
  20  163 1457 2198  880  175    5
> normalize <- function(x){
+   return((x-min(x))/(max(x)-min(x)))
+ }
> wn <- as.data.frame(lapply(w[-12], normalize))
> head(wn)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1     0.3076923        0.1862745   0.2168675     0.30828221 0.1068249
2     0.2403846        0.2156863   0.2048193     0.01533742 0.1186944
3     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
4     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
5     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
6     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
  free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
1          0.14982578            0.3735499 0.2677848 0.2545455 0.2674419
2          0.04181185            0.2853828 0.1328321 0.5272727 0.3139535
3          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
4          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
5          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
6          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
    alcohol
1 0.1290323
2 0.2419355
3 0.3387097
4 0.3064516
5 0.3064516
6 0.3387097
> wn$quality <- w$quality
> View(wn)
> str(wn)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  0.308 0.24 0.413 0.327 0.327 ...
 $ volatile.acidity    : num  0.186 0.216 0.196 0.147 0.147 ...
 $ citric.acid         : num  0.217 0.205 0.241 0.193 0.193 ...
 $ residual.sugar      : num  0.3083 0.0153 0.0966 0.1212 0.1212 ...
 $ chlorides           : num  0.107 0.119 0.122 0.145 0.145 ...
 $ free.sulfur.dioxide : num  0.1498 0.0418 0.0976 0.1568 0.1568 ...
 $ total.sulfur.dioxide: num  0.374 0.285 0.204 0.411 0.411 ...
 $ density             : num  0.268 0.133 0.154 0.164 0.164 ...
```

```
 $ pH                  : num  0.255 0.527 0.491 0.427 0.427 ...
 $ sulphates           : num  0.267 0.314 0.256 0.209 0.209 ...
 $ alcohol             : num  0.129 0.242 0.339 0.306 0.306 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> library("caTools")
Warning message:
package 'caTools' was built under R version 4.1.3
> set.seed(1234)
> split <- sample.split(wn$quality, SplitRatio=0.8)
> train <- subset(wn, split==TRUE)
> test <- subset(wn, split==FALSE)
> trainl <- subset(wn$quality, split==TRUE)
> testl <- subset(wn$quality, split==FALSE)
> library(class)
Warning message:
package 'class' was built under R version 4.1.3
> pred <- knn(train = train, test = test, cl = trainl, k = 62)
> pred
  [1] 6 8 7 6 6 7 6 6 6 6 6 7 5 5 5 5 5 5 5 5 6 6 7 5 7 7 5 6 7 5 6 4 6 5 5 5 5 4
 [39] 6 5 5 5 5 6 7 4 5 5 5 6 5 6 5 6 6 6 6 6 5 6 6 7 6 5 6 5 6 6 7 6 6 6 5 7 5 7
 [77] 7 6 5 5 5 5 6 6 5 5 7 6 6 6 6 7 5 5 7 6 5 6 5 6 6 5 4 6 7 6 6 5 5 6 4 6 6 6
[115] 6 7 5 7 6 7 5 5 5 6 6 5 7 5 5 4 6 5 5 5 4 7 7 6 5 6 8 7 5 5 5 6 6 5 5 6 5 8
[153] 5 6 6 6 5 7 6 6 6 7 7 5 8 5 6 7 6 6 5 6 5 6 6 7 8 7 8 5 5 6 6 5 5 6 7 6 6 7
[191] 8 5 7 7 7 5 6 6 7 5 6 4 6 6 6 7 6 5 6 7 4 6 5 5 5 6 4 6 7 6 7 6 6 8 7 8 6 6
[229] 7 5 5 7 6 5 5 5 5 5 6 5 6 5 5 5 7 6 8 8 6 6 6 5 7 5 4 6 5 6 6 6 7 5 5 6 6 5
[267] 7 6 5 5 6 6 6 6 7 6 6 6 6 6 6 7 4 6 8 4 7 6 6 6 6 7 6 7 7 6 8 6 4 6 6 6 6 5
[305] 6 7 5 7 6 6 6 6 5 6 7 7 6 6 7 6 6 5 6 6 6 6 6 6 6 7 6 6 5 6 5 5 5 6 6 6 6 5
[343] 5 8 6 6 5 6 5 5 6 5 6 6 6 6 6 5 6 4 5 6 6 6 6 6 6 5 5 6 7 6 5 5 6 8 6 5 6 6 6
[381] 6 5 6 5 7 7 7 6 7 5 6 6 6 5 5 5 6 6 5 5 5 4 5 6 6 7 5 5 5 6 5 7 5 4 5 5 6 8
[419] 6 6 6 5 6 5 5 6 6 6 5 5 6 6 5 5 5 5 6 4 6 5 5 5 6 6 5 7 6 6 5 5 4 6 7 6 6 5
[457] 7 5 7 5 7 5 6 5 5 6 6 7 5 5 6 5 6 6 6 5 6 6 7 6 7 7 6 5 6 6 5 6 5 5 5 6 6 7
[495] 6 6 7 8 7 6 5 6 6 6 6 5 6 4 5 5 5 5 6 5 6 5 6 7 6 6 4 5 5 4 6 5 6 5 5 8 6 6
[533] 4 5 5 5 5 7 5 7 5 6 6 7 4 5 5 7 5 6 7 5 4 7 7 6 6 6 5 5 6 5 5 6 5 6 6 7 6 7
[571] 8 5 6 6 5 8 8 5 5 5 6 5 7 7 5 6 8 5 7 6 7 5 6 7 5 7 5 6 8 6 6 5 5 5 6 6 7 6
[609] 6 6 4 6 5 6 6 5 5 6 6 7 7 7 6 7 6 6 6 6 6 6 7 6 6 7 5 5 5 8 7 7 6 6 7 7 6 6
[647] 5 6 6 7 7 6 6 6 8 6 5 6 6 6 6 6 5 6 5 6 5 8 6 7 7 5 5 6 7 7 4 7 6 6 6 7 6 7
[685] 6 6 6 7 7 6 7 6 6 8 6 6 5 4 6 6 6 6 6 5 6 6 6 6 8 7 7 6 7 7 6 6 6 5 6 6 4 6
[723] 6 6 6 7 4 5 7 7 6 7 7 6 5 7 7 7 6 7 5 5 5 5 7 5 6 7 7 7 7 8 5 7 6 6 7 5 6 7
[761] 6 7 5 5 7 6 7 8 5 5 7 6 5 5 5 6 6 6 6 6 6 6 6 6 6 6 7 6 5 6 5 7 7 7 5 5 5 7
[799] 6 5 6 7 7 7 7 5 5 6 5 5 4 6 7 6 8 6 6 6 6 5 6 6 7 5 6 6 6 6 6 7 6 7 6 7 7 7
[837] 7 8 6 6 8 5 5 6 6 5 4 6 4 6 6 5 5 5 5 5 7 5 6 6 6 5 6 5 5 5 5 6 6 5 7 7 6 5
[875] 6 5 6 5 6 6 5 5 5 5 5 7 5 6 6 6 6 5 5 6 5 5 6 6 4 6 6 6 6 6 6 5 5 6 6 6 6 6
[913] 5 6 6 6 7 5 5 6 6 6 6 6 4 5 5 6 7 6 6 6 7 6 7 7 7 6 5 6 7 7 7 6 6 6 6 6 6 6
[951] 6 6 7 6 8 8 6 4 5 8 5 7 4 6 6 6 6 7 5 5 6 6 6 6 5 6 5 6 6 7
Levels: 3 4 5 6 7 8 9
> table(pred)
pred
  3   4   5   6   7   8   9
  0  37 291 440 176  36   0
> cm <- table(pred, test$quality)
> cm

pred   3   4   5   6   7   8   9
   3   0   0   0   0   0   0   0
   4   4  33   0   0   0   0   0
   5   0   0 291   0   0   0   0
   6   0   0   0 440   0   0   0
   7   0   0   0   0 176   0   0
   8   0   0   0   0   0  35   1
   9   0   0   0   0   0   0   0
> library("gmodels")
Warning message:
package 'gmodels' was built under R version 4.1.3
> CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)


   Cell Contents
```

```
> CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  980


             | pred
test$quality |         4 |         5 |         6 |         7 |         8 | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           3 |         4 |         0 |         0 |         0 |         0 |         4 |
             |     1.000 |     0.000 |     0.000 |     0.000 |     0.000 |     0.004 |
             |     0.108 |     0.000 |     0.000 |     0.000 |     0.000 |           |
             |     0.004 |     0.000 |     0.000 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           4 |        33 |         0 |         0 |         0 |         0 |        33 |
             |     1.000 |     0.000 |     0.000 |     0.000 |     0.000 |     0.034 |
             |     0.892 |     0.000 |     0.000 |     0.000 |     0.000 |           |
             |     0.034 |     0.000 |     0.000 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           5 |         0 |       291 |         0 |         0 |         0 |       291 |
             |     0.000 |     1.000 |     0.000 |     0.000 |     0.000 |     0.297 |
             |     0.000 |     1.000 |     0.000 |     0.000 |     0.000 |           |
             |     0.000 |     0.297 |     0.000 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           6 |         0 |         0 |       440 |         0 |         0 |       440 |
             |     0.000 |     0.000 |     1.000 |     0.000 |     0.000 |     0.449 |
             |     0.000 |     0.000 |     1.000 |     0.000 |     0.000 |           |
             |     0.000 |     0.000 |     0.449 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           7 |         0 |         0 |         0 |       176 |         0 |       176 |
             |     0.000 |     0.000 |     0.000 |     1.000 |     0.000 |     0.180 |
             |     0.000 |     0.000 |     0.000 |     1.000 |     0.000 |           |
             |     0.000 |     0.000 |     0.000 |     0.180 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

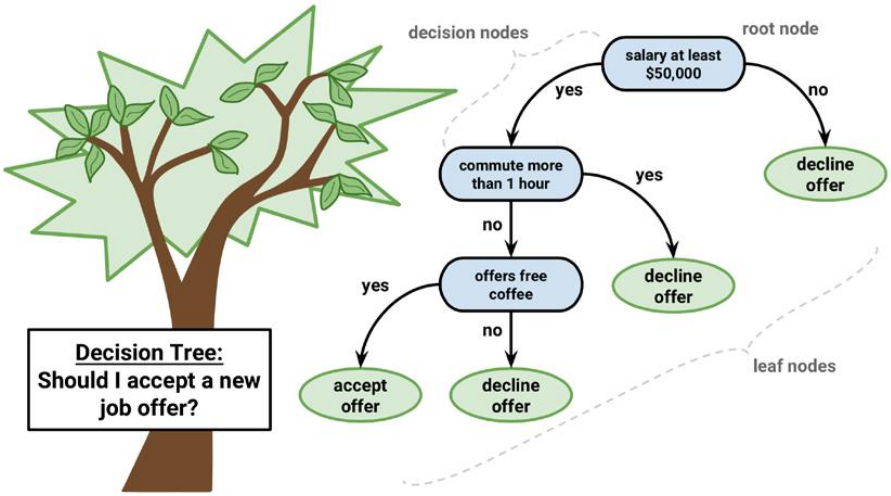
```
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           8 |         0 |         0 |         0 |         0 |        35 |        35 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     1.000 |     0.036 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.972 |           |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.036 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           9 |         0 |         0 |         0 |         0 |         1 |         1 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     1.000 |     0.001 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.028 |           |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.001 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
Column Total |        37 |       291 |       440 |       176 |        36 |       980 |
             |     0.038 |     0.297 |     0.449 |     0.180 |     0.037 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|


> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(cm)
[1] 99.4898
> library("caret")
Loading required package: ggplot2
Loading required package: lattice
> confusionMatrix(pred,test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8   9
         3   0   0   0   0   0   0   0
         4   4  33   0   0   0   0   0
         5   0   0 291   0   0   0   0
         6   0   0   0 440   0   0   0
         7   0   0   0   0 176   0   0
         8   0   0   0   0   0  35   1
         9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.9949
                 95% CI : (0.9881, 0.9983)
    No Information Rate : 0.449
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9924

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8 Class: 9
Sensitivity          0.000000  1.00000   1.0000    1.000   1.0000  1.00000  0.00000
Specificity          1.000000  0.99578   1.0000    1.000   1.0000  0.99894  1.00000
Pos Pred Value            NaN  0.89189   1.0000    1.000   1.0000  0.97222      NaN
Neg Pred Value       0.995918  1.00000   1.0000    1.000   1.0000  1.00000  0.99898
Prevalence           0.004082  0.03367   0.2969    0.449   0.1796  0.03571  0.00102
Detection Rate       0.000000  0.03367   0.2969    0.449   0.1796  0.03571  0.00000
Detection Prevalence 0.000000  0.03776   0.2969    0.449   0.1796  0.03673  0.00000
Balanced Accuracy    0.500000  0.99789   1.0000    1.000   1.0000  0.99947  0.50000
>
```

On applying KNN model to my dataset, I attained 99.49% accuracy (80:20 Split) which is very high, and this implies that KNN fits optimally on my dataset.

Objective 2: Application of Decision Tree.

Decision Tree:

Decision tree learners are powerful classifiers, which utilize a tree structure to model the relationships among the features and the potential outcomes. This structure earned its name due to the fact that it mirrors how a literal tree begins at a wide trunk, which if followed upward, splits into narrower and narrower branches. Similarly, a decision tree classifier uses a structure of branching decisions, which channel examples into a final predicted class value.



Decision trees are built using a heuristic called recursive partitioning. This approach is also commonly known as divide and conquer because it splits the data into subsets, which are then split repeatedly into even smaller subsets, and so on and so forth until the process stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.

The Strengths and Weaknesses of this algorithm are as follows:

Strengths	Weaknesses
• An all-purpose classifier that does well on most problems • Highly automatic learning process, which can handle numeric or nominal features, as well as missing data • Excludes unimportant features • Can be used on both small and large datasets • Results in a model that can be interpreted without a mathematical background (for relatively small trees) • More efficient than other complex models	• Decision tree models are often biased toward splits on features having a large number of levels • It is easy to overfit or underfit the model • Can have trouble modeling some relationships due to reliance on axis-parallel splits • Small changes in the training data can result in large changes to decision logic • Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

Application of Decision Tree Model:

- **Libraries used:**
  - readxl: To import/load the dataset into R we used the "readxl" library.
  - caTools: It was used for sample splitting.
  - rpart: It was used to implement our decision tree using Recursive PARTitioning.
  - rpart.plot: It is used to plot an rpart model.
  - gmodels: It was used for implementing the CrossTable.
  - caret: It was used for extracting Confusion Matrix with Accuracy.

- **Steps involved in the Decision Tree Model:**
  - Using the readxl library the dataset was imported.
  - Then data exploration was carried out.
  - We checked for the null values and in my dataset, there were none.
  - And then "quality" column was converted into factor for classification.
  - There are 7 distinct levels/classes.
  - Using the table function we determined the frequency of the different levels present in quality column.
  - Data was then then normalized/scaled as there was a significant difference in the range of the values.
  - The data was then split into Training and Testing data.
  - Then the Decision Tree model was applied on the training dataset
  - The model then made predictions based on the data with respect to the class/labels.
  - The quality column which is having classes was treated as the target variable and linked to other independent features such as fixed acidity, pH, alcohol, density etc.
  - Finally, a CrossTable and Confusion Matrix with Accuracy were calculated.

Code for Decision Tree on winequality-white dataset:

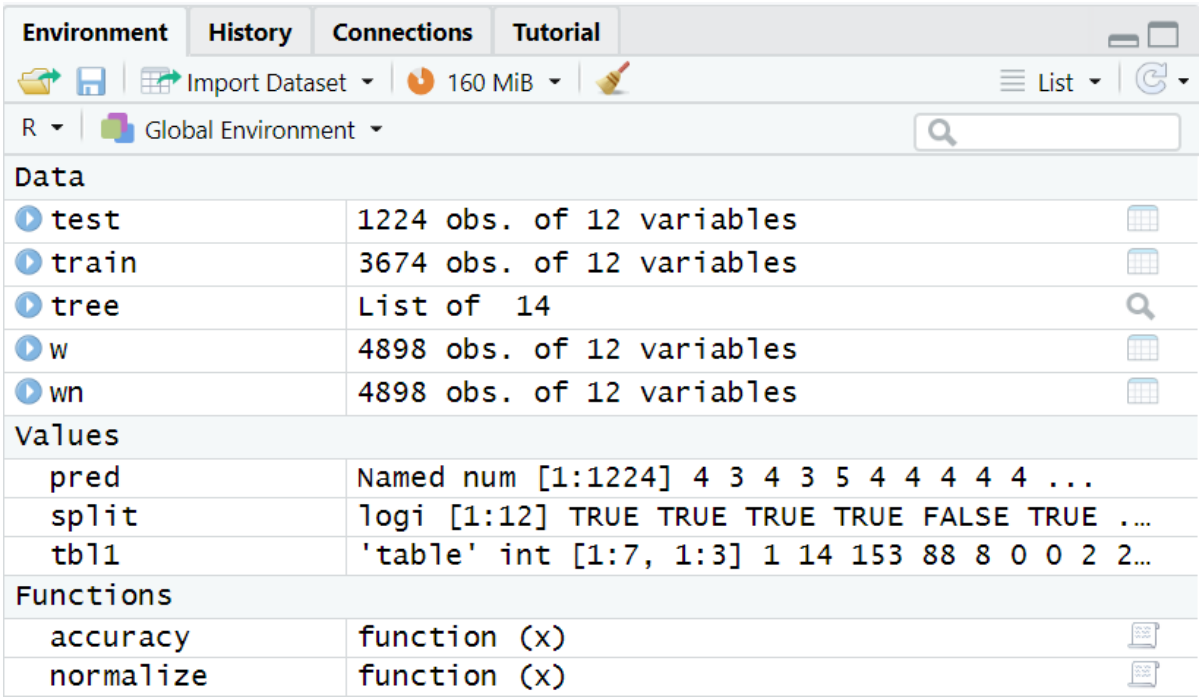
Wine_knn.R | Wine_decisiontree.R | Wine_svm.R | Wine_ANN.R | Wine_MLR.R | Wine_F

Source on Save | Run | Source

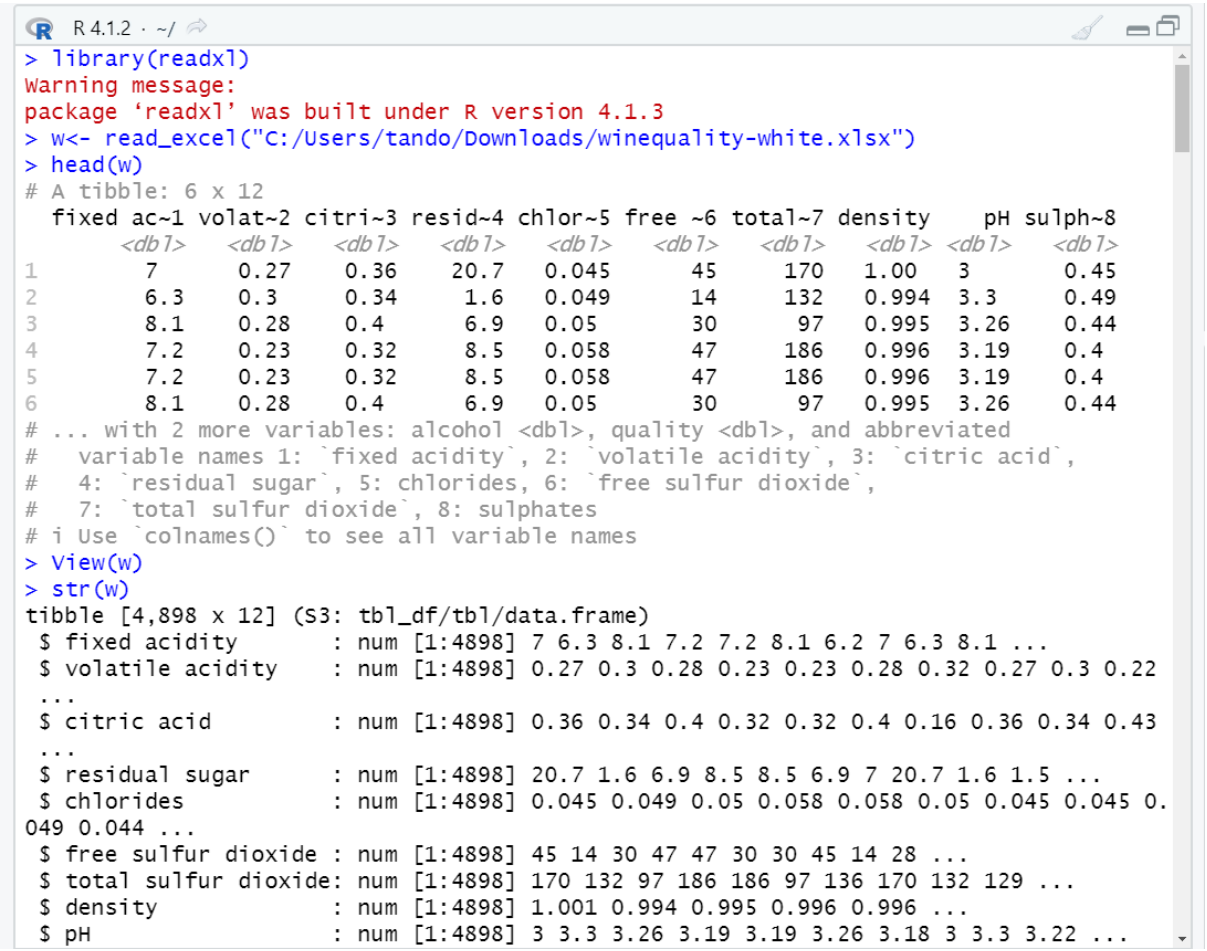
```
1  library(readxl)
2  w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
3  head(w)
4  View(w)
5  str(w)
6  sum(is.na(w))
7  #install.packages("rpart")
8  library("rpart")
9  w$quality <- factor(w$quality)#integer variable will be converted into factor
10 str(w)
11
12 normalize <- function(x){
13   return((x-min(x))/(max(x)-min(x)))
14 }
15 wn <- as.data.frame(lapply(w[-12], normalize))
16 head(wn)
17
18 wn$quality <- w$quality
19 View(wn)
20 str(wn)
21
22 #traning and testing data
23 library("caTools")
24 set.seed(1234)
25 split <- sample.split(wn, SplitRatio=0.8)
26
27 train <- subset(wn, split==TRUE)
28 test <- subset(wn, split==FALSE)
29
30 tree<- rpart(quality~.,train)
31 tree
32 #install.packages("rpart.plot")
33 library("rpart.plot")
34 rpart.plot(tree)
35 pred= predict(tree, test, type = "vector")
36 pred
37
38 library("gmodels")
39 CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)
40
41 tbl1 <- table(test$quality, pred)
42 tbl1
43
44 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
45 accuracy(tbl1)
46
47 #install.packages("caret")
48 library("caret")
49 confusionMatrix(as.factor(pred),test$quality)
50
```

1:1 (Top Level) R Script

Environment Variables:



Console:



```
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
> #install.packages("rpart")
> library("rpart")
> w$quality <- factor(w$quality)#integer variable will be converted into factor
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22
 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43
 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.
049 0.044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> normalize <- function(x){
+   return((x-min(x))/(max(x)-min(x)))
+ }
> wn <- as.data.frame(lapply(w[-12], normalize))
> head(wn)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1     0.3076923        0.1862745   0.2168675     0.30828221 0.1068249
2     0.2403846        0.2156863   0.2048193     0.01533742 0.1186944
3     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
4     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
5     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
6     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
  free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
1          0.14982578            0.3735499 0.2677848 0.2545455 0.2674419
2          0.04181185            0.2853828 0.1328321 0.5272727 0.3139535
3          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
4          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
5          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
6          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
    alcohol
1 0.1290323
2 0.2419355
3 0.3387097
4 0.3064516
5 0.3064516
6 0.3387097
> wn$quality <- w$quality
> View(wn)
> str(wn)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  0.308 0.24 0.413 0.327 0.327 ...
 $ volatile.acidity    : num  0.186 0.216 0.196 0.147 0.147 ...
 $ citric.acid         : num  0.217 0.205 0.241 0.193 0.193 ...
 $ residual.sugar      : num  0.3083 0.0153 0.0966 0.1212 0.1212 ...
 $ chlorides           : num  0.107 0.119 0.122 0.145 0.145 ...
 $ free.sulfur.dioxide : num  0.1498 0.0418 0.0976 0.1568 0.1568 ...
 $ total.sulfur.dioxide: num  0.374 0.285 0.204 0.411 0.411 ...
 $ density             : num  0.268 0.133 0.154 0.164 0.164 ...
 $ pH                  : num  0.255 0.527 0.491 0.427 0.427 ...
 $ sulphates           : num  0.267 0.314 0.256 0.209 0.209 ...
```

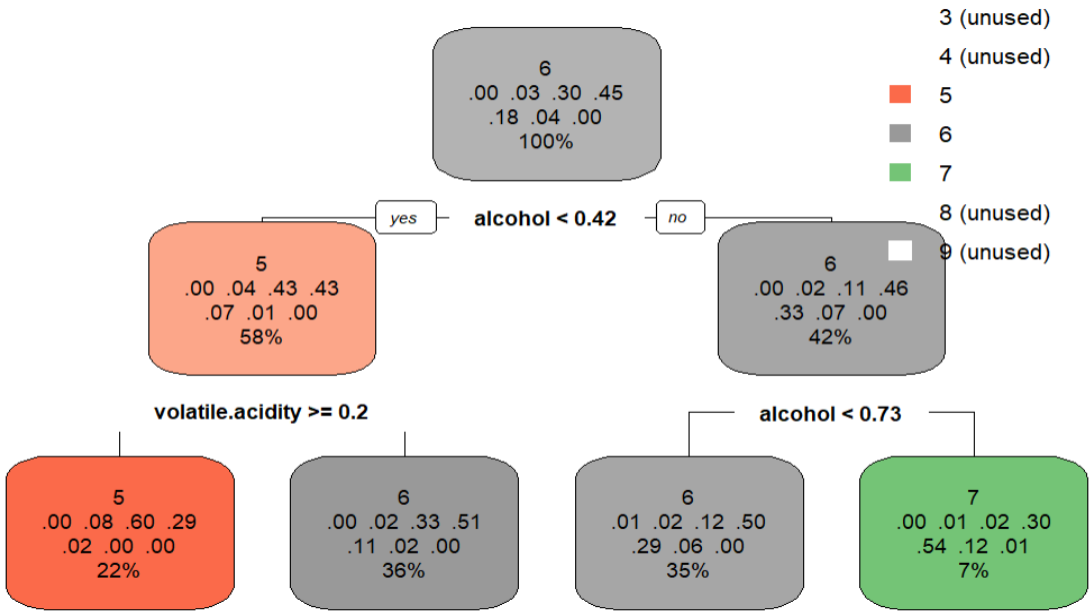
```
 $ alcohol             : num  0.129 0.242 0.339 0.306 0.306 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> #traning and testing data
> library("caTools")
Warning message:
package 'caTools' was built under R version 4.1.3
> set.seed(1234)
> split <- sample.split(wn, SplitRatio=0.8)
> train <- subset(wn, split==TRUE)
> test <- subset(wn, split==FALSE)
> tree<- rpart(quality~.,train)
> tree
n= 3674

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 3674 2038 6 (0.0044 0.033 0.3 0.45 0.18 0.036 0.0011)
  2) alcohol< 0.4233871 2134 1207 5 (0.0042 0.045 0.43 0.43 0.074 0.011 0.00047)
    4) volatile.acidity>=0.2034314 794  314 5 (0.0038 0.078 0.6 0.29 0.019 0 0) *
    5) volatile.acidity< 0.2034314 1340  654 6 (0.0045 0.025 0.33 0.51 0.11 0.018 0.00
075) *
  3) alcohol>=0.4233871 1540  824 6 (0.0045 0.018 0.11 0.46 0.33 0.07 0.0019)
    6) alcohol< 0.733871 1276  638 6 (0.0055 0.02 0.12 0.5 0.29 0.059 0.00078) *
    7) alcohol>=0.733871 264  121 7 (0 0.0076 0.023 0.3 0.54 0.12 0.0076) *
> #install.packages("rpart.plot")
> library("rpart.plot")
Warning message:
package 'rpart.plot' was built under R version 4.1.3
> rpart.plot(tree)
> pred= predict(tree, test, type = "vector")
> pred
   5    9   11   17   21   23   29   33   35   41   45   47   53   57   59   65
   4    3    4    3    5    4    4    4    4    4    4    3    4    4    3    4
  69   71   77   81   83   89   93   95  101  105  107  113  117  119  125  129
   4    4    5    4    3    3    5    4    4    4    4    3    4    3    4    4
 131  137  141  143  149  153  155  161  165  167  173  177  179  185  189  191
   4    4    3    4    4    4    4    4    4    4    4    5    3    4    5    4
```

```
3909 3911 3917 3921 3923 3929 3933 3935 3941 3945 3947 3953 3957 3959 3965 3969
   4    5    5    4    5    3    4    4    3    4    4    4    4    4    3    3
3971 3977 3981 3983 3989 3993 3995 4001
   4    5    4    4    4    4    4    4
 [ reached getOption("max.print") -- omitted 224 entries ]
> library("gmodels")
Warning message:
package 'gmodels' was built under R version 4.1.3
> CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1224
```

```
              | pred
test$quality |         3 |         4 |         5 | Row Total |
-------------|-----------|-----------|-----------|-----------|
           3 |         1 |         2 |         1 |         4 |
             |     0.250 |     0.500 |     0.250 |     0.003 |
             |     0.004 |     0.002 |     0.011 |           |
             |     0.001 |     0.002 |     0.001 |           |
-------------|-----------|-----------|-----------|-----------|
           4 |        14 |        24 |         2 |        40 |
             |     0.350 |     0.600 |     0.050 |     0.033 |
             |     0.053 |     0.028 |     0.022 |           |
             |     0.011 |     0.020 |     0.002 |           |
-------------|-----------|-----------|-----------|-----------|
           5 |       153 |       210 |         3 |       366 |
             |     0.418 |     0.574 |     0.008 |     0.299 |
             |     0.580 |     0.242 |     0.033 |           |
             |     0.125 |     0.172 |     0.002 |           |
-------------|-----------|-----------|-----------|-----------|
           6 |        88 |       433 |        41 |       562 |
             |     0.157 |     0.770 |     0.073 |     0.459 |
             |     0.333 |     0.498 |     0.451 |           |
             |     0.072 |     0.354 |     0.033 |           |
-------------|-----------|-----------|-----------|-----------|
           7 |         8 |       168 |        32 |       208 |
             |     0.038 |     0.808 |     0.154 |     0.170 |
             |     0.030 |     0.193 |     0.352 |           |
             |     0.007 |     0.137 |     0.026 |           |
-------------|-----------|-----------|-----------|-----------|
           8 |         0 |        31 |        12 |        43 |
             |     0.000 |     0.721 |     0.279 |     0.035 |
             |     0.000 |     0.036 |     0.132 |           |
             |     0.000 |     0.025 |     0.010 |           |
-------------|-----------|-----------|-----------|-----------|
           9 |         0 |         1 |         0 |         1 |
             |     0.000 |     1.000 |     0.000 |     0.001 |
             |     0.000 |     0.001 |     0.000 |           |
             |     0.000 |     0.001 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|
Column Total |       264 |       869 |        91 |      1224 |
             |     0.216 |     0.710 |     0.074 |           |
-------------|-----------|-----------|-----------|-----------|


> tbl1 <- table(test$quality, pred)
> tbl1
   pred
      3   4   5
  3   1   2   1
  4  14  24   2
  5 153 210   3
  6  88 433  41
  7   8 168  32
  8   0  31  12
  9   0   1   0
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(tbl1)
[1] 2.287582
> #install.packages("caret")
> library("caret")
> confusionMatrix(as.factor(pred),test$quality)
Confusion Matrix and Statistics
```

```
          Reference
Prediction   3   4   5   6   7   8   9
         3   1  14 153  88   8   0   0
         4   2  24 210 433 168  31   1
         5   1   2   3  41  32  12   0
         6   0   0   0   0   0   0   0
         7   0   0   0   0   0   0   0
         8   0   0   0   0   0   0   0
         9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.0229
                 95% CI : (0.0153, 0.0329)
    No Information Rate : 0.4592
    P-Value [Acc > NIR] : 1

                  Kappa : -0.0244

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8 Class: 9
Sensitivity          0.250000  0.60000 0.008197   0.0000   0.0000  0.00000 0.000000
Specificity          0.784426  0.28632 0.897436   1.0000   1.0000  1.00000 1.000000
Pos Pred Value       0.003788  0.02762 0.032967      NaN      NaN      NaN      NaN
Neg Pred Value       0.996875  0.95493 0.679612   0.5408   0.8301  0.96487 0.999183
Prevalence           0.003268  0.03268 0.299020   0.4592   0.1699  0.03513 0.000817
Detection Rate       0.000817  0.01961 0.002451   0.0000   0.0000  0.00000 0.000000
Detection Prevalence 0.215686  0.70997 0.074346   0.0000   0.0000  0.00000 0.000000
Balanced Accuracy    0.517213  0.44316 0.452816   0.5000   0.5000  0.50000 0.500000
```

Plots:



On applying Decision tree model to my dataset, I attained 2.29% (80:20 Split) accuracy which is very low and this implies that decision tree model is not the right fit for my dataset.

This much less accuracy is due to oversampling and undersampling of the classes. As we can see through the plot that in our sample 4 classes (3,4,8,9) were unused and majority of the classification is done from prominent classes only.

Objective 3: Application of SVM Model.

SVM Model:

In machine learning, Support vector machine(SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that best differentiates the two classes.

A Support Vector Machine (SVM) can be imagined as a surface that creates a boundary between points of data plotted in multidimensional that represent examples and their feature values. The goal of a SVM is to create a flat boundary called a hyperplane, which divides the space to create fairly homogeneous partitions on either side. In this way, the SVM learning combines aspects of both the instance-based nearest neighbor learning and the linear regression modeling.

Application of SVM Model:

- **Libraries used:**
    - readxl: To import/load the dataset into R we used the “readxl” library.
    - caTools: It was used for sample splitting.
    - e1071 : Used for SVM classifier and Plotting SVM.
    - gmodels: It was used for implementing the CrossTable.
    - caret: It was used for extracting Confusion Matrix with Accuracy.

- **Steps involved in the SVM Model:**
    - Using the readxl library the dataset was imported.
    - Then data exploration was carried out.
    - We checked for the null values and in my dataset, there were none.
    - And then “quality” column was converted into factor for classification.
    - There are 7 distinct levels/classes.
    - Using the table function we determined the frequency of the different levels present in quality column.

- Data was then then normalized/scaled as there was a significant difference in the range of the values.
- The data was then split into Training and Testing data.
- Then the SVM model was applied on the training dataset.
- The model then made predictions based on the data with respect to the class/labels.
- The quality column which is having classes was treated as the target variable and linked to other independent features such as fixed acidity, pH, alcohol, density etc.
- Finally, a CrossTable and Confusion Matrix with Accuracy were calculated.

Code:

Wine_decisiontree.R | Wine_svm.R | Wine_ANN.R | Wine_MLR.R | Wine_RandomForest.R

Source on Save | Run | Source

```
1  library(readxl)
2  w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
3  head(w)
4  View(w)
5  str(w)
6  sum(is.na(w))
7
8  w$quality <- factor(w$quality)
9  str(w)
10 table(w$quality)
11
12 normalize <- function(x){
13   return((x-min(x))/(max(x)-min(x)))
14 }
15 wn <- as.data.frame(lapply(w[-12], normalize))
16 head(wn)
17
18 wn$quality <- w$quality
19 View(wn)
20 str(wn)
21
22 library("caTools")
23 set.seed(1234)
24
25 split <- sample.split(wn$quality, SplitRatio=0.8)
26
27 train <- subset(wn, split==TRUE)
28 test <- subset(wn, split==FALSE)
29
30 library("e1071")
31 svm_model = svm(formula = quality~.,
32                 type = "C-classification",
33                 data = train,
34                 kernel = "linear")
35
36 summary(svm_model)
37
38 pred = predict(svm_model, test[-12])
39 pred
```

```
40
41 cm <- table(test$quality, pred)
42 cm
43
44 library("gmodels")
45 CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)
46
47 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
48 accuracy(cm)
49
50 library("caret")
51 confusionMatrix(pred,test$quality)
```

11:1 (Top Level) R Script

Environment Variables:

Environment | History | Connections | Tutorial

Import Dataset | 165 MiB | List

R | Global Environment

Data	
svm_model	Large svm.formula (30 elements, 1.9 MB)
test	980 obs. of 12 variables
train	3918 obs. of 12 variables
w	4898 obs. of 12 variables
wn	4898 obs. of 12 variables
Values	
cm	'table' int [1:7, 1:7] 0 0 0 0 0 0 0 0 0 0 0 ....
pred	Factor w/ 7 levels "3","4","5","6",..: 4 4 4...
split	logi [1:4898] TRUE TRUE TRUE TRUE FALSE TRUE...
Functions	
accuracy	function (x)
normalize	function (x)

Console:

R 4.1.2 · ~/

```
> library(readxl)
Warning message:
package 'readxl' was built under R version 4.1.3
> w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
> head(w)
# A tibble: 6 x 12
  fixed ac~1 volat~2 citri~3 resid~4 chlor~5 free ~6 total~7 density    pH sulph~8
       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
1        7      0.27    0.36    20.7   0.045      45     170   1.00   3       0.45
2        6.3    0.3     0.34     1.6   0.049      14     132   0.994  3.3     0.49
3        8.1    0.28    0.4      6.9   0.05       30      97   0.995  3.26    0.44
4        7.2    0.23    0.32     8.5   0.058      47     186   0.996  3.19    0.4
5        7.2    0.23    0.32     8.5   0.058      47     186   0.996  3.19    0.4
6        8.1    0.28    0.4      6.9   0.05       30      97   0.995  3.26    0.44
# ... with 2 more variables: alcohol <dbl>, quality <dbl>, and abbreviated
#   variable names 1: `fixed acidity`, 2: `volatile acidity`, 3: `citric acid`,
```

```
#   4: `residual sugar`, 5: chlorides, 6: `free sulfur dioxide`,
#   7: `total sulfur dioxide`, 8: sulphates
# i Use `colnames()` to see all variable names
> View(w)
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22
 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43
 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.
049 0.044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
> w$quality <- factor(w$quality)
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22
 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43
 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.
049 0.044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> table(w$quality)

   3    4    5    6    7    8    9
  20  163 1457 2198  880  175    5
> normalize <- function(x){
+   return((x-min(x))/(max(x)-min(x)))
+ }
> wn <- as.data.frame(lapply(w[-12], normalize))
> head(wn)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1     0.3076923        0.1862745   0.2168675     0.30828221 0.1068249
2     0.2403846        0.2156863   0.2048193     0.01533742 0.1186944
3     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
4     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
5     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
6     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
  free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
1          0.14982578            0.3735499 0.2677848 0.2545455 0.2674419
2          0.04181185            0.2853828 0.1328321 0.5272727 0.3139535
3          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
4          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
5          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
6          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
```

```
    alcohol
1 0.1290323
2 0.2419355
3 0.3387097
4 0.3064516
5 0.3064516
6 0.3387097
> wn$quality <- w$quality
> View(wn)
> str(wn)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  0.308 0.24 0.413 0.327 0.327 ...
 $ volatile.acidity    : num  0.186 0.216 0.196 0.147 0.147 ...
 $ citric.acid         : num  0.217 0.205 0.241 0.193 0.193 ...
 $ residual.sugar      : num  0.3083 0.0153 0.0966 0.1212 0.1212 ...
 $ chlorides           : num  0.107 0.119 0.122 0.145 0.145 ...
 $ free.sulfur.dioxide : num  0.1498 0.0418 0.0976 0.1568 0.1568 ...
 $ total.sulfur.dioxide: num  0.374 0.285 0.204 0.411 0.411 ...
 $ density             : num  0.268 0.133 0.154 0.164 0.164 ...
 $ pH                  : num  0.255 0.527 0.491 0.427 0.427 ...
 $ sulphates           : num  0.267 0.314 0.256 0.209 0.209 ...
 $ alcohol             : num  0.129 0.242 0.339 0.306 0.306 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4
 ...
> library("caTools")
Warning message:
package 'caTools' was built under R version 4.1.3
> set.seed(1234)
> split <- sample.split(wn$quality, SplitRatio=0.8)
> train <- subset(wn, split==TRUE)
> test <- subset(wn, split==FALSE)
> library("e1071")
Warning message:
package 'e1071' was built under R version 4.1.3
> svm_model = svm(formula = quality~.,
+                 type = "C-classification",
+                 data = train,
+                 kernel = "linear")
> summary(svm_model)

Call:
svm(formula = quality ~ ., data = train, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  3648

 ( 1580 1074 704 140 130 16 4 )


Number of Classes:  7

Levels:
 3 4 5 6 7 8 9



> pred = predict(svm_model, test[-12])
> pred
   5   18   22   26   28   30   44   49   51   64   65   67   73   85  104  107
   6    6    6    6    6    6    5    5    6    5    6    6    6    5    5    6
 113  119  120  122  124  125  129  131  132  139  141  152  161  175  176  177
   5    5    5    5    5    6    6    6    6    6    5    6    6    6    6    6
 191  192  194  198  201  205  210  216  217  220  221  222  248  252  253  262
   5    6    6    5    6    6    5    6    6    6    6    5    6    6    5    6
```

```
- - -

4714 4717 4723 4727 4729 4736 4748 4756 4757 4761 4766 4767 4771 4780 4785 4788
   6    6    5    6    6    6    6    6    6    6    6    6    6    5    6    6
4795 4801 4805 4815 4819 4822 4826 4835 4837 4852 4857 4870 4875 4884 4885 4886
   5    6    6    6    6    6    6    6    5    5    5    6    6    6    5    5
4889 4890 4891 4897
   5    6    6    6
Levels: 3 4 5 6 7 8 9
> cm <- table(test$quality, pred)
> cm
   pred
      3   4   5   6   7   8   9
  3   0   0   1   3   0   0   0
  4   0   0  25   8   0   0   0
  5   0   0 156 135   0   0   0
  6   0   0  78 362   0   0   0
  7   0   0   8 168   0   0   0
  8   0   0   0  35   0   0   0
  9   0   0   0   1   0   0   0
> library("gmodels")
Warning message:
package 'gmodels' was built under R version 4.1.3
> CrossTable(x = test$quality, y = pred, prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  980


             | pred
test$quality |         5 |         6 | Row Total |
-------------|-----------|-----------|-----------|
           3 |         1 |         3 |         4 |
             |     0.250 |     0.750 |     0.004 |
             |     0.004 |     0.004 |           |
             |     0.001 |     0.003 |           |
-------------|-----------|-----------|-----------|
           4 |        25 |         8 |        33 |
             |     0.758 |     0.242 |     0.034 |
             |     0.093 |     0.011 |           |
             |     0.026 |     0.008 |           |
-------------|-----------|-----------|-----------|
           5 |       156 |       135 |       291 |
             |     0.536 |     0.464 |     0.297 |
             |     0.582 |     0.190 |           |
             |     0.159 |     0.138 |           |
-------------|-----------|-----------|-----------|
           6 |        78 |       362 |       440 |
             |     0.177 |     0.823 |     0.449 |
             |     0.291 |     0.508 |           |
             |     0.080 |     0.369 |           |
-------------|-----------|-----------|-----------|
           7 |         8 |       168 |       176 |
             |     0.045 |     0.955 |     0.180 |
             |     0.030 |     0.236 |           |
             |     0.008 |     0.171 |           |
-------------|-----------|-----------|-----------|
           8 |         0 |        35 |        35 |
             |     0.000 |     1.000 |     0.036 |
             |     0.000 |     0.049 |           |
             |     0.000 |     0.036 |           |
-------------|-----------|-----------|-----------|
```

```
           9 |         0 |         1 |         1 |
             |     0.000 |     1.000 |     0.001 |
             |     0.000 |     0.001 |           |
             |     0.000 |     0.001 |           |
-------------|-----------|-----------|-----------|
Column Total |       268 |       712 |       980 |
             |     0.273 |     0.727 |           |
-------------|-----------|-----------|-----------|

> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(cm)
[1] 52.85714
> library("caret")
> confusionMatrix(pred,test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8   9
         3   0   0   0   0   0   0   0
         4   0   0   0   0   0   0   0
         5   1  25 156  78   8   0   0
         6   3   8 135 362 168  35   1
         7   0   0   0   0   0   0   0
         8   0   0   0   0   0   0   0
         9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.5286
                 95% CI : (0.4968, 0.5602)
    No Information Rate : 0.449
    P-Value [Acc > NIR] : 3.498e-07

                  Kappa : 0.2045

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8 Class: 9
Sensitivity          0.000000  0.00000   0.5361   0.8227   0.0000  0.00000  0.00000
Specificity          1.000000  1.00000   0.8374   0.3519   1.0000  1.00000  1.00000
Pos Pred Value            NaN      NaN   0.5821   0.5084      NaN      NaN      NaN
Neg Pred Value       0.995918  0.96633   0.8104   0.7090   0.8204  0.96429  0.99898
Prevalence           0.004082  0.03367   0.2969   0.4490   0.1796  0.03571  0.00102
Detection Rate       0.000000  0.00000   0.1592   0.3694   0.0000  0.00000  0.00000
Detection Prevalence 0.000000  0.00000   0.2735   0.7265   0.0000  0.00000  0.00000
Balanced Accuracy    0.500000  0.50000   0.6868   0.5873   0.5000  0.50000  0.50000
>
```

On applying SVM model(linear) to my dataset, I attained 52.88% (80:20 Split) accuracy which is low, and this implies that SVM model is not the right fit for my dataset.

This much less accuracy is due to the fact that there are not very distinct levels/classes, and the classes will be very much randomized in the plot. It is not possible to find a linear hyperplane for them.

Objective 4: Application of ANN Model.

ANN Model:

An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, ANN uses a network of artificial neurons or nodes to solve learning problems.

The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons in brains.

ANNs are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output.

I have applied ANN Model on winequality-white dataset. In this **"quality"** column is used as the factor/classification variable, having 7 distinct levels/classes. I have applied ANN classification model to predict the **"quality"** of wine using various features like fixed acidity, pH, alcohol, density etc.

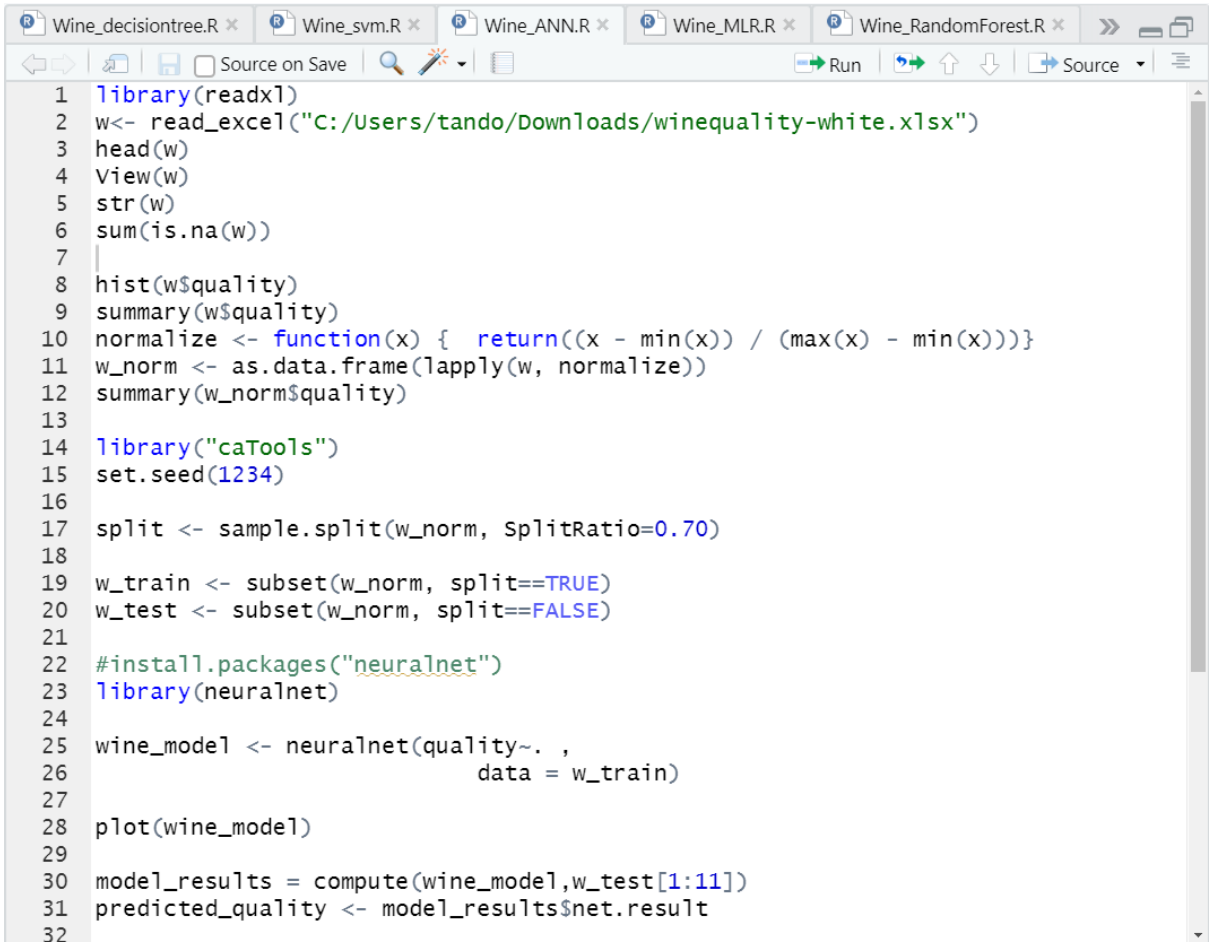
Application of ANN Model:

- **Libraries used:**
  - readxl: To import/load the dataset into R we used the "readxl" library.
  - caTools: It was used for sample splitting.
  - neuralnet: Neuralnet package is used for neural network implementation.

- **Steps involved in the ANN Model:**
  - Using the readxl library the dataset was imported.
  - Then data exploration was carried out.
  - We checked for the null values and in my dataset, there were none.
  - Data was then then normalized/scaled as there was a significant difference in the range of the values.
  - The data was then split into Training and Testing data.

- Then the ANN model was applied on the training dataset keeping hidden layers as 5.
- The model then made predictions based on the data with respect to the dependent variable i.e., quality.
- The quality column which is having classes was treated as the target variable and linked to other independent features such as fixed acidity, pH, alcohol, density etc.
- Then the plot for our model is created. In this model, there is one input node for each of the 11 features, followed by 5 hidden nodes and a single output node that predicts the quality.
- Then using the compute function on test dataset, which returns a list with two components: $neurons, which stores the neurons for each layer in the network, and $net.result, which stores the predicted values, we got the predictions.
- Finally, correlation between predicted quality and test quality were calculated.

Code:

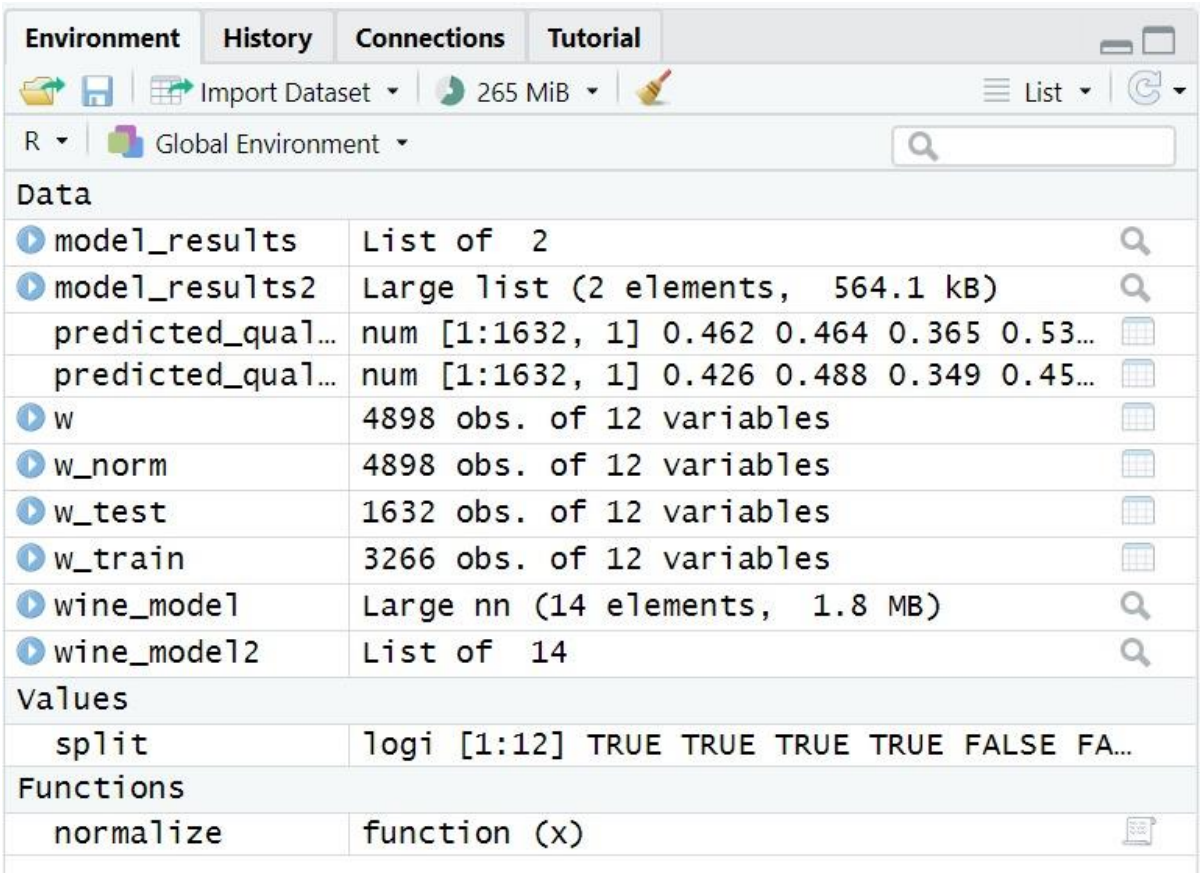



```
1  library(readxl)
2  w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
3  head(w)
4  View(w)
5  str(w)
6  sum(is.na(w))
7
8  hist(w$quality)
9  summary(w$quality)
10 normalize <- function(x) {  return((x - min(x)) / (max(x) - min(x)))}
11 w_norm <- as.data.frame(lapply(w, normalize))
12 summary(w_norm$quality)
13
14 library("caTools")
15 set.seed(1234)
16
17 split <- sample.split(w_norm, SplitRatio=0.70)
18
19 w_train <- subset(w_norm, split==TRUE)
20 w_test <- subset(w_norm, split==FALSE)
21
22 #install.packages("neuralnet")
23 library(neuralnet)
24
25 wine_model <- neuralnet(quality~. ,
26                         data = w_train)
27
28 plot(wine_model)
29
30 model_results = compute(wine_model,w_test[1:11])
31 predicted_quality <- model_results$net.result
32
```

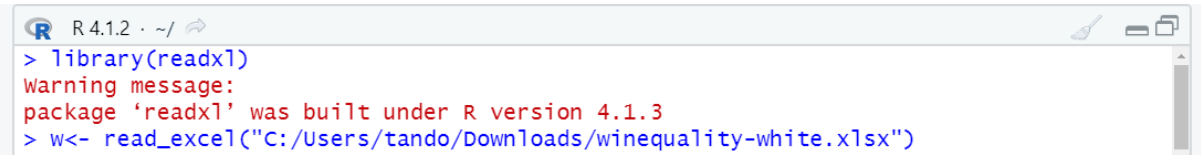
```
33 cor(predicted_quality, w_test$quality)
34
35
36
37 wine_model2 <- neuralnet(quality~. ,
38                          data = w_train, hidden = 5)
39
40 plot(wine_model2)
41
42 model_results2 <- compute(wine_model2, w_test[1:11])
43 predicted_quality2 <- model_results2$net.result
44
45 cor(predicted_quality2, w_test$quality)
46
```

7:1 (Top Level) R Script

Environment Variables:

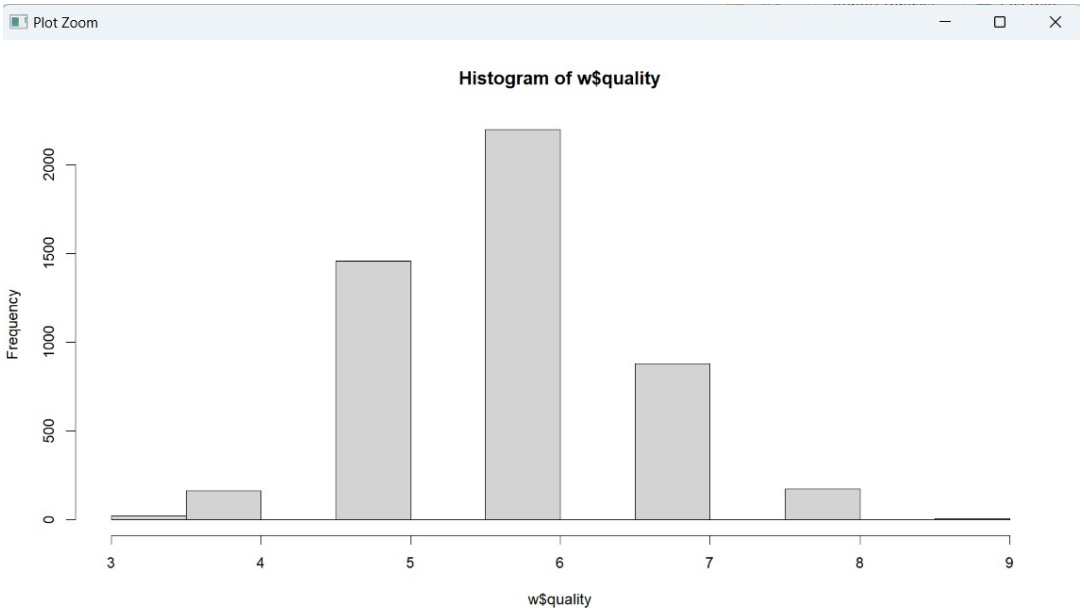


Console:

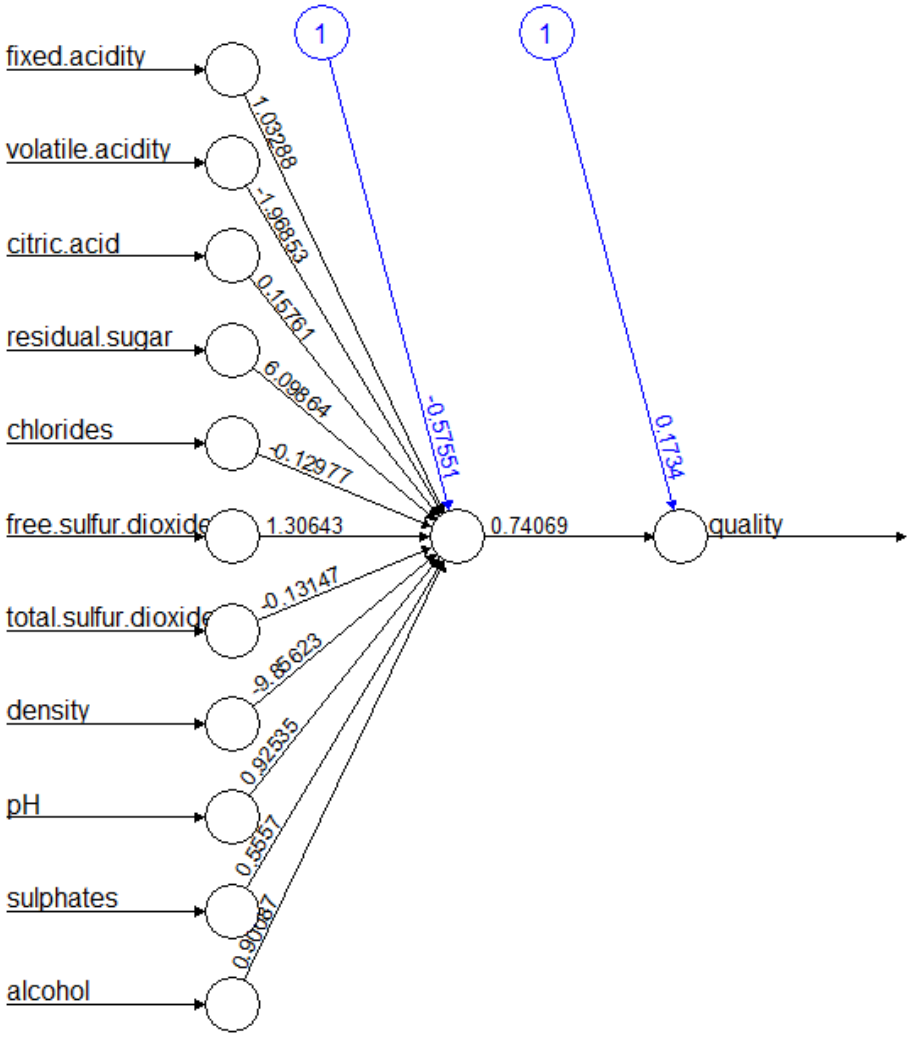


```
> head(w)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.0             0.27        0.36           20.7     0.045
2           6.3             0.30        0.34            1.6     0.049
3           8.1             0.28        0.40            6.9     0.050
4           7.2             0.23        0.32            8.5     0.058
5           7.2             0.23        0.32            8.5     0.058
6           8.1             0.28        0.40            6.9     0.050
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol quality
1                  45                  170  1.0010 3.00      0.45     8.8       6
2                  14                  132  0.9940 3.30      0.49     9.5       6
3                  30                   97  0.9951 3.26      0.44    10.1       6
4                  47                  186  0.9956 3.19      0.40     9.9       6
5                  47                  186  0.9956 3.19      0.40     9.9       6
6                  30                   97  0.9951 3.26      0.44    10.1       6
> View(w)
> str(w)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044
 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
> hist(w$quality)
> summary(w$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.878   6.000   9.000
> normalize <- function(x) {  return((x - min(x)) / (max(x) - min(x)))}
> w_norm <- as.data.frame(lapply(w, normalize))
> summary(w_norm$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.3333  0.5000  0.4797  0.5000  1.0000
> library("caTools")
> set.seed(1234)
> split <- sample.split(w_norm, SplitRatio=0.70)
> w_train <- subset(w_norm, split==TRUE)
> w_test <- subset(w_norm, split==FALSE)
> #install.packages("neuralnet")
> library(neuralnet)
> wine_model <- neuralnet(quality~. ,
+                         data = w_train)
> plot(wine_model)
> model_results = compute(wine_model,w_test[1:11])
> predicted_quality <- model_results$net.result
> cor(predicted_quality, w_test$quality)
          [,1]
[1,] 0.4828592
> wine_model2 <- neuralnet(quality~. ,
+                          data = w_train, hidden = 5)
Warning message:
Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
> wine_model2 <- neuralnet(quality~. ,
+                          data = w_train, hidden = 5)
> plot(wine_model2)
> model_results2 <- compute(wine_model2, w_test[1:11])
> predicted_quality2 <- model_results2$net.result
> cor(predicted_quality2, w_test$quality)
          [,1]
[1,] 0.5735015
```

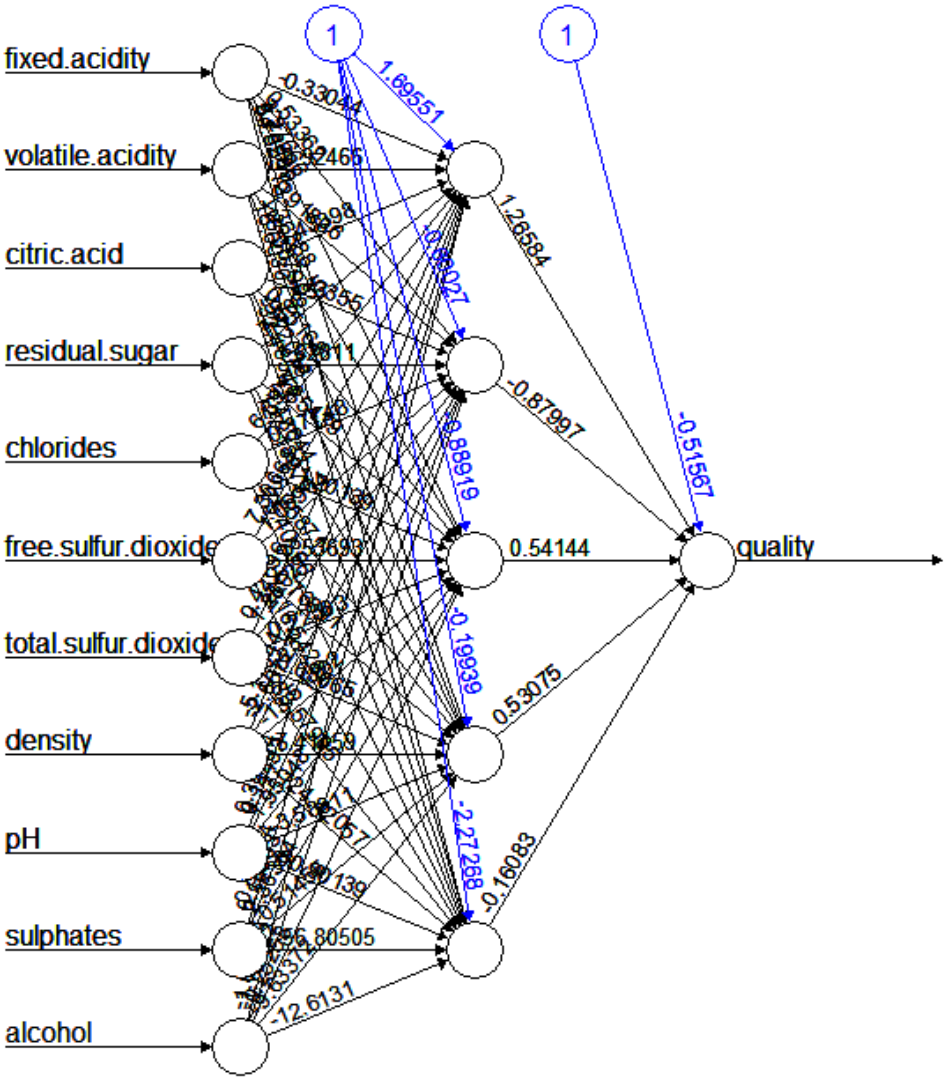
Plots:



For single Hidden layer:



For 5 Hidden layers:



In ANN Model, firstly I performed it using only single hidden layer and, in that case, I got the accuracy (here, correlation) as 48.28% (70:30 Split).

Then I performed it using 5 hidden layers and, in that case, I got the accuracy (here, correlation) as 57.35% (70:30 Split).

This implies that ANN is not the right fit for my model as the accuracy here is very low.

This is due to the fact that on my dataset there are a lot number of parameters and there is the overfitting problem with them.

Objective 5: Comparative analysis of the above-mentioned Classification models.

S.No.	Classification Model	Accuracy	Remarks
1	KNN Model	99.49%	The accuracy is very high, and this implies that KNN fits optimally on my dataset. The reason for this much high accuracy is that a nearest neighbor approach is working for our classes i.e., it is able to form correct neighbors that can differentiate the classes.
2	Decision Tree	2.29%	The accuracy is very low, and this implies that decision tree model is not the right fit for my dataset. This much less accuracy is due to oversampling and undersampling of the classes. As we saw through the plot (decision tree) that in our sample 4 classes (3,4,8,9) were unused and majority of the classification is done from prominent classes only.
3	SVM Model	52.88%	The accuracy is low, and this implies that SVM model is not the right fit for my dataset. This much less accuracy is due to the fact that there are not very distinct levels/classes, and the classes will be very much randomized in the plot. It is not possible to find a linear hyperplane for them.
4	ANN Model	57.35%	In ANN Model, the accuracy is low, this implies that ANN is not the right fit for my model. This much less accuracy is due to the fact that on my dataset there are a lot number of parameters and there is the overfitting problem with them.

Therefore, from the above-mentioned observations we can conclude that for my dataset (white wine quality dataset) KNN fits the best and it is the optimal model with 99.49% accuracy.

Objective 6: Application of MLR.

Multiple Linear Regression:

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is –

$$\boldsymbol{y = a + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n}$$

Following is the description of the parameters used −

- y is the response variable.
- a, b_1, b_2...b_n are the coefficients.
- x_1, x_2, ...x_n are the predictor variables.

We create the regression model using the lm() function in R.

The model determines the value of the coefficients using the input data. Next, we can predict the value of the response variable for a given set of predictor variables using these coefficients.

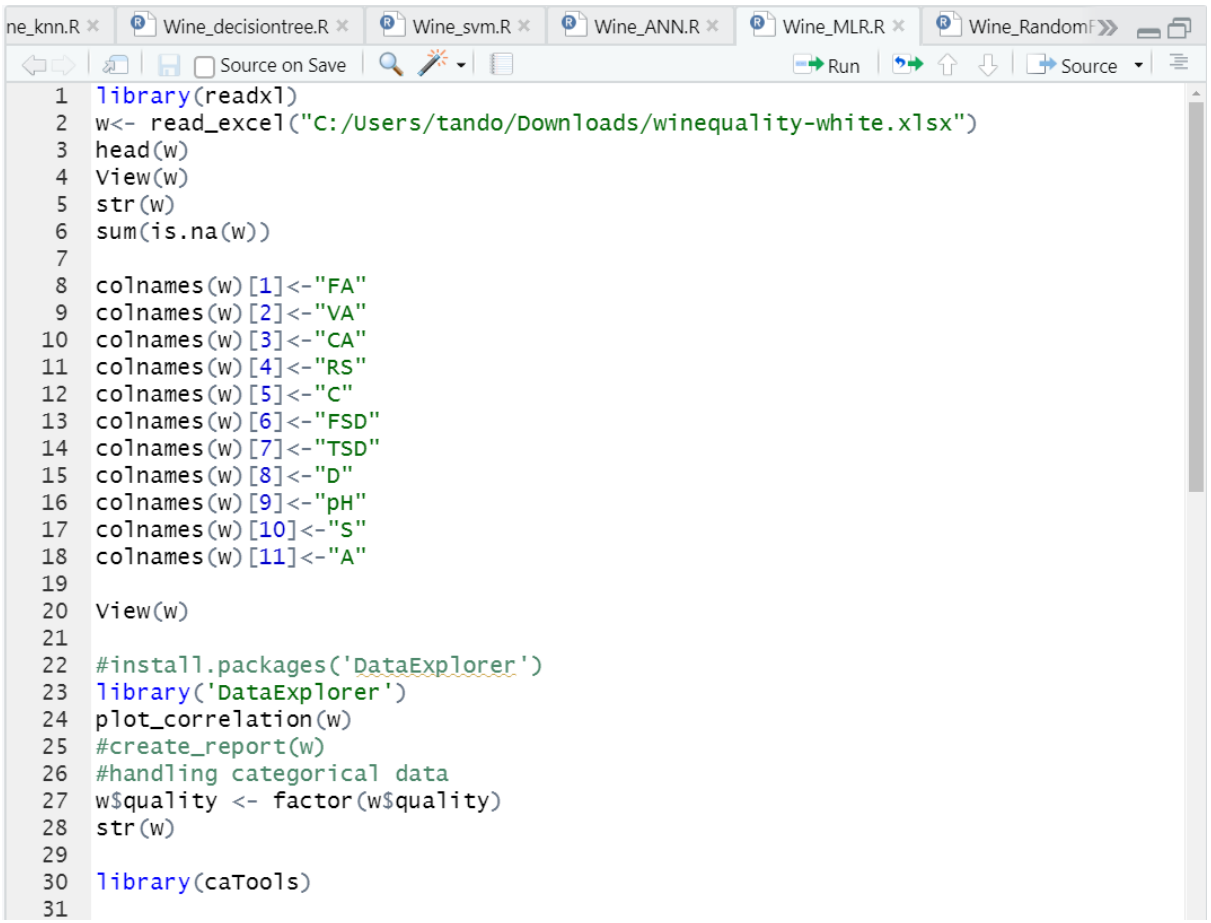
Application of MLR Model:

- **Libraries used:**
    - readxl: To import/load the dataset into R we used the "readxl" library.
    - DataExplorer: Simplify and automate EDA (Exploratory Data Analysis) process and report generation.
    - caTools: It was used for sample splitting.
    - modelr: It was used to calculate the rsquared value of the model.
    - ggplot2: ggplot2 is a plotting package that provides helpful commands to create complex plots.

- **Steps involved in the MLR Model:**
    - Using the readxl library the dataset was imported.
    - Then data exploration was carried out.
    - We checked for the null values and in my dataset, there were none.
    - Also here, I changed my column names for the ease.
    - Correlation for the dataset was plotted.

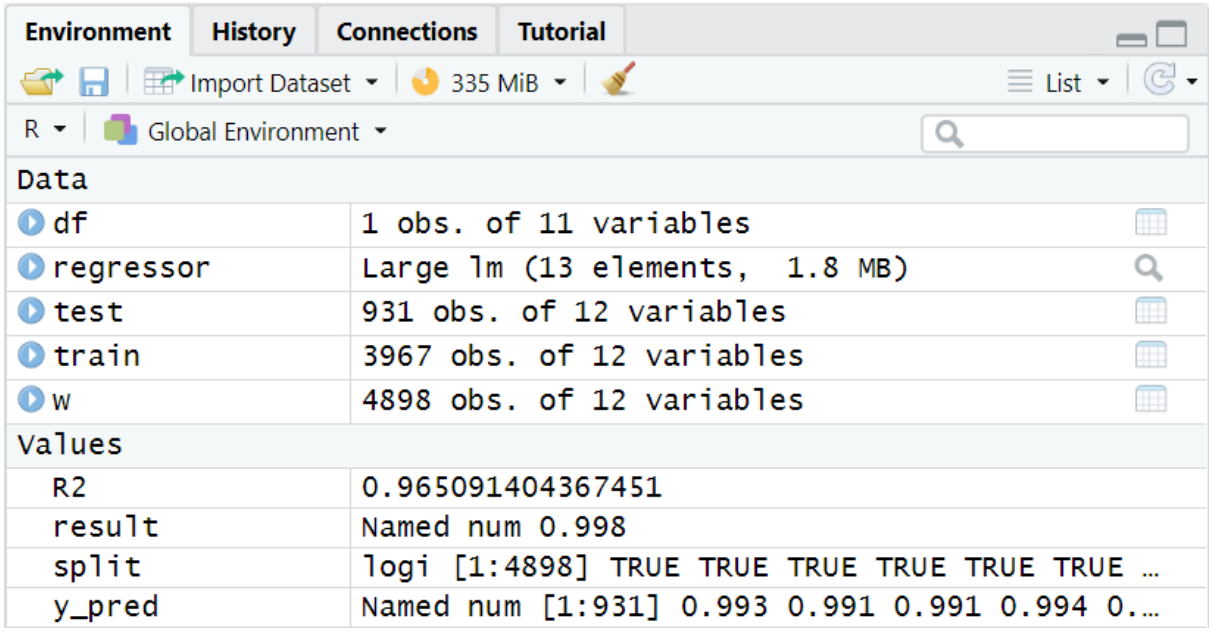
- And then “quality” column was converted into factor.
- There are 7 distinct levels/classes.
- The data was then split into Training and Testing data.
- Then the MLR model was applied on the training dataset.
- The model then made predictions through independent variables with respect to the dependent variable i.e., density here.
- Then the predictions on the test dataset were made.
- After that, I passed some values from the dataset and checked for the value of density that my model would predict for that dataset.
- At last, I calculated the accuracy of my regression model using rsquare function.
- Finally, a plot between one of the independent variable (Fixed Acidity) and density which is our dependent variable was plotted.

Code:



```r
#splitting the dataset
set.seed(1234)

split <- sample.split(w$D, SplitRatio=0.8)

train <- subset(w, split==TRUE)
test <- subset(w, split==FALSE)

#fitting multiple linear regression model
regressor=lm(formula = D~.,
             data=train)
y_pred=predict(regressor,newdata=test)
y_pred

#prediction
df <- data.frame(FA = 7.4,VA = 0.25, CA = 0.37, RS = 13.5,
                 C = 0.06, FSD = 52, TSD = 192, pH=3,
                 S = 0.44, A = 9.1, quality = as.factor(5))

result <- predict(regressor,df)
print(result)

#install.packages("modelr")
library(modelr)
R2 = rsquare(regressor, data = w)
R2*100

library(ggplot2)
ggplot(train, aes(FA, D)) +
  geom_smooth(method="lm") +
  geom_point(size=1) +
  theme_bw() +
  xlab("Fixed Acidity") +
  ylab("Density") +
  ggtitle("FA vs D")

67:1  (Top Level)  R Script
```

Environment Variables:



Console:

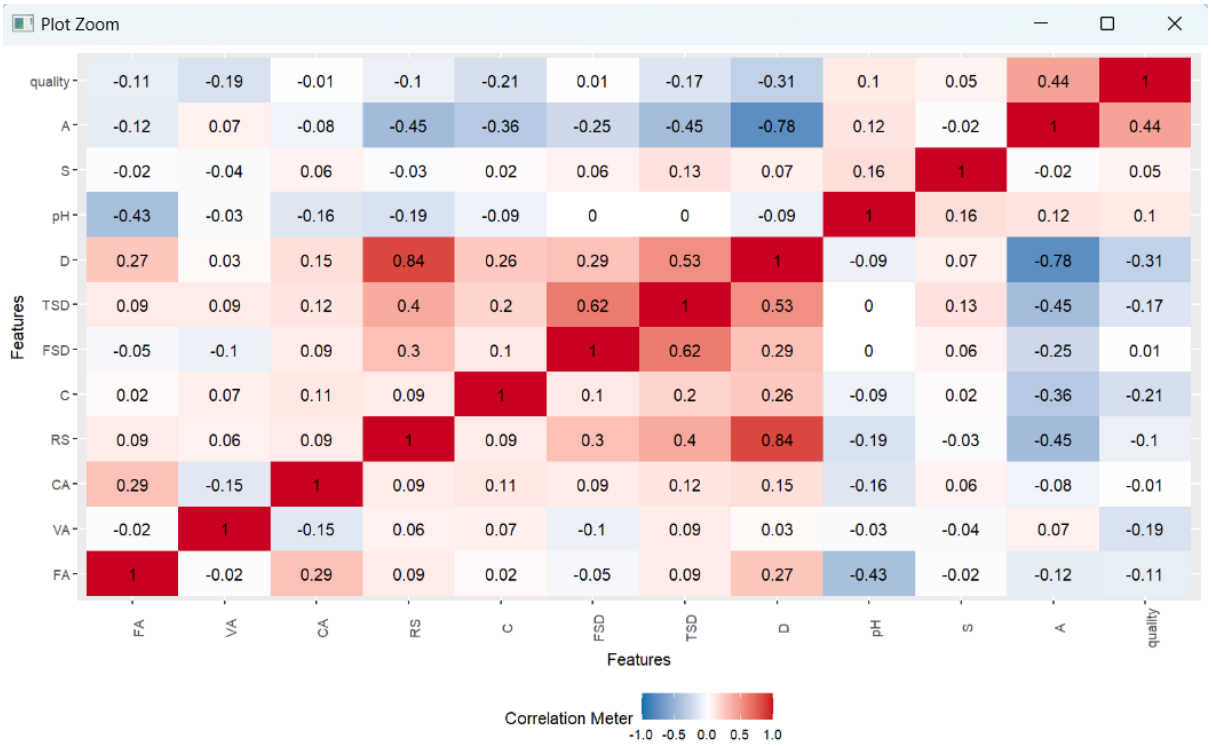
```
R 4.1.2 · ~/Predictive Analysis/
> library(readxl)
Warning message:
package 'readxl' was built under R version 4.1.3
> w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
> head(w)
# A tibble: 6 x 12
  fixed ac~1 volat~2 citri~3 resid~4 chlor~5 free ~6 total~7 density    pH sulph~8
       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
1        7      0.27    0.36    20.7   0.045      45     170   1.00    3       0.45
2        6.3    0.3     0.34     1.6   0.049      14     132   0.994   3.3     0.49
3        8.1    0.28    0.4      6.9   0.05       30      97   0.995   3.26    0.44
4        7.2    0.23    0.32     8.5   0.058      47     186   0.996   3.19    0.4
5        7.2    0.23    0.32     8.5   0.058      47     186   0.996   3.19    0.4
6        8.1    0.28    0.4      6.9   0.05       30      97   0.995   3.26    0.44
# ... with 2 more variables: alcohol <dbl>, quality <dbl>, and abbreviated
#   variable names 1: `fixed acidity`, 2: `volatile acidity`, 3: `citric acid`,
#   4: `residual sugar`, 5: chlorides, 6: `free sulfur dioxide`,
#   7: `total sulfur dioxide`, 8: sulphates
# i Use `colnames()` to see all variable names
> View(w)
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22
 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43
 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.
049 0.044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45
 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
> colnames(w)[1]<-"FA"
> colnames(w)[2]<-"VA"
> colnames(w)[3]<-"CA"
> colnames(w)[4]<-"RS"
> colnames(w)[5]<-"C"
> colnames(w)[6]<-"FSD"
> colnames(w)[7]<-"TSD"
> colnames(w)[8]<-"D"
> colnames(w)[9]<-"pH"
> colnames(w)[10]<-"S"
> colnames(w)[11]<-"A"
> View(w)
> #install.packages('DataExplorer')
> library('DataExplorer')
> plot_correlation(w)
> #create_report(w)
> #handling categorical data
> w$quality <- factor(w$quality)
> str(w)
tibble [4,898 x 12] (S3: tbl_df/tbl/data.frame)
 $ FA     : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ VA     : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ CA     : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ RS     : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ C      : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ FSD    : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ TSD    : num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ D      : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
```

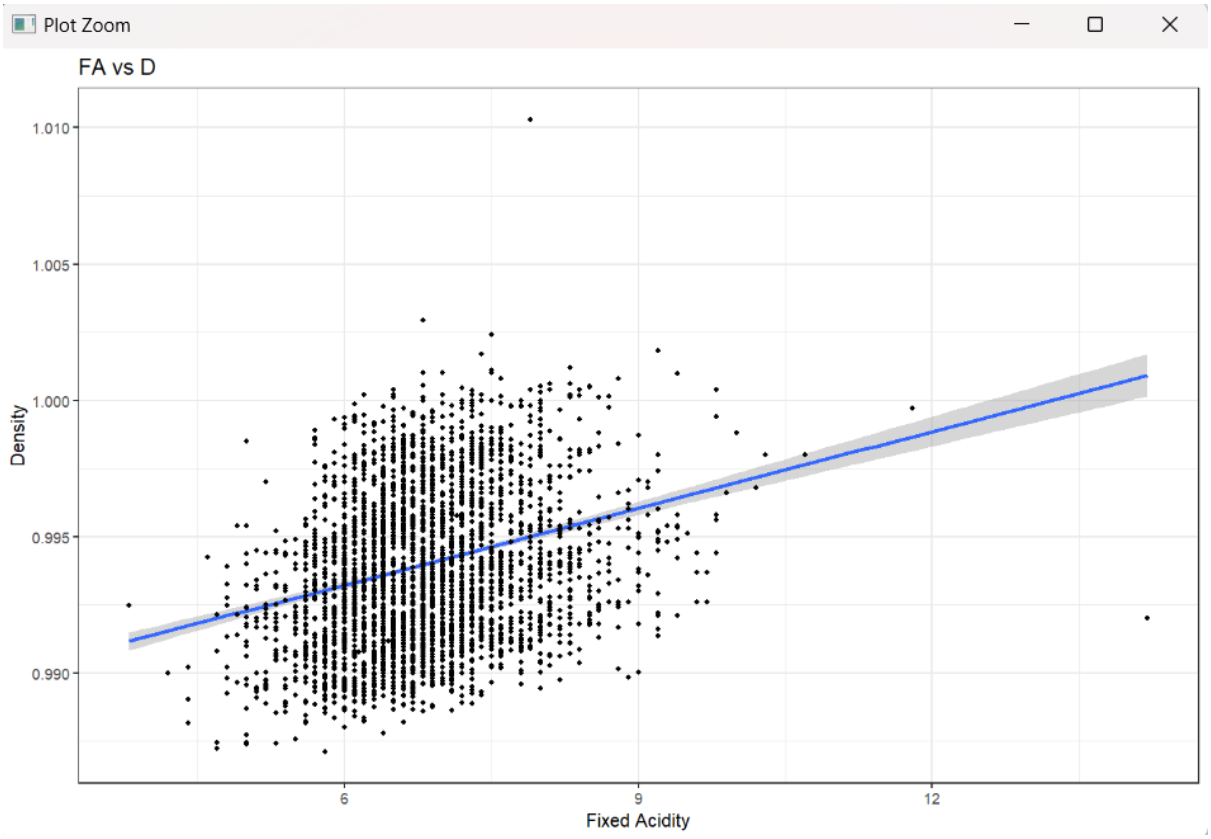
```
 $ pH     : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ S      : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ A      : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality: Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 4 4 4 4 ...
> library(caTools)
Warning message:
package 'caTools' was built under R version 4.1.3
> #splitting the dataset
> set.seed(1234)
> split <- sample.split(w$D, SplitRatio=0.8)
> train <- subset(w, split==TRUE)
> test <- subset(w, split==FALSE)
> #fitting multiple linear regression model
> regressor=lm(formula = D~.,
+              data=train)
> y_pred=predict(regressor,newdata=test)
> y_pred
        1         2         3         4         5         6         7         8
0.9925233 0.9914322 0.9909772 0.9935195 0.9915324 0.9913320 0.9965622 0.9954057
        9        10        11        12        13        14        15        16
0.9957890 0.9940355 0.9965622 0.9968041 0.9980882 0.9977000 0.9995389 0.9930709
       17        18        19        20        21        22        23        24
0.9968830 0.9907107 0.9981599 0.9927863 0.9923753 0.9960839 0.9991032 0.9915954
```

- - -

```
      913       914       915       916       917       918       919       920
0.9904123 0.9956201 0.9963166 0.9937476 0.9911806 0.9893583 0.9940337 0.9922023
      921       922       923       924       925       926       927       928
0.9942626 0.9966109 0.9969193 0.9932454 0.9896449 0.9901229 0.9978155 0.9926531
      929       930       931
0.9914855 0.9897301 0.9953802
> #prediction
> df <- data.frame(FA = 7.4,VA = 0.25, CA = 0.37, RS = 13.5,
+                  C = 0.06, FSD = 52, TSD = 192, pH=3,
+                  S = 0.44, A = 9.1, quality = as.factor(5))
> result <- predict(regressor,df)
> print(result)
        1
0.9981394
> #install.packages("modelr")
> library(modelr)
> R2 = rsquare(regressor, data = w)
> R2*100
[1] 96.50914
> library(ggplot2)
> ggplot(train, aes(FA, D)) +
+   geom_smooth(method="lm") +
+   geom_point(size=1) +
+   theme_bw() +
+   xlab("Fixed Acidity") +
+   ylab("Density") +
+   ggtitle("FA vs D")
`geom_smooth()` using formula = 'y ~ x'
> 
```

Plots:





On applying Multiple Linear Regression model to my dataset, I attained 96.50% (80:20 Split) accuracy which is very high, and this implies that MLR model is the right fit for my dataset to predict the values of the dependent variable i.e., density.

Objective 7: Application of Random Forest

Random Forest:

Random Forest approach is a supervised learning algorithm. It builds the multiple decision trees which are known as forest and glue them together to urge a more accurate and stable prediction. The random forest approach is similar to the ensemble technique called as Bagging. In this approach, multiple trees are generated by bootstrap samples from training data and then we simply reduce the correlation between the trees. Performing this approach increases the performance of decision trees and helps in avoiding overriding. In this article, let's learn to use a random forest approach for regression in R programming.

Application of Random Forest Model:

- **Libraries used:**
  - readxl: To import/load the dataset into R we used the "readxl" library.
  - randomForest: It implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression.
  - modelr: It was used to calculate the rsquared value of the model.

- **Steps involved in the Random Forest Model:**
  - Using the readxl library the dataset was imported.
  - Then data exploration was carried out.
  - We checked for the null values and in my dataset, there were none.
  - Also here, I changed my column names for the ease.
  - The data was then split into Training and Testing data.
  - Then the Random Forest model was applied on the dataset.
  - The model then made predictions for the test dataset.
  - The random forest plot summary is calculated.
  - After that, the error vs the number of trees graph was plotted.
  - At last, I calculated the accuracy of my regression model using rsquare function.

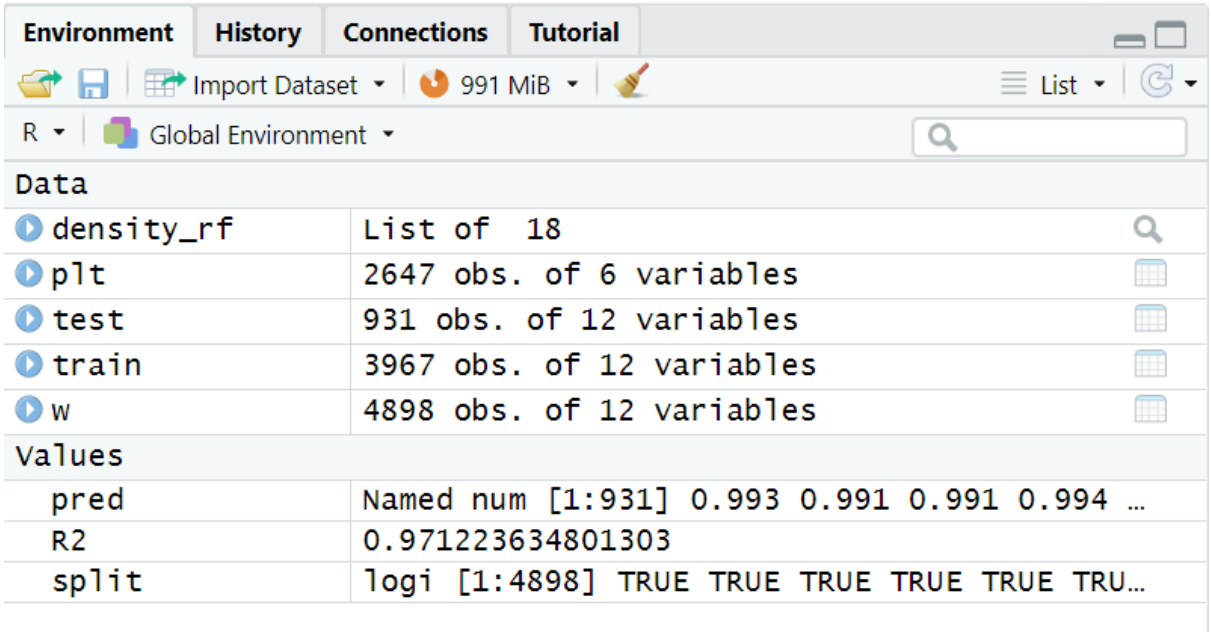
Code:

Wine_MLR.R | Wine_RandomForest.R | Wine_knn.R | Multiple_Linear_Regression.R | Wine_ANN.R

Source on Save | Run | Source

```
1  library(readxl)
2  w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
3  head(w)
4  View(w)
5  str(w)
6  sum(is.na(w))
7
8  colnames(w)[1]<-"FA"
9  colnames(w)[2]<-"VA"
10 colnames(w)[3]<-"CA"
11 colnames(w)[4]<-"RS"
12 colnames(w)[5]<-"C"
13 colnames(w)[6]<-"FSD"
14 colnames(w)[7]<-"TSD"
15 colnames(w)[8]<-"D"
16 colnames(w)[9]<-"pH"
17 colnames(w)[10]<-"S"
18 colnames(w)[11]<-"A"
19
20 View(w)
21 #splitting the dataset
22 set.seed(1234)
23 split <- sample.split(w$D, SplitRatio=0.8)
24 train <- subset(w, split==TRUE)
25 test <- subset(w, split==FALSE)
26
27
28 #install.packages("randomForest")
29 library(randomForest)
30
31 # Create random forest for regression
32 density_rf <- randomForest(D ~ ., data = train)
33
34 # Print regression model
35 density_rf
36 summary(density_rf)
37
38 # Prediction
39 pred = predict(density_rf, test)
40 pred
41
42
43 plt = getTree(density_rf, k=1, labelVar = TRUE)
44 plt
45
46 # Plot the error vs the number of trees graph
47 plot(density_rf)
48
49 library(modelr)
50 R2 = rsquare(density_rf, data = test)
51 R2*100
52
```

Environment Variables:



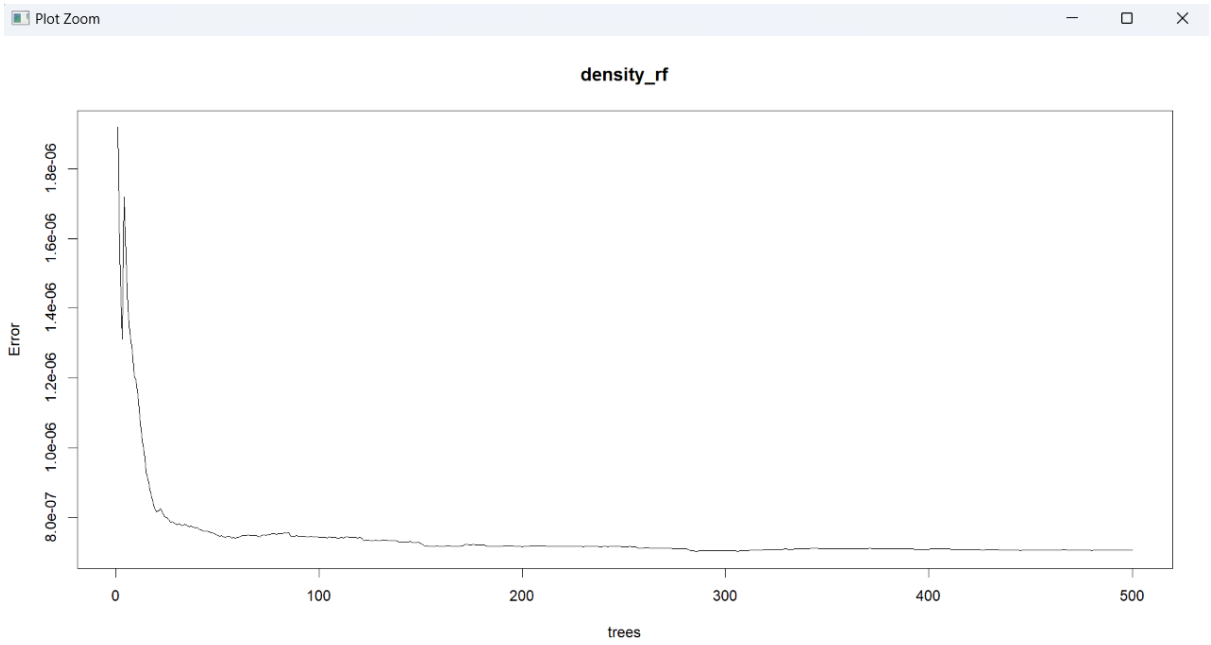
Console:

```
R 4.2.2 · ~/Predictive Analysis/
> library(readxl)
> w<- read_excel("C:/Users/tando/Downloads/winequality-white.xlsx")
> head(w)
# A tibble: 6 × 12
  fixed a…¹ volat…² citri…³ resid…⁴ chlor…⁵ free …⁶ total…⁷ density    pH sulph…⁸ alcohol
      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
1       7      0.27    0.36    20.7   0.045      45     170    1.00   3       0.45     8.8
2       6.3    0.3     0.34     1.6   0.049      14     132    0.994  3.3     0.49     9.5
3       8.1    0.28    0.4      6.9   0.05       30      97    0.995  3.26    0.44    10.1
4       7.2    0.23    0.32     8.5   0.058      47     186    0.996  3.19    0.4      9.9
5       7.2    0.23    0.32     8.5   0.058      47     186    0.996  3.19    0.4      9.9
6       8.1    0.28    0.4      6.9   0.05       30      97    0.995  3.26    0.44    10.1
# … with 1 more variable: quality <dbl>, and abbreviated variable names
#   ¹`fixed acidity`, ²`volatile acidity`, ³`citric acid`, ⁴`residual sugar`,
#   ⁵chlorides, ⁶`free sulfur dioxide`, ⁷`total sulfur dioxide`, ⁸sulphates
# ℹ Use `colnames()` to see all variable names
> View(w)
> str(w)
tibble [4,898 × 12] (S3: tbl_df/tbl/data.frame)
 $ fixed acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.
044 ...
 $ free sulfur dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
> sum(is.na(w))
[1] 0
```

```
> colnames(w)[1]<-"FA"
> colnames(w)[2]<-"VA"
> colnames(w)[3]<-"CA"
> colnames(w)[4]<-"RS"
> colnames(w)[5]<-"C"
> colnames(w)[6]<-"FSD"
> colnames(w)[7]<-"TSD"
> colnames(w)[8]<-"D"
> colnames(w)[9]<-"pH"
> colnames(w)[10]<-"S"
> colnames(w)[11]<-"A"
> View(w)
> #splitting the dataset
> set.seed(1234)
> split <- sample.split(w$D, SplitRatio=0.8)
> train <- subset(w, split==TRUE)
> test <- subset(w, split==FALSE)
> #install.packages("randomForest")
> library(randomForest)
> # Create random forest for regression
> density_rf <- randomForest(D ~ ., data = train)
> # Print regression model
> density_rf

Call:
 randomForest(formula = D ~ ., data = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 7.064311e-07
                    % Var explained: 92.32
> summary(density_rf)
                Length Class  Mode
call               3   -none- call
type               1   -none- character
predicted       3967   -none- numeric
mse              500   -none- numeric
rsq              500   -none- numeric
oob.times       3967   -none- numeric
importance        11   -none- numeric
importanceSD       0   -none- NULL
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            11   -none- list
coefs              0   -none- NULL
y               3967   -none- numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
> # Prediction
> pred = predict(density_rf, test)
> pred
        1         2         3         4         5         6         7         8
0.9930703 0.9914041 0.9913258 0.9937892 0.9925351 0.9914845 0.9963229 0.9954385
        9        10        11        12        13        14        15        16
0.9949371 0.9937529 0.9963229 0.9964504 0.9974728 0.9974563 0.9990465 0.9934254
       17        18        19        20        21        22        23        24
0.9967330 0.9907305 0.9981737 0.9935325 0.9926270 0.9958519 0.9981236 0.9920532

      - - -

      913       914       915       916       917       918       919       920
0.9907151 0.9951961 0.9965415 0.9944860 0.9914962 0.9894327 0.9941366 0.9926622
      921       922       923       924       925       926       927       928
0.9943583 0.9967056 0.9967185 0.9927556 0.9906674 0.9907690 0.9977854 0.9928608
      929       930       931
0.9912209 0.9907957 0.9949893
```

```
> plt = getTree(density_rf, k=1, labelVar = TRUE)
> plt
    left daughter right daughter split var split point status prediction
1               2              3       TSD    144.5000     -3  0.9940956
2               4              5       TSD    109.5000     -3  0.9927441
3               6              7        VA      0.9075     -3  0.9959296
4               8              9       FSD     27.5000     -3  0.9919414
5              10             11         A     10.7500     -3  0.9934308
6              12             13         C      0.0435     -3  0.9959036
7               0              0      <NA>      0.0000     -1  1.0177900
8              14             15   quality      6.5000     -3  0.9921646
9              16             17         A     10.7500     -3  0.9914873
10             18             19        pH      3.1050     -3  0.9950993

      ---

160           286            287       FSD     33.0000     -3  0.9976587
161           288            289        RS     16.4500     -3  0.9995309
162           290            291       TSD    227.5000     -3  0.9968734
163           292            293       TSD    169.5000     -3  0.9984720
164           294            295        pH      3.5600     -3  0.9948277
165           296            297        CA      0.7450     -3  0.9957938
166           298            299        RS      6.0500     -3  0.9934360
 [ reached 'max' / getOption("max.print") -- omitted 2481 rows ]
> # Plot the error vs the number of trees graph
> plot(density_rf)
> library(modelr)
> R2 = rsquare(density_rf, data = test)
> R2*100
[1] 97.12236
```

Plots:



On applying Random Forest to my dataset, I attained 97.12% (80:20 Split) accuracy which is very high, and this implies that Random Forest model is the right fit for my dataset to predict the values of the dependent variable i.e., density.

Also, the plot depicts that as the number of trees are increasing, the error value is decreasing.

Objective 8: Comparative analysis of the above-mentioned Regression models.

S.No.	Regression Model	Accuracy	Remarks
1	MLR Model	96.50%	The accuracy is very high, which implies that MLR model is the right fit for my dataset to predict the values of the dependent variable i.e., density. This much high accuracy is due to the fact that in MLR the dependent variable is linearly related with all the independent variables.
2	Random Forest	97.12%	The accuracy is very high, which implies that Random Forest model is the best fit for my dataset to predict the values of the dependent variable i.e., density. This much high accuracy is due to the fact that in random forest there are multiple trees.