Part 1:

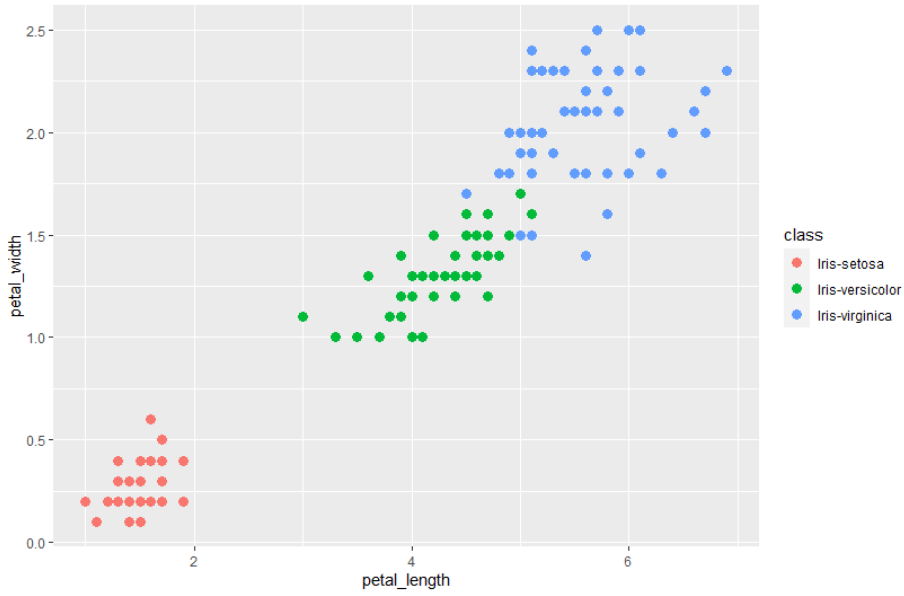
Rows: 150 Columns: 5
Delimiter: ","
chr (1): X5
dbl (4): X1, X2, X3, X4

a) There are 150 instances in this dataset.
b) There are 5 features in the dataset, out of which 4 are input features, and 1 is the output variable.

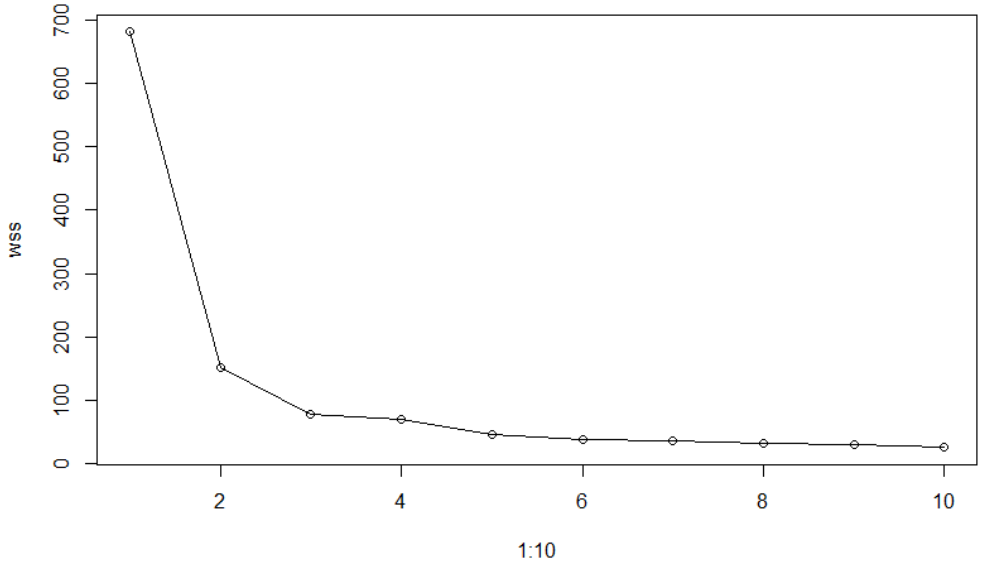
1) i) We have loaded the iris dataset in R. The dataset describes the sepal length, sepla width, petal length and petal width for 3 species of iris, namely, iris-setosa, iris-versicolor and iris-virginica.

	sepal_length	sepal_width	petal_length	petal_width
Min.	4.3	2	1	0.100
1st Qu.	5.1	2.8	1.6	0.300
Median	5.8	3	4.35	1.300
Mean	5.843	3.054	3.759	1.199
3rd Qu.	6.4	3.3	5.1	1.800
Max.	7.9	4.4	6.9	2.500

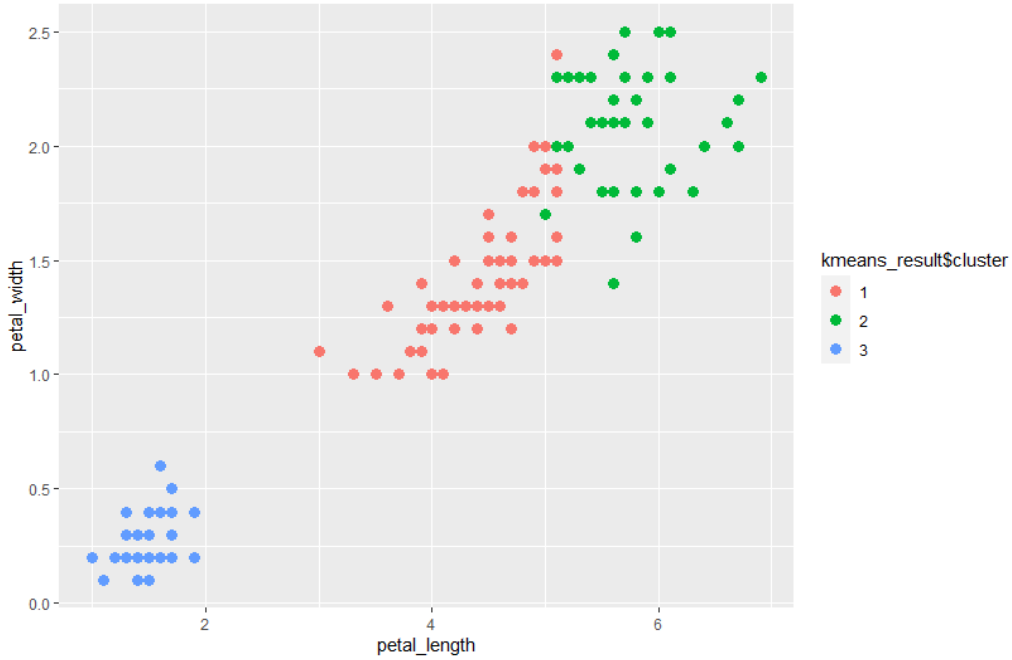
Iris-setosa	Iris-versicolor	Iris-virginica
50	50	50



1) ii) From the elbow plot, we can see that 3 is the optimal number of clusters.

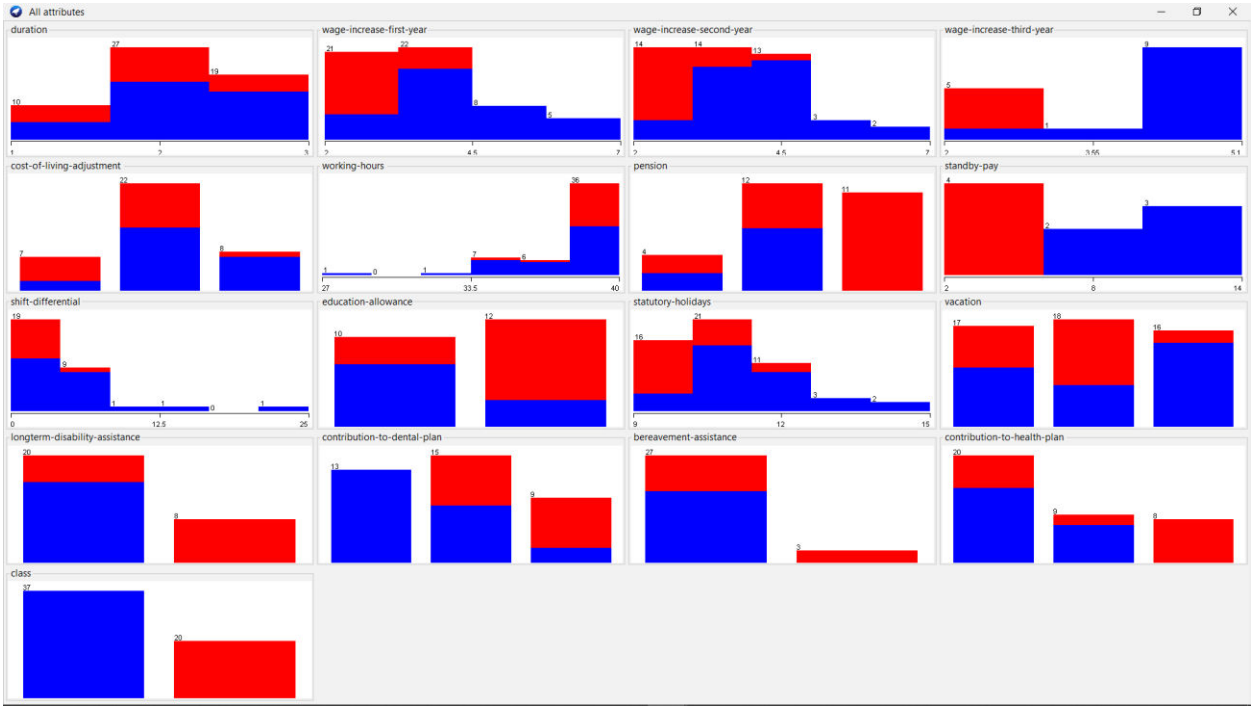


	1	2	3
Iris-setosa	0	0	50
Iris-versicolor	48	2	0
Iris-virginica	14	36	0

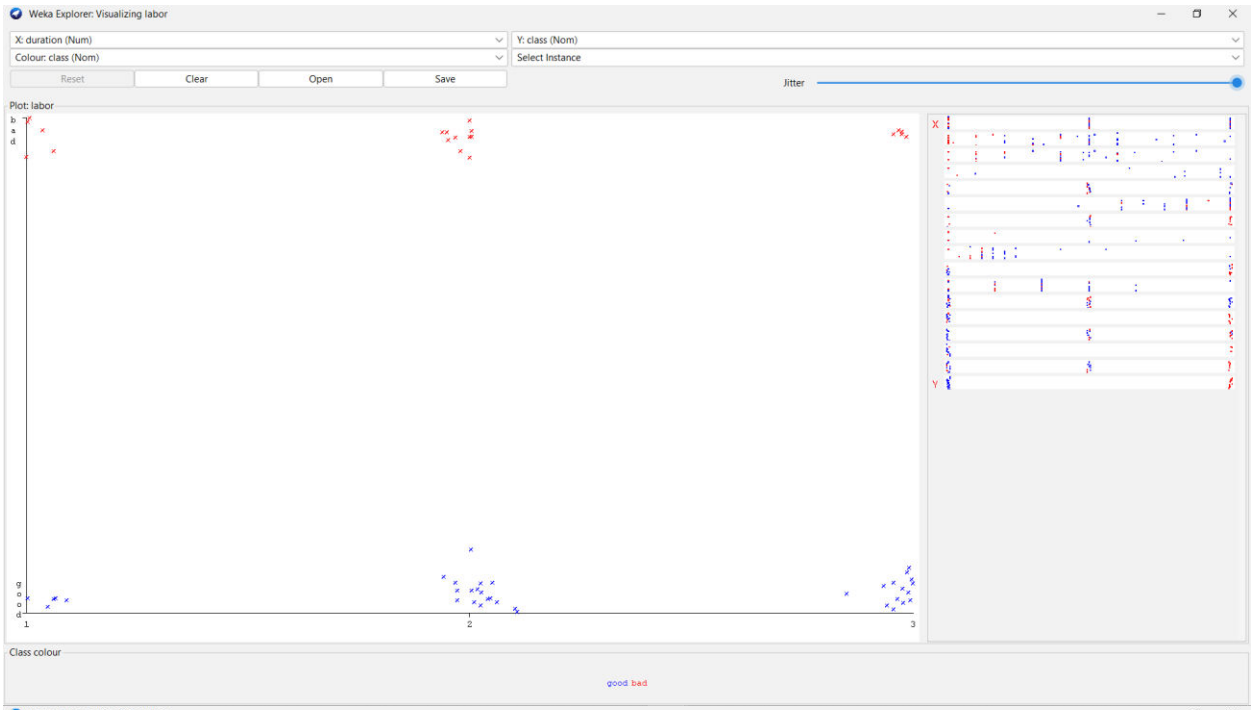


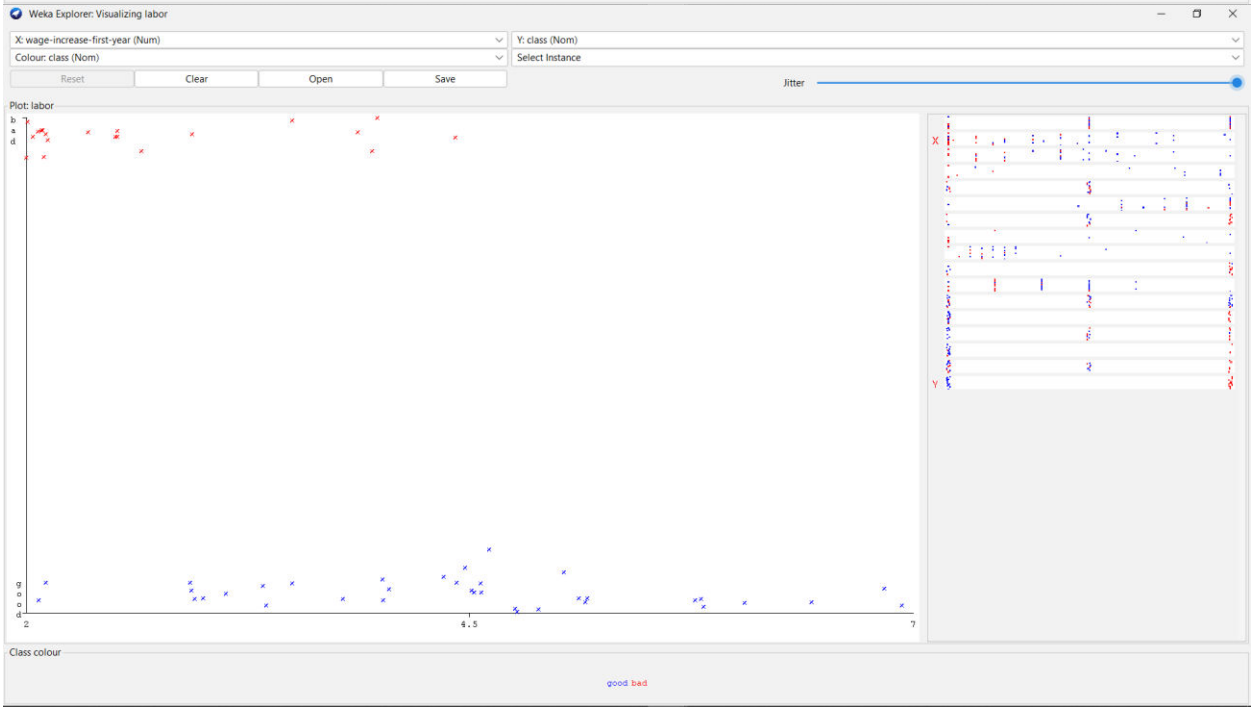
Part 2:

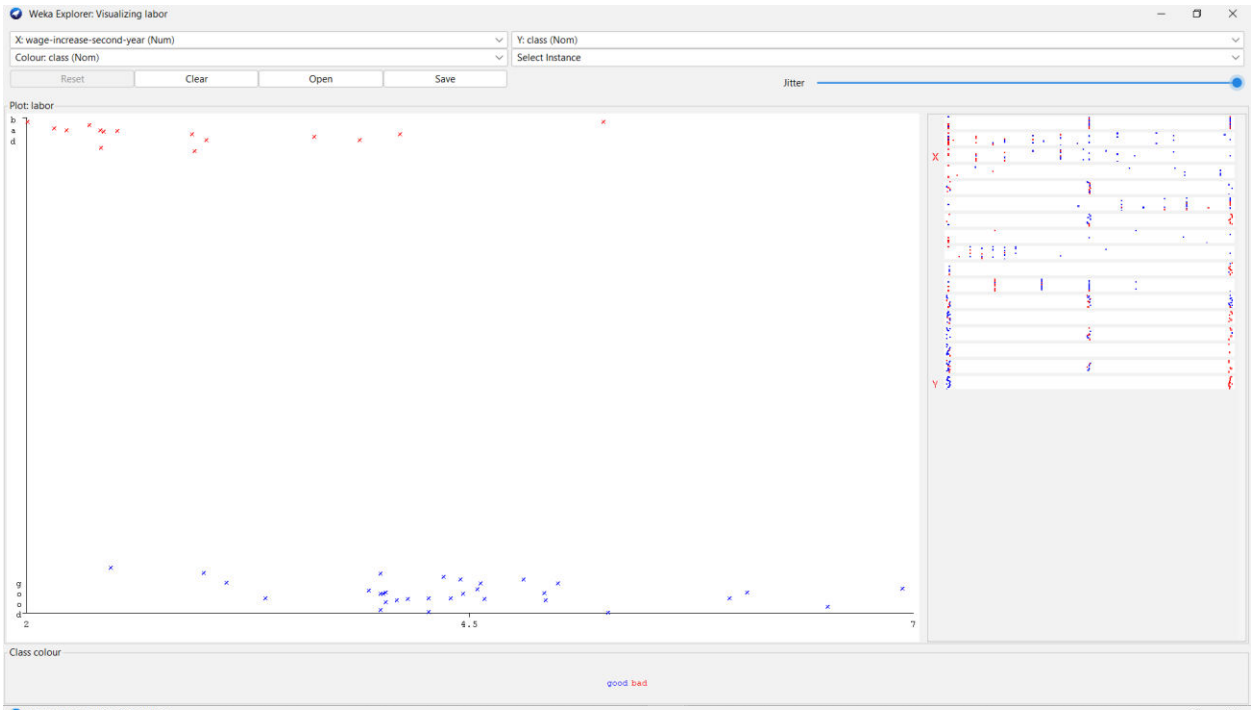
1) We have restructured the dataset as per Weka format.
2) We have uploaded the data to Weka explorer.
3) Distribution of data over classes:

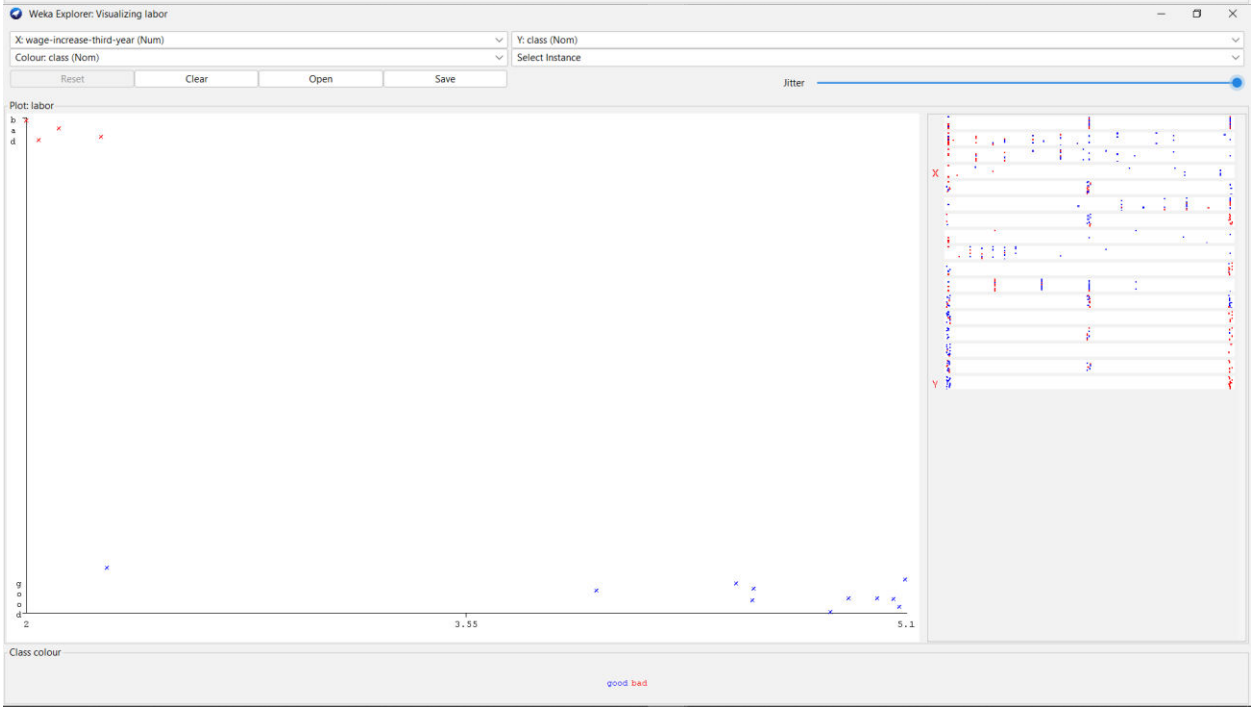


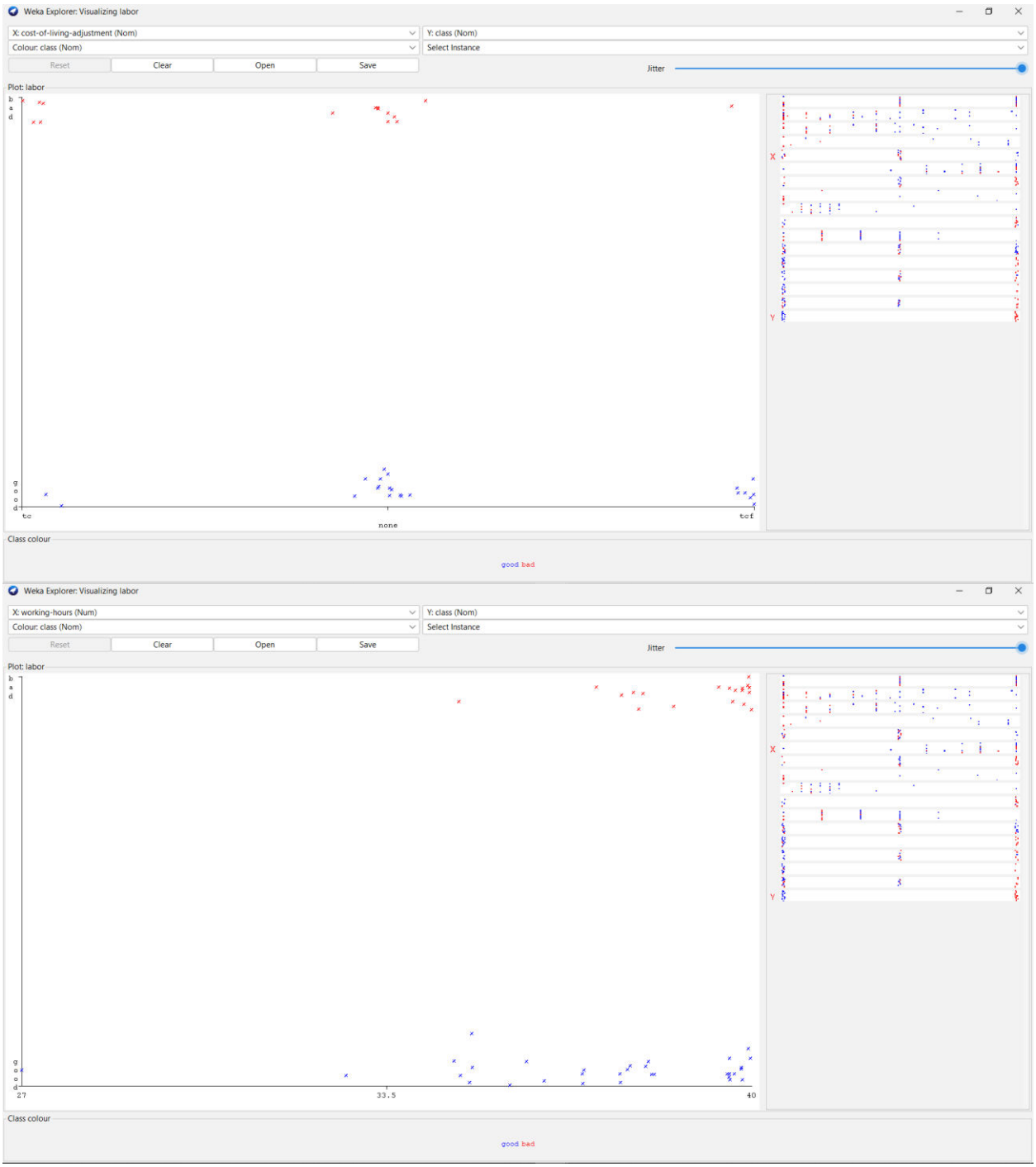
1. Labors with longer duration of agreement have more probability of being good.
2. Labors with higher wage increase in first year of contract are almost surely good.
3. Labors with higher wage increase in second year of contract are almost surely good.
4. Labors with higher wage increase in third year of contract are almost surely good.
5. Labors with higher cost of living allowance are highly likely to be good.
6. Labors with lower working hours are highly likely to be good.
7. Labors with higher pension are almost surely bad.
8. Labors with lower standby-pay are almost surely bad, whereas labors with higher standby-pay are almost surely good.
9. Labors with higher shift-differencial are almost surely good.
10. Labors with higher education allowance are more likely to be bad.
11. Labors with more statutory-holidays are almost surely good.
12. Labors with more vacations are more likely to be good
13. Labors with higher longterm disability assistance are almost surely bad.
14. Labors with less contribution to dental plan are almost surely good.
15. Labors with more bereavement assistance are almost surely bad.
16. Labors with more contribution to health plan are almost surely bad.

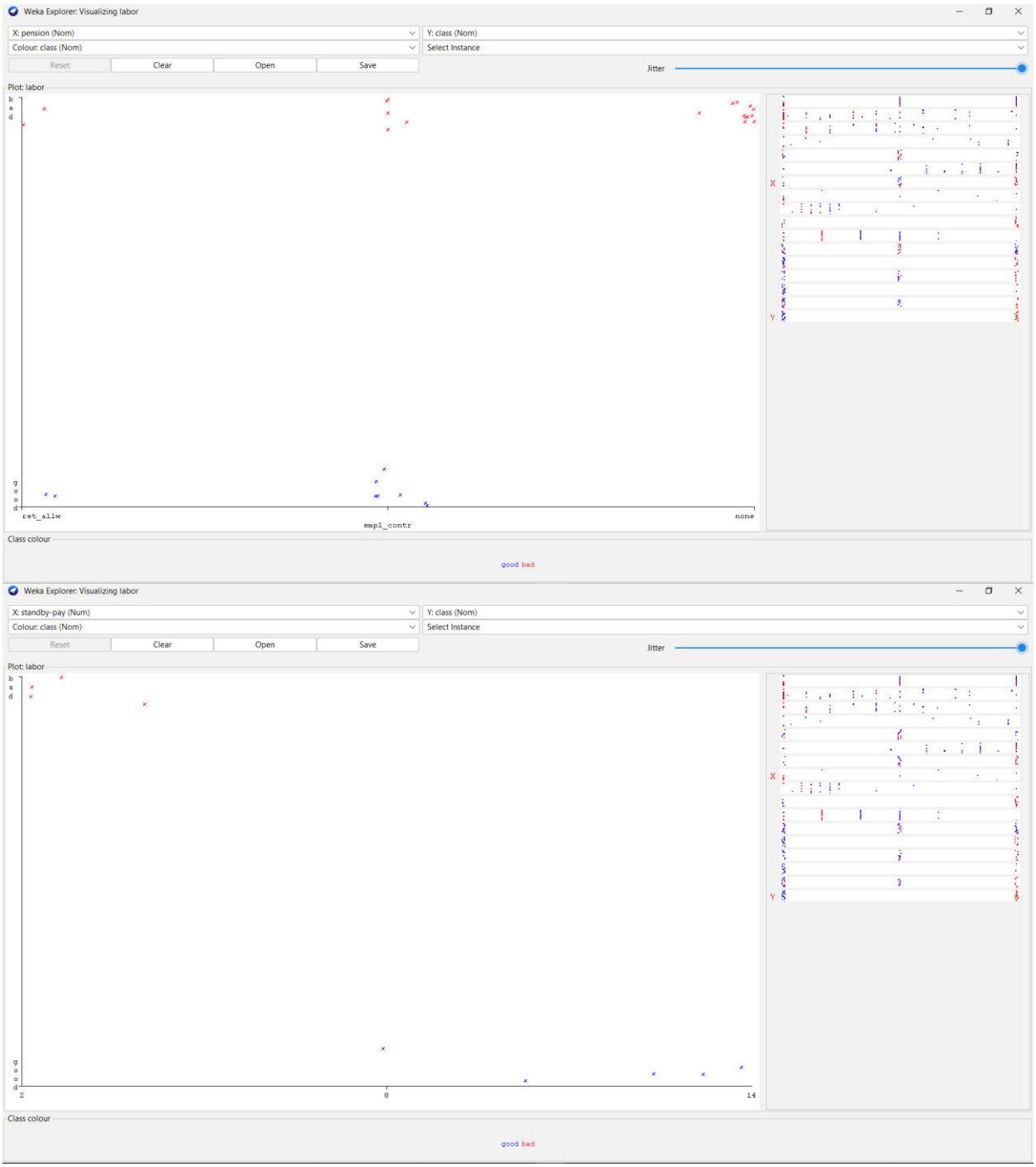

Weka Explorer: Visualizing labor
X: duration (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
1
2
3
X
Y
Class colour
good bad

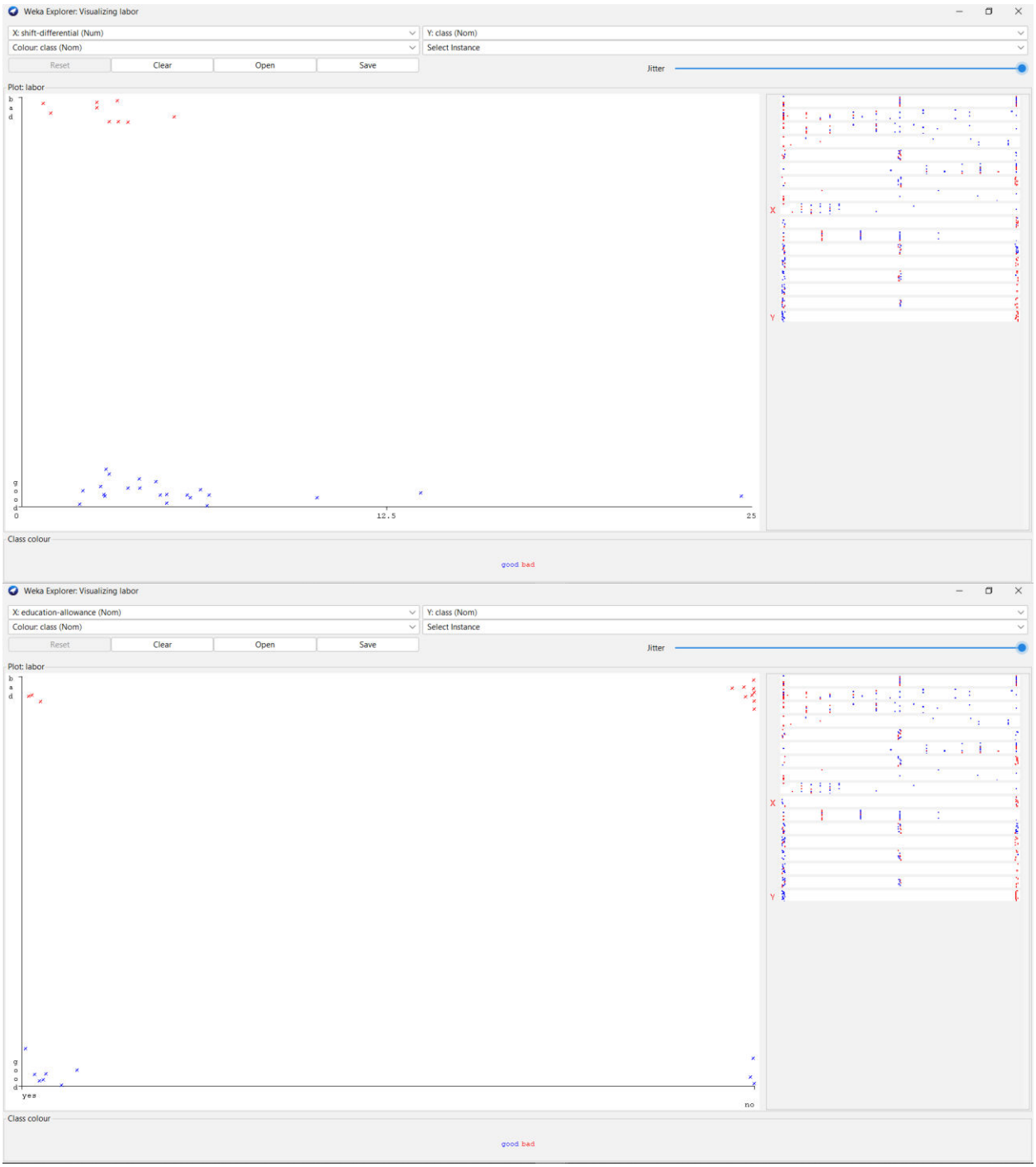

Weka Explorer: Visualizing labor
X: wage-increase-first-year (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
2
4.5
7
X
Y
Class colour
good bad

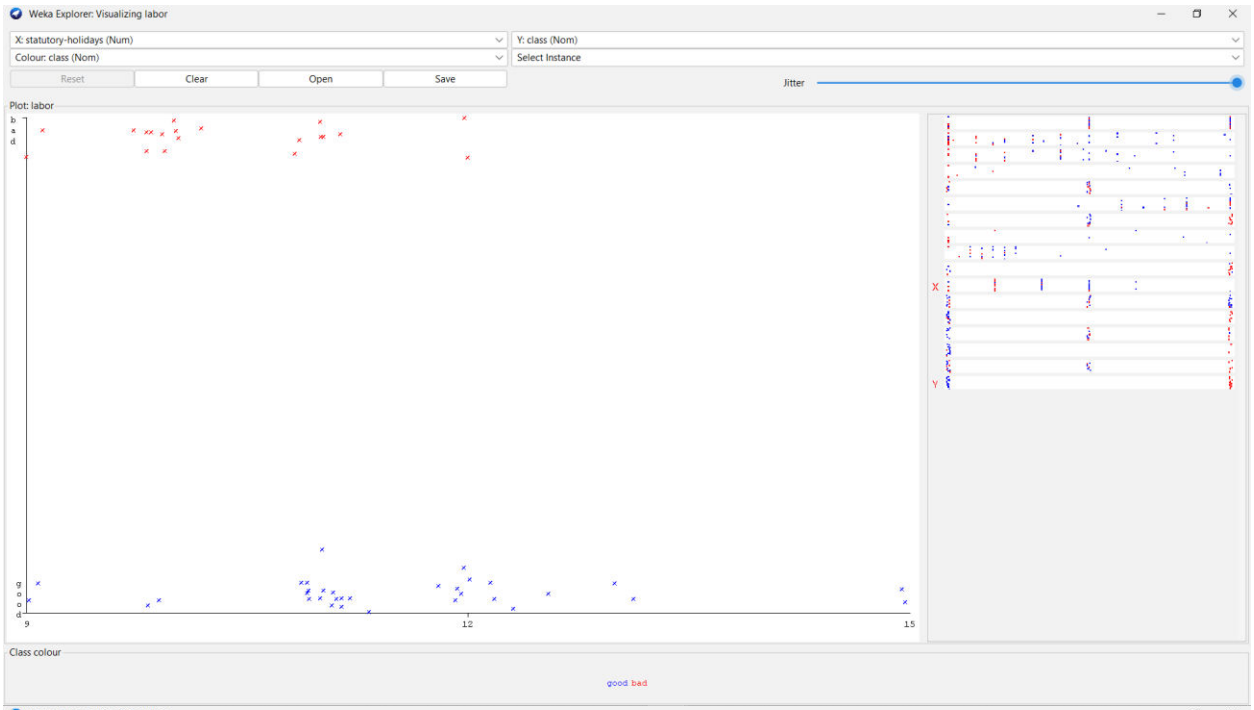

Weka Explorer: Visualizing labor
X: wage-increase-second-year (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
2
4.5
7
X
Y
Class colour
good bad

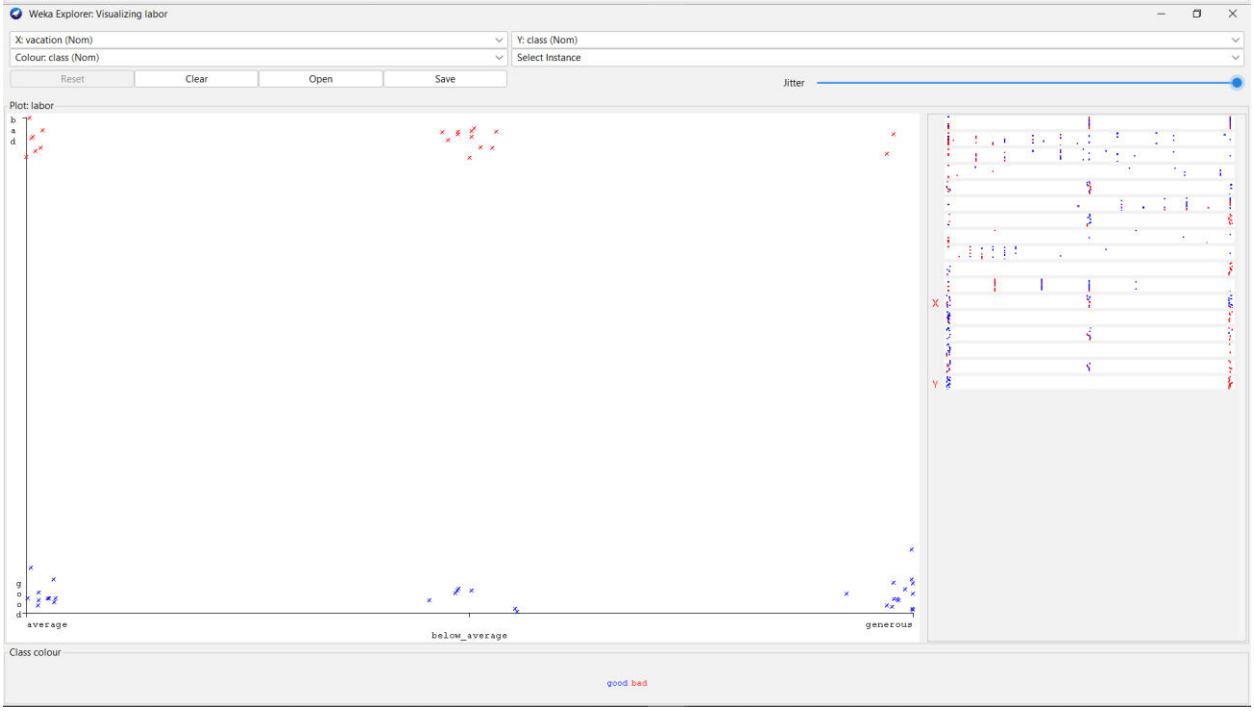

Weka Explorer: Visualizing labor
X: wage-increase-third-year (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
2
3.55
5.1
X
Y
Class colour
good bad

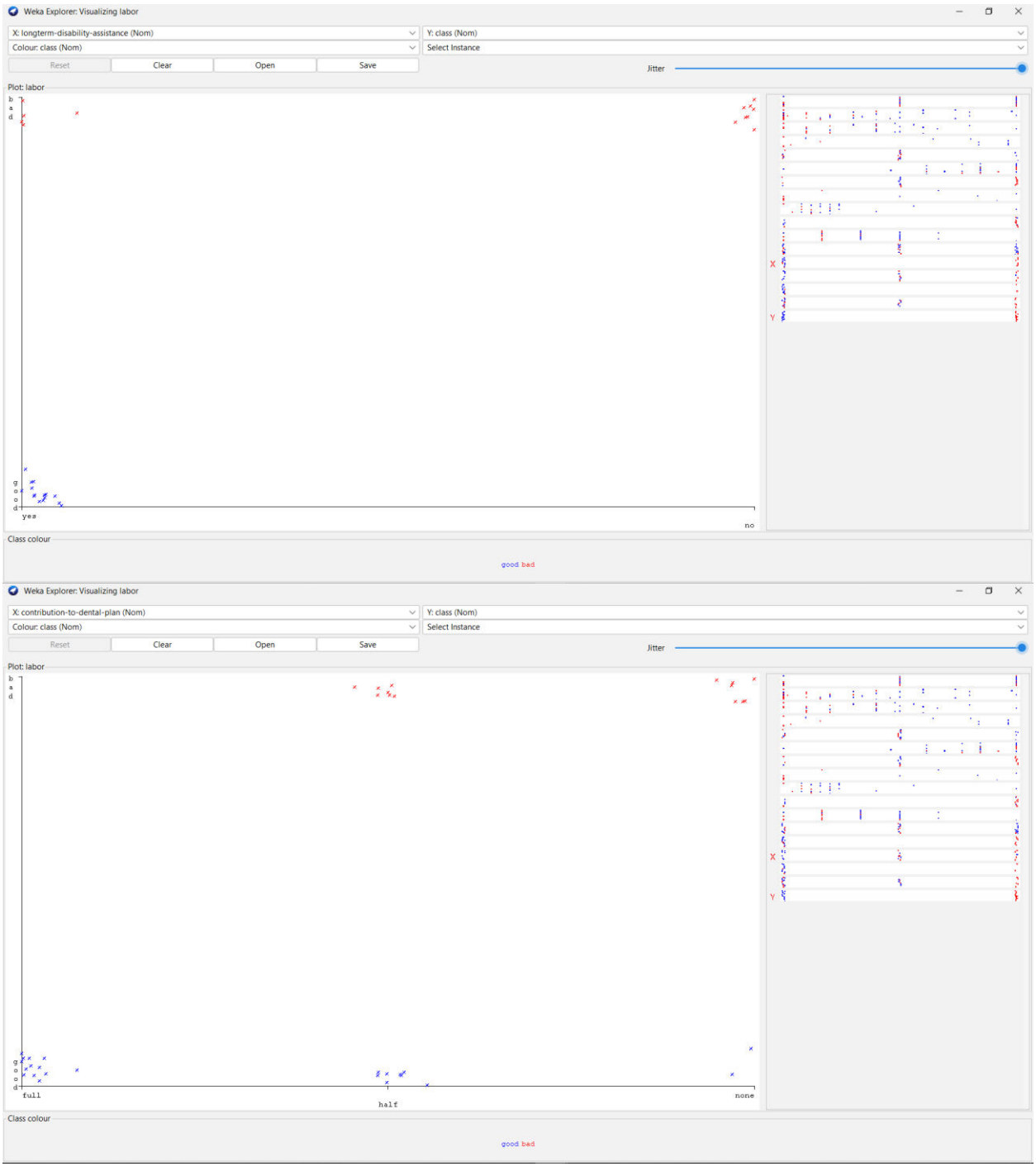

Weka Explorer: Visualizing labor
X: cost-of-living-adjustment (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
tc
none
tcf
X
Y
Class colour
good bad
Weka Explorer: Visualizing labor
X: working-hours (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
27
33.5
40
X
Y
Class colour
good bad

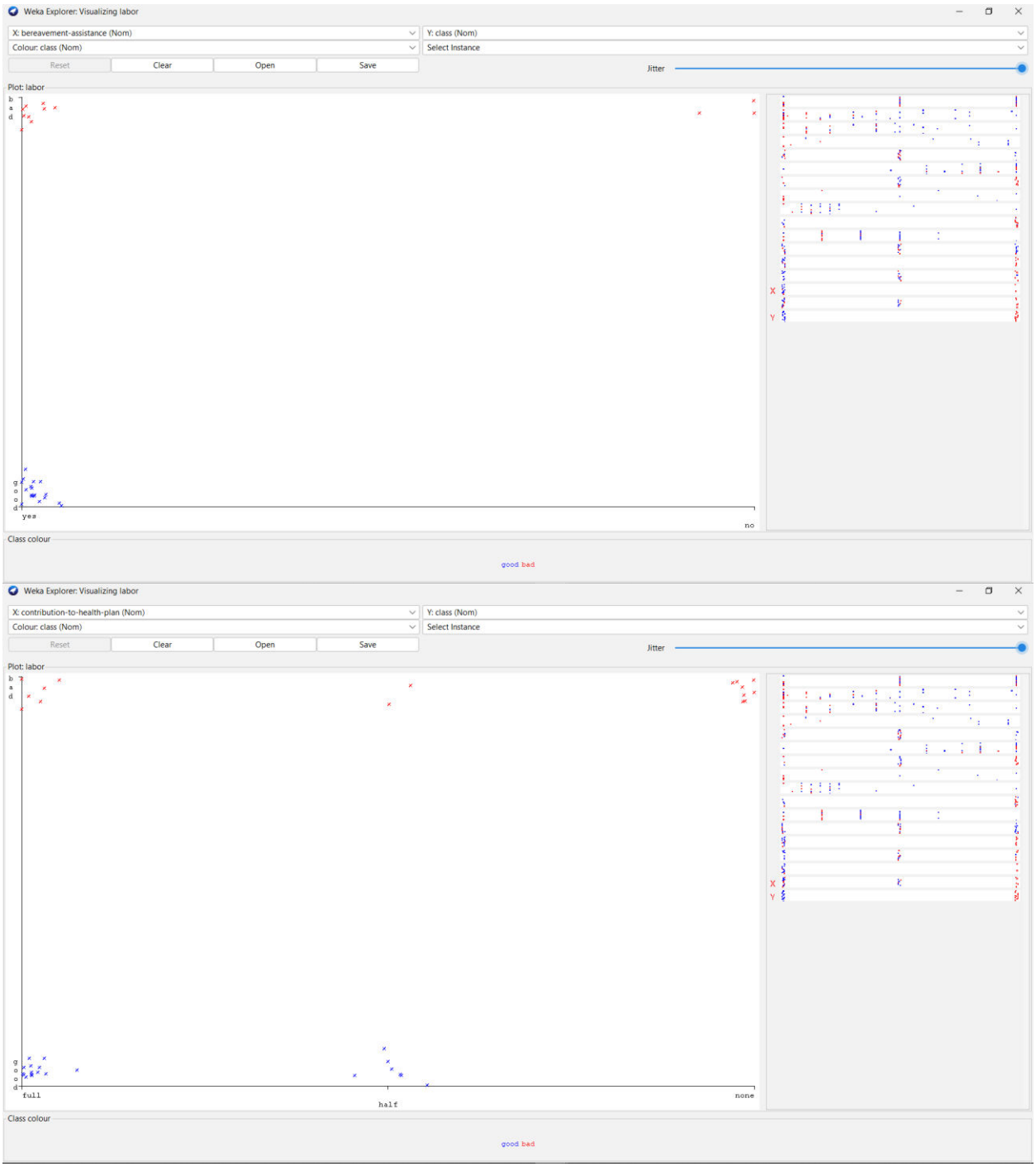

Weka Explorer: Visualizing labor
X: pension (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
ret_allw
empl_contr
none
X
Y
Class colour
good bad
Weka Explorer: Visualizing labor
X: standby-pay (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
2
8
14
X
Y
Class colour
good bad


Weka Explorer: Visualizing labor
X: shift-differential (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
0
12.5
25
X
Y
Class colour
good bad
Weka Explorer: Visualizing labor
X: education-allowance (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
yes
no
X
Y
Class colour
good bad


Weka Explorer: Visualizing labor
X: statutory-holidays (Num)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
9
12
15
X
Y
Class colour
good bad


Weka Explorer: Visualizing labor
X: vacation (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
average
below_average
generous
X
Y
Class colour
good bad


Weka Explorer: Visualizing labor
X: longterm-disability-assistance (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
yes
no
X
Y
Class colour
good bad
Weka Explorer: Visualizing labor
X: contribution-to-dental-plan (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
bad
good
full
half
none
X
Y
Class colour
good bad


Weka Explorer: Visualizing labor
X: bereavement-assistance (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
yes
no
Class colour
good bad
Weka Explorer: Visualizing labor
X: contribution-to-health-plan (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: labor
full
half
none
Class colour
good bad

Part-1

1. Classification – Diabetes dataset
    a. Downloaded the dataset and dataset description.
    b. Installed the required packages and imported the Diabetes Dataset.

```
# Importing the required libraries and importing the dataset
library(readr)
library(FSelector)
library(party)
df1 = read_csv("diabetes.csv")
```

    c. Structure of the dataset:

```
# Dataset structure & summary
dim(df1)
View(df1)
```

[1] 768 9

No. of columns = 9

No. of rows = 768

Summary:

```
summary(df1)
```

preg	plas	pres	skin	insu	mass	pedi
Min. : 0.000	Min. : 0.0	Min. : 0.00	Min. : 0.00	Min. : 0.0	Min. : 0.00	Min. :0.0780
1st Qu.: 1.000	1st Qu.: 99.0	1st Qu.: 62.00	1st Qu.: 0.00	1st Qu.: 0.0	1st Qu.:27.30	1st Qu.:0.2437
Median : 3.000	Median :117.0	Median : 72.00	Median :23.00	Median : 30.5	Median :32.00	Median :0.3725
Mean : 3.845	Mean :120.9	Mean : 69.11	Mean :20.54	Mean : 79.8	Mean :31.99	Mean :0.4719
3rd Qu.: 6.000	3rd Qu.:140.2	3rd Qu.: 80.00	3rd Qu.:32.00	3rd Qu.:127.2	3rd Qu.:36.60	3rd Qu.:0.6262
Max. :17.000	Max. :199.0	Max. :122.00	Max. :99.00	Max. :846.0	Max. :67.10	Max. :2.4200

age	classlbl
Min. :21.00	Length:768
1st Qu.:24.00	Class :character
Median :29.00	Mode :character
Mean :33.24	
3rd Qu.:41.00	
Max. :81.00	

    d. Class variable: classlbl

Class variable is telling if a person has diabetes or not.

positive: has diabetes

negative: doesn't have diabetes

Converted the class variable as factor.

```
# Converting the class variable as factor
df1$classlbl = as.factor(df1$classlbl)
table(df1$classlbl)
prop.table(table(df1$classlbl))
```

    e. After splitting,

```
# Make this example reproducible
set.seed(1)
```

```
# Use 75% of dataset as training set and remaining 25% as testing set
sample1 = sample(c(TRUE, FALSE), nrow(df1), replace=TRUE, prob=c(0.8,0.2))
train1 = df1[sample1, ]
test1 = df1[!sample1, ]
dim(train1)
dim(test1)
```

Train data dimension: (621, 9)

Test data dimension: (147, 9)

f. Information gain on the training set:

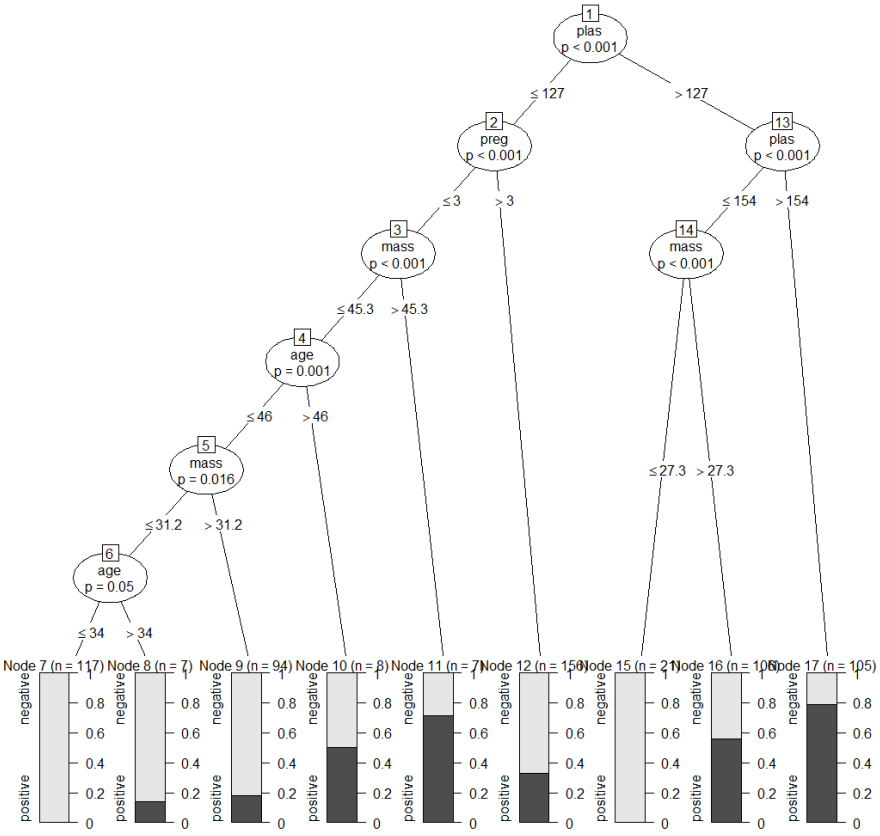
```
# Computing Information Gain
gain.ratio(classlbl~., train1)
```

	attr_importance
preg	0.06349480
plas	0.09809795
pres	0.00000000
skin	0.00000000
insu	0.04575926
mass	0.09910313
pedi	0.00000000
age	0.06304881

Best 4 variables: mass, plas, preg, age

g. Plotting the decision tree:

```
# Creating the tree model on best 4 variables
formula1=classlbl~mass+plas+preg+age
diabetes_tree = ctree(formula1, data=train1)
plot(diabetes_tree)
```



h. Explanation on the tree:
If plas<=127, mass<=45.3 and ag>46; then there is higher chance of getting diabetes.
Again, if plas<=127 and mass>45.3; there is a high chance of getting diabetes.
On the other side, if plas>127, plas<=154 and mass<=27.3; there is a verry little chance of getting diabetes.
Etc.

i. Stored the predictions on test data using the tree model.

j. Confusion matrix:

```
# Model performance
pred1 = predict(diabetes_tree, newdata=test1)
table(pred1, test1$classlbl)
```

pred1	negative	positive
negative	84	20
positive	15	28

Accuracy of the model = 76.19%
TPR = 0.65
FPR = 0.19
FNR = 0.35

2. Classification – Heart Attack dataset
    a. Downloaded the dataset.
    b. Imported the dataset into RStudio.

```
# Importing the dataset
df2 <- read_csv("Heart_Attack.csv")
```

    c. Structure of the dataset:

```
# Dataset structure & summary
View(df2)
dim(df2)
```

[1] 303 14

No. of columns = 14

No. of rows = 303

Datatype of the variables:

```
sapply(df2, class)
```

age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng
"numeric"	"numeric"	"numeric"	"numeric"	"numeric"	"numeric"	"numeric"	"numeric"	"numeric"
oldpeak	**slp**	**caa**	**thall**	**output**				
"numeric"	"numeric"	"numeric"	"numeric"	"numeric"				

    d. Column names:

```
names(df2)
```

[1] "age" "sex" "cp" "trtbps" "chol" "fbs" "restecg" "thalachh"
[9] "exng" "oldpeak" "slp" "caa" "thall" "output"

    e. Sum of the missing variables:

```
# Missing value handling
colSums(is.na(df2))
```

age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp
0	1	0	0	0	10	3	0	22	1	0
caa	**thall**	**output**								
13	0	0								

    f. Replaced the missing values with 0.

```
df2[is.na(df2)] = 0
```

After replacing, sum of the missing variables:

```
colSums(is.na(df2))
```

age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp
0	0	0	0	0	0	0	0	0	0	0
caa	**thall**	**output**								
0	0	0								

    g. Plots:

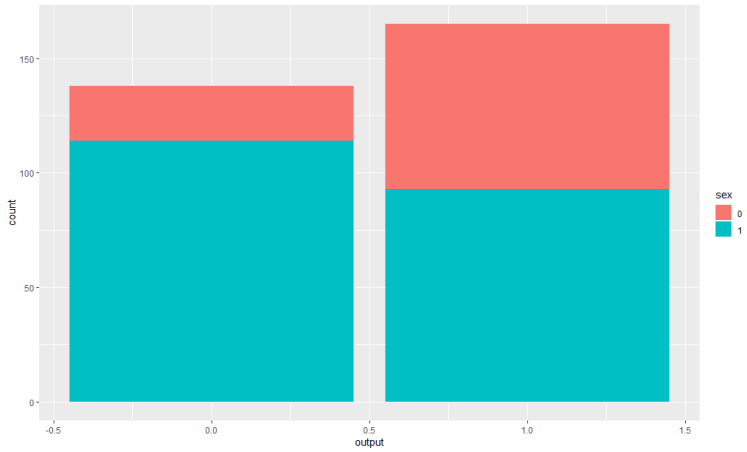
Correlation plot between the continuous features:

```
# Correlation plot between the continuous features
library(ggcorrplot)
library(ggplot2)
con_feat = df2[,c(1,4,5,8,10)]
corr = round(cor(con_feat), 2)
```

```
ggcorrplot(corr,hc.order = TRUE, outline.col = "white", lab=TRUE)
```

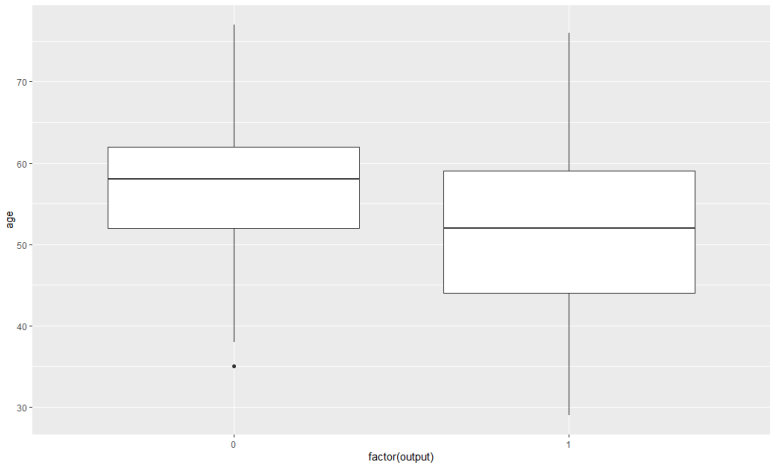
Barplot of the output with the portion of each count that is from each sex:

```
# Barplot of the output with the portion of each count that is from each sex
df2$sex = as.factor(df2$sex)
ggplot(data = df2, aes(x = output, fill = sex)) + geom_bar()
```



Box plots of age, grouped by output:

```
# Box plots of age, grouped by output
ggplot(data = df2, aes(x = factor(output), y = age)) + geom_boxplot()
```



h. Class variable: output
This attribute is considered as class variable because it consists of 0s and 1s only, where 1: has heart disease, 0: doesn't have heart disease.

i. Converted the class variable as factor.

```
# Converting the class variable as factor
df2$output = as.factor(df2$output)
table(df2$output)
prop.table(table(df2$output))
```

j. After splitting the dataset:

```
# Make this example reproducible
set.seed(1)

# Use 80% of dataset as training set and remaining 20% as testing set
sample2 = sample(c(TRUE, FALSE), nrow(df2), replace=TRUE, prob=c(0.8,0.2))
train2 = df2[sample2, ]
test2 = df2[!sample2, ]
dim(train2)
dim(test2)
```

Train data dimension: (254, 14)
Test data dimension: (49, 14)

k. Information gain on the training set:

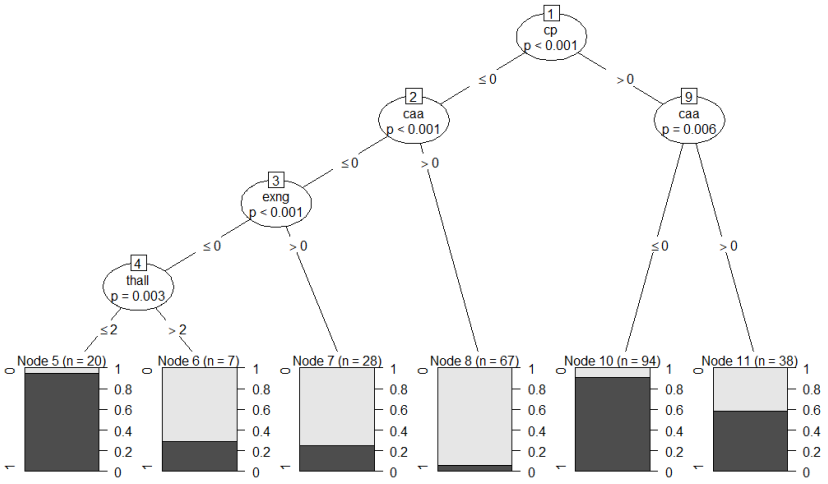
```
# Computing Information Gain
gain.ratio(output~., train2)
```

	attr_importance
age	0.07812208
sex	0.05578433
cp	0.23117160
trtbps	0.00000000
chol	0.00000000
fbs	0.00000000
restecg	0.00000000
thalachh	0.12629630
exng	0.18399344
oldpeak	0.14424141
slp	0.10319972
caa	0.19606674
thall	0.17449078

Top 4 variables: cp, exng, caa, thall

l. Plotting the decision tree:

```
# Creating the tree model on best 4 variables
formula2=output~cp+caa+exng+thall
hrtattck_tree = ctree(formula2, data=train2)
plot(hrtattck_tree)
```



m. Explanation on the tree:
   If cp<=0, caa<=0, exng<=0 and thall<=2; there is a high chance of getting a heart attack.
   On the other side, if cp<=0 and caa>0; there is a very low chance of getting a heart attack.
   Etc.
n. Stored the predictions on the test data using the decision tree.
o. Confusion matrix:

```
# Model performance
pred2 = predict(hrtattck_tree, newdata=test2)
table(pred2, test2$output)
```

predictions	0	1
0	13	4
1	10	22

Accuracy of the model = 71.43%
TPR = 0.69
FPR = 0.24
FNR = 0.31

Overall Code for Part-1

```
# Names and ID

# Mozoon Ahmed Alharbi 201905074

# Alya Ahmed Khadim 201902453

# Raysa Rashed Abduljalil 201916402



## Q1: Diabetes dataset

# Importing the required libraries and importing the dataset
```

```
library(readr)
library(FSelector)
library(party)
df1 = read_csv("diabetes.csv")

# Dataset structure & summary
dim(df1)
View(df1)
summary(df1)

# Converting the class variable as factor
df1$classlbl = as.factor(df1$classlbl)
table(df1$classlbl)
prop.table(table(df1$classlbl))

# Make this example reproducible
set.seed(1)

# Use 75% of dataset as training set and remaining 25% as testing set
sample1 = sample(c(TRUE, FALSE), nrow(df1), replace=TRUE, prob=c(0.8,0.2))
train1 = df1[sample1, ]
test1 = df1[!sample1, ]
dim(train1)
dim(test1)

# Computing Information Gain
gain.ratio(classlbl~., train1)

# Creating the tree model on best 4 variables
formula1=classlbl~mass+plas+preg+age
diabetes_tree = ctree(formula1, data=train1)
```

```
plot(diabetes_tree)

# Model performance
pred1 = predict(diabetes_tree, newdata=test1)
table(pred1, test1$classlbl)



## Q2: Heart Attack dataset
# Importing the dataset
df2 <- read_csv("Heart_Attack.csv")

# Dataset structure & summary
View(df2)
dim(df2)
sapply(df2, class)
names(df2)

# Missing value handling
colSums(is.na(df2))
df2[is.na(df2)] = 0
colSums(is.na(df2))

# Plots
# Correlation plot between the continuous features
library(ggcorrplot)
library(ggplot2)
con_feat = df2[,c(1,4,5,8,10)]
corr = round(cor(con_feat), 2)
ggcorrplot(corr,hc.order = TRUE, outline.col = "white", lab=TRUE)
# Barplot of the output with the portion of each count that is from each sex
df2$sex = as.factor(df2$sex)
```

```
ggplot(data = df2, aes(x = output, fill = sex)) + geom_bar()
# Box plots of age, grouped by output
ggplot(data = df2, aes(x = factor(output), y = age)) + geom_boxplot()

# Converting the class variable as factor
df2$output = as.factor(df2$output)
table(df2$output)
prop.table(table(df2$output))

# Make this example reproducible
set.seed(1)

# Use 80% of dataset as training set and remaining 20% as testing set
sample2 = sample(c(TRUE, FALSE), nrow(df2), replace=TRUE, prob=c(0.8,0.2))
train2 = df2[sample2, ]
test2 = df2[!sample2, ]
dim(train2)
dim(test2)

# Computing Information Gain
gain.ratio(output~., train2)

# Creating the tree model on best 4 variables
formula2=output~cp+caa+exng+thall
hrtattck_tree = ctree(formula2, data=train2)
plot(hrtattck_tree)

# Model performance
pred2 = predict(hrtattck_tree, newdata=test2)
table(pred2, test2$output)
```

Part-2

2. Linear Regression – 50_Startups dataset:

a. Downloaded the dataset.

b. Imported the dataset into RStudio.

```
# Importing the dataset
library(readr)
df <- read_csv("50_Startups.csv")
```

c. Structure of the dataset:

```
# Dataset info
dim(df)
[1] 50  5
```

No. of columns = 5

No. of rows = 50

Datatypes of the variables:

```
# State variable
table(df$State)
```

R&D Spend	Administration	Marketing Spend	State	Profit
"numeric"	"numeric"	"numeric"	"character"	"numeric"

d. Class variable: State

It gives the names of the states.

e. Number of 'State' variable in the dataset:

```
California    Florida   New York
        17         16         17
```

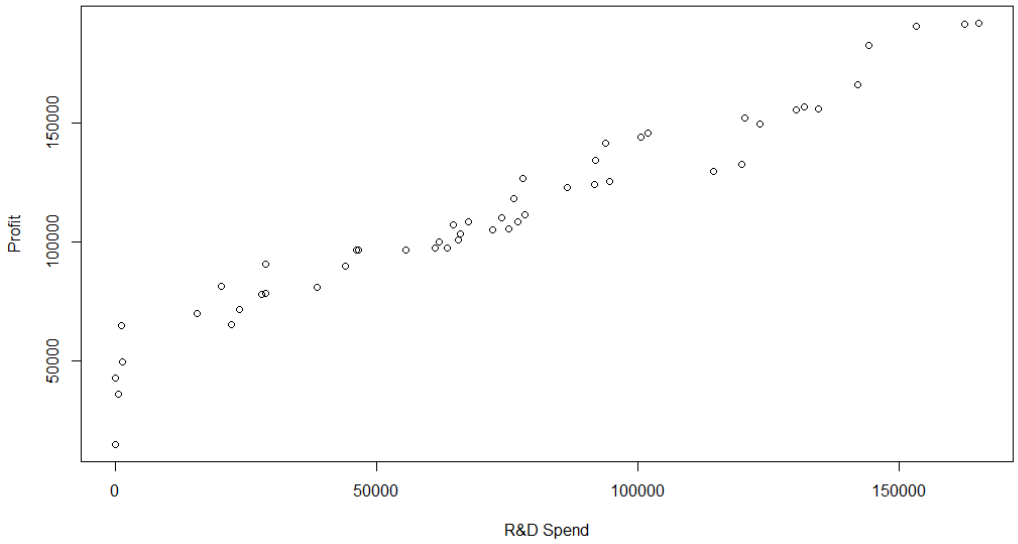
f. Here the column 'State' is needed to be converted as factor so that it can be used in the model.

```
# Converting the class variable as factor
df$State = as.factor(df$State)
```

g. Scatter plots:

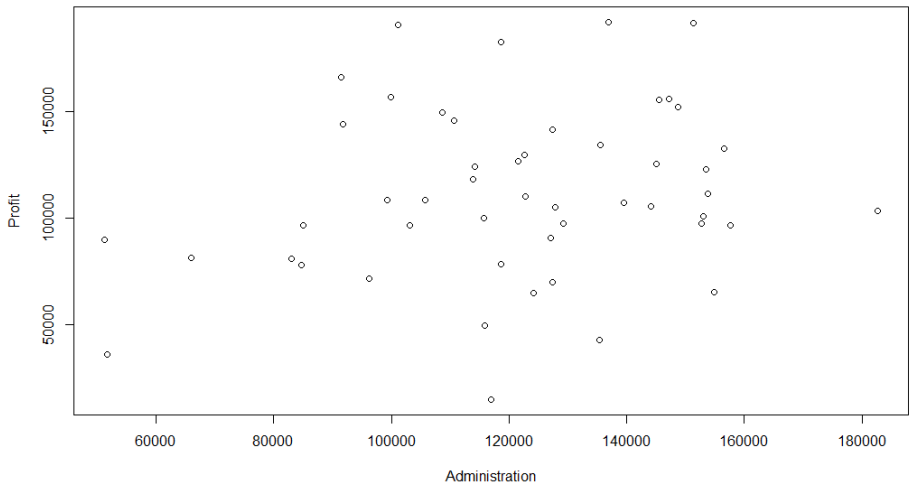
```
# Scatter plots
attach(df)
plot(`R&D Spend`, Profit)
plot(Administration, Profit)
plot(`Marketing Spend`, Profit)
```

Profit vs R&D Spend:

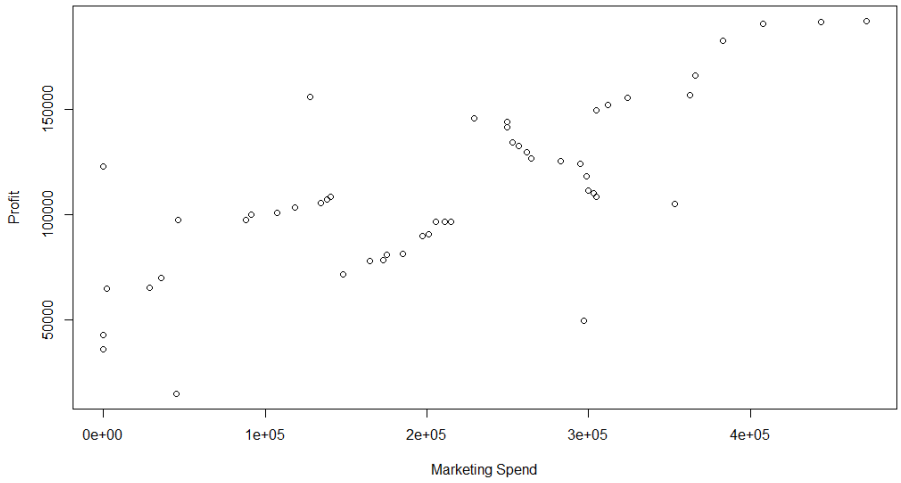


This plot is showing that there is a positive correlation between Profit and R&D Spend.

Profit vs Administration:



This plot is showing that there is not much relationship between Profit and Administration.
Profit vs Marketing Spend:



This plot is telling that there is a positive correlation between Profit and Marketing Spend.

h. The variable 'Administration' doesn't seem to have good correlation with Profit, so we may drop that column.

i. Correlations between the variables:

```
# Correlations
cor(df[,c(1,2,3,5)])
```

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1	0.241955	0.724248	0.972901
Administration	0.241955	1	-0.03215	0.200717
Marketing Spend	0.724248	-0.03215	1	0.747766
Profit	0.972901	0.200717	0.747766	1

We encounter positive correlations between Profit-R&D Spend, Profit-Administration, Profit-Marketing Spend, Administration-R&D Spend, Marketing Spend-R&D Spend and negative correlations between Marketing Spend-Administration.

j. Built the Linear Regression Model after dropping the variable 'Administration'.

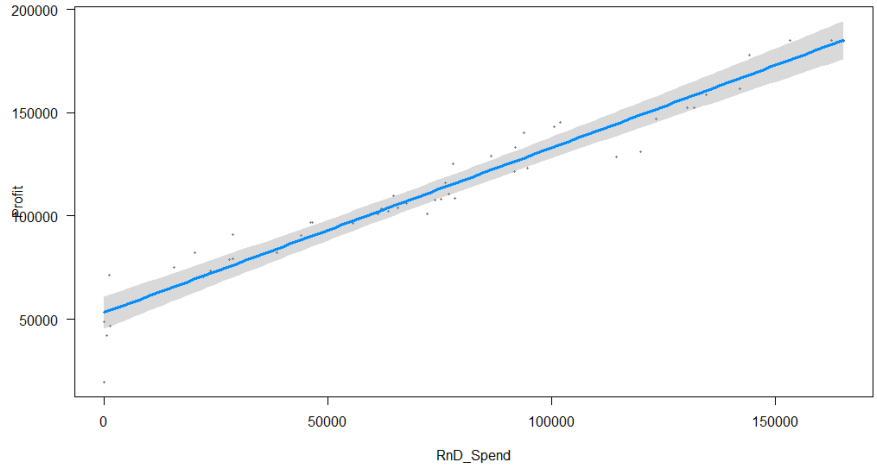
```
# Building the linear model
names(df) = c('RnD_Spend', 'Administration', 'Marketing_Spend', 'State', 'Profit')
attach(df)
fitted_model = lm(Profit~RnD_Spend+Marketing_Spend+State)
fitted_model
```

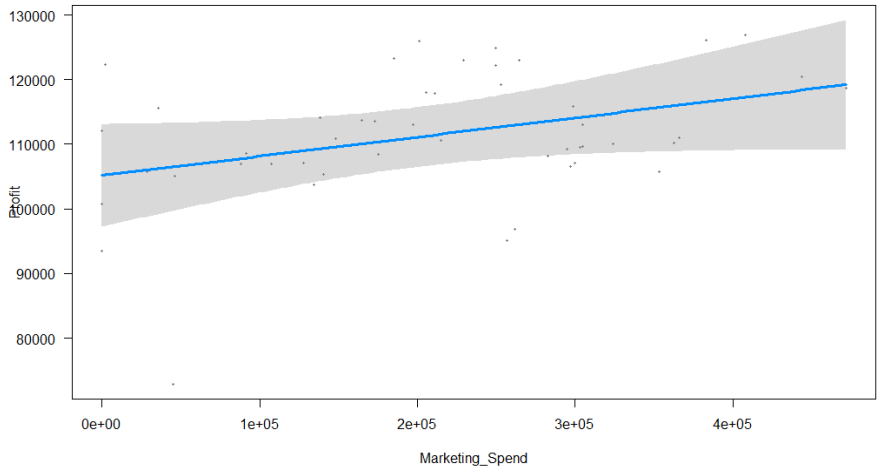
k. Output of the model:

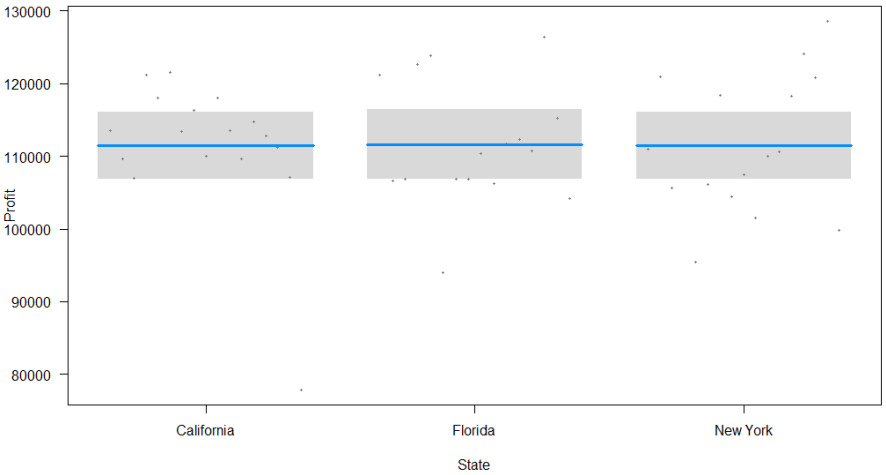
```
Call:
lm(formula = Profit ~ RnD_Spend + Marketing_Spend + State)

Coefficients:
     (Intercept)         RnD_Spend   Marketing_Spend      StateFlorida   StateNew York
       4.696e+04         7.967e-01         2.975e-02         1.408e+02        -1.952e+01
```

l. Model visualization:

```
# Visualizing the model
library(visreg)
visreg(fitted_model)
```







Overall Code for Part-2

Names and ID

```
# Mozoon Ahmed Alharbi 201905074
# Alya Ahmed Khadim 201902453
# Raysa Rashed Abduljalil 201916402


## Q2: 50_Startups Dataset
# Importing the dataset
library(readr)
df <- read_csv("50_Startups.csv")

# Dataset info
dim(df)
sapply(df, class)

# State variable
table(df$State)

# Converting the class variable as factor
df$State = as.factor(df$State)

# Scatter plots
attach(df)
plot(`R&D Spend`, Profit)
plot(Administration, Profit)
plot(`Marketing Spend`, Profit)

# Correlations
cor(df[,c(1,2,3,5)])

# Building the linear model
names(df) = c('RnD_Spend', 'Administration', 'Marketing_Spend', 'State', 'Profit')
```

```
attach(df)
fitted_model = lm(Profit~RnD_Spend+Marketing_Spend+State)
fitted_model

# Visualizing the model
library(visreg)
visreg(fitted_model)
```

1. Downloaded the ‘glass.arff’ dataset from Weka dataset folder from BB.
2. Reason for choosing the dataset:
    a. The dataset has multiple classes in its target variable.
    b. The dataset has multiple decision variables.
    c. The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified!
3. Dataset’s attribute information:

    1. Id number: 1 to 214

    2. RI: refractive index

    3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)

    4. Mg: Magnesium

    5. Al: Aluminum

    6. Si: Silicon

    7. K: Potassium

    8. Ca: Calcium

    9. Ba: Barium

    10. Fe: Iron

    11. Type of glass: (class attribute)

    -- 1 building_windows_float_processed

    -- 2 building_windows_non_float_processed

    -- 3 vehicle_windows_float_processed

    -- 4 vehicle_windows_non_float_processed (none in this database)

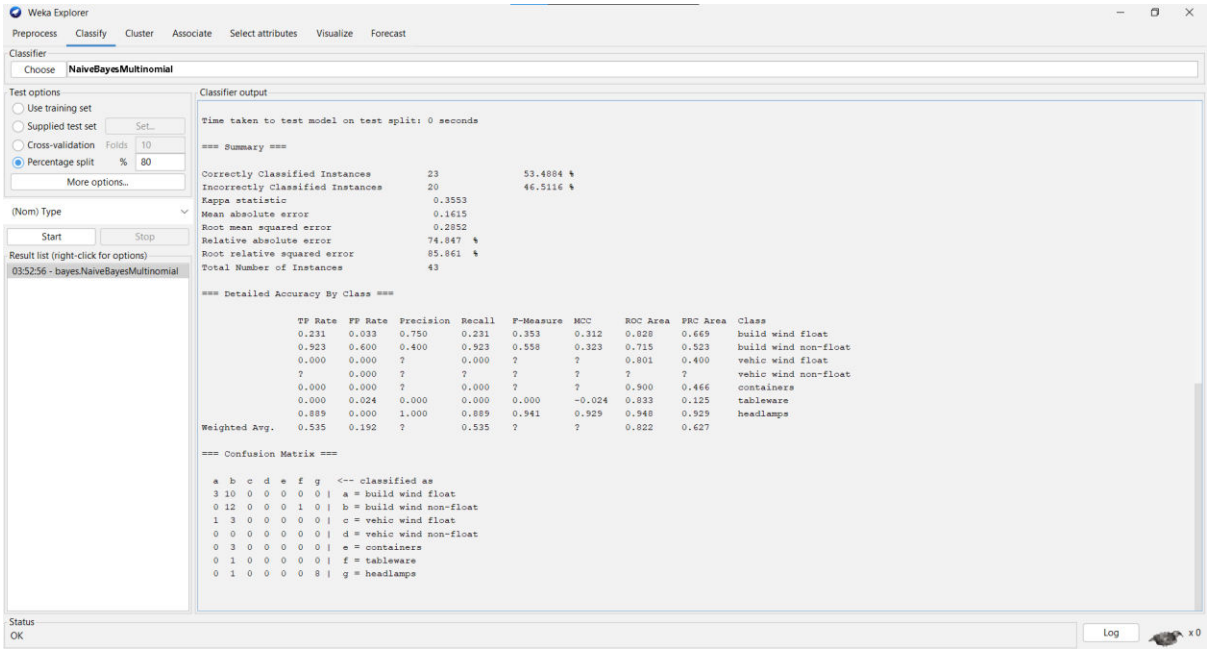
    -- 5 containers

    -- 6 tableware

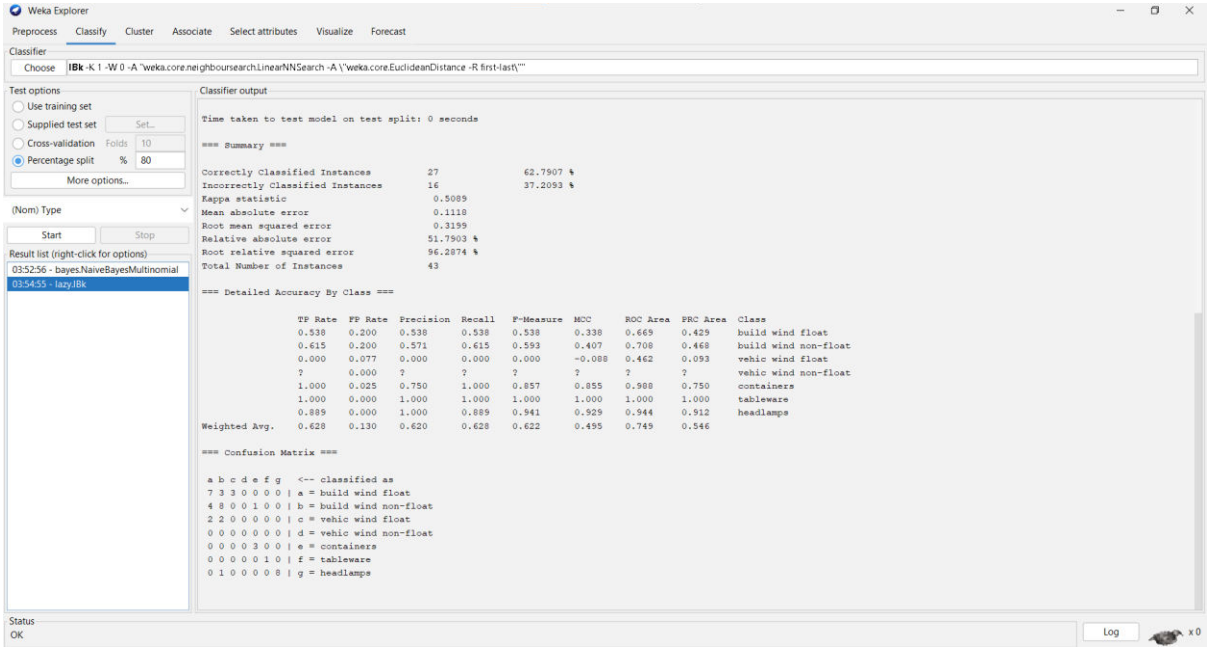
    -- 7 headlamps

4. Different classification models:

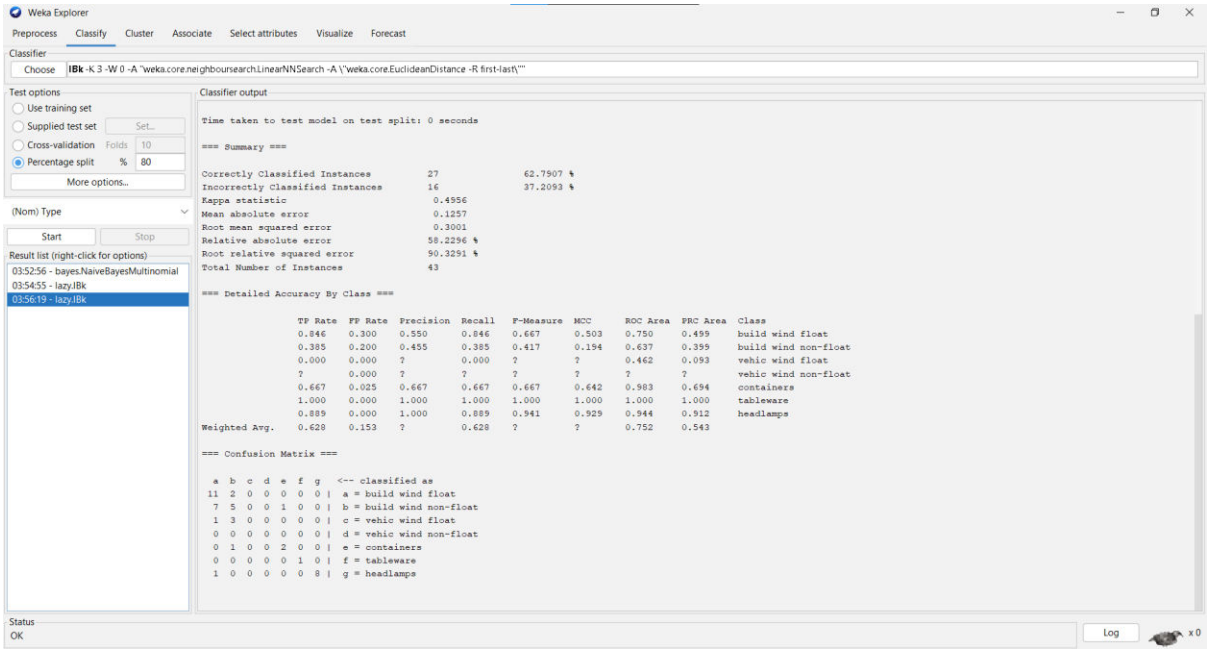
    NaiveBayesMultinomial (Train-test split: 80%-20%)



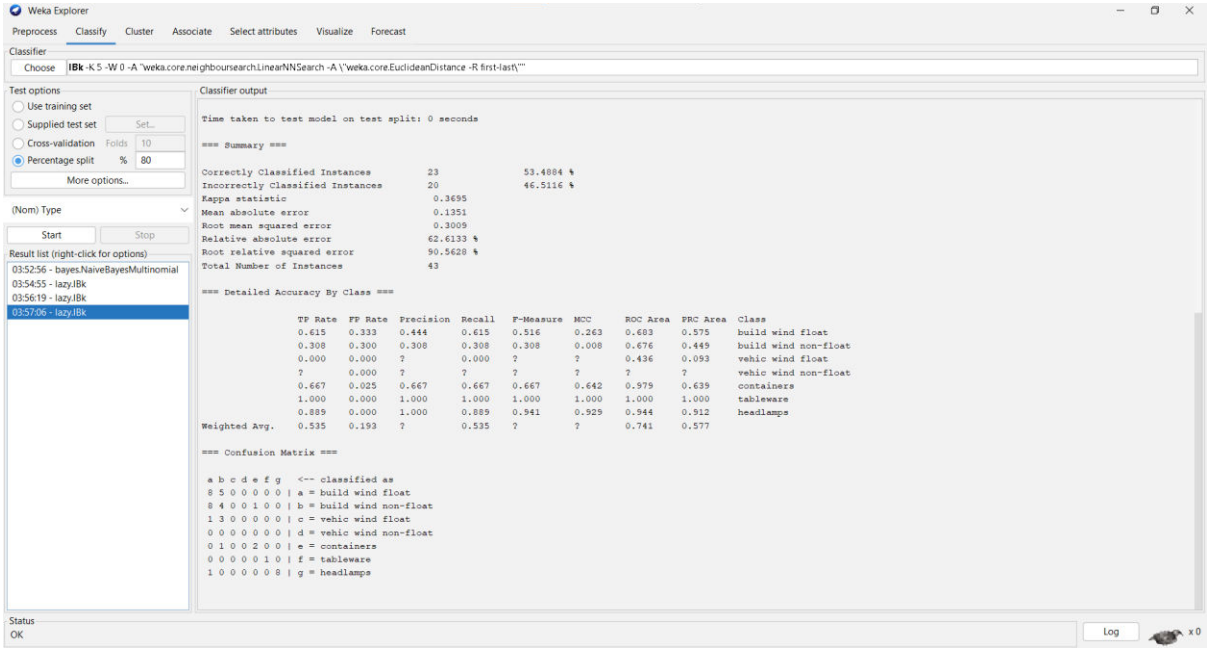
KNN (K=1) (Train-test split: 80%-20%)



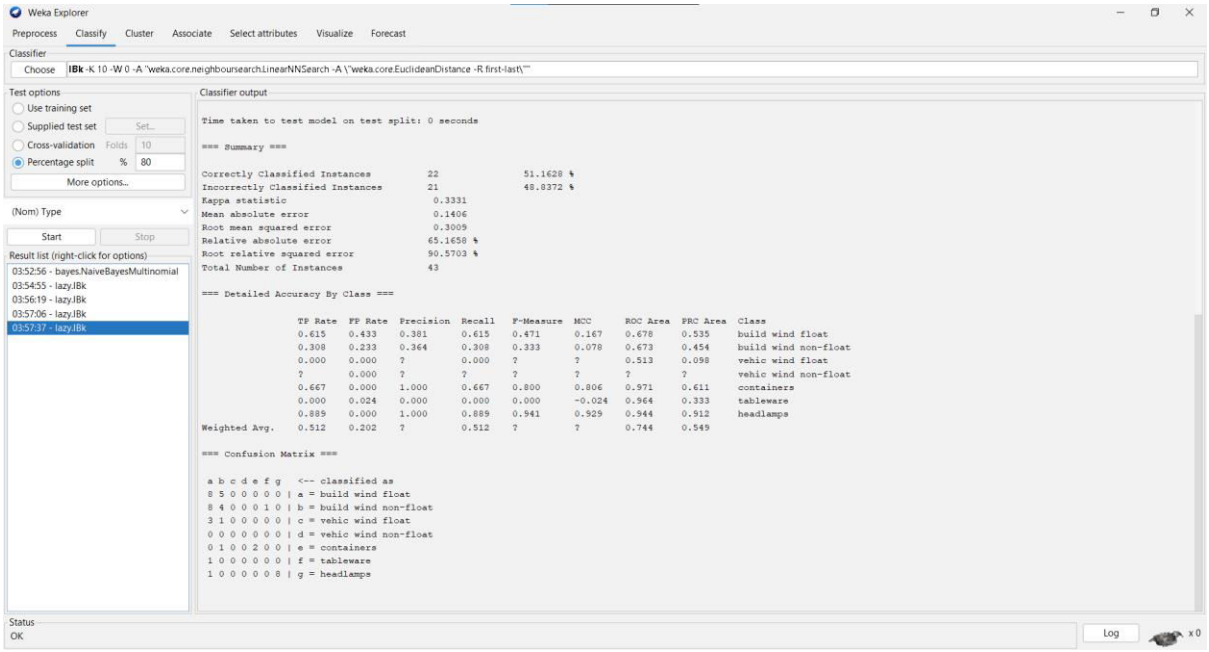
KNN (K=3) (Train-test split: 80%-20%)



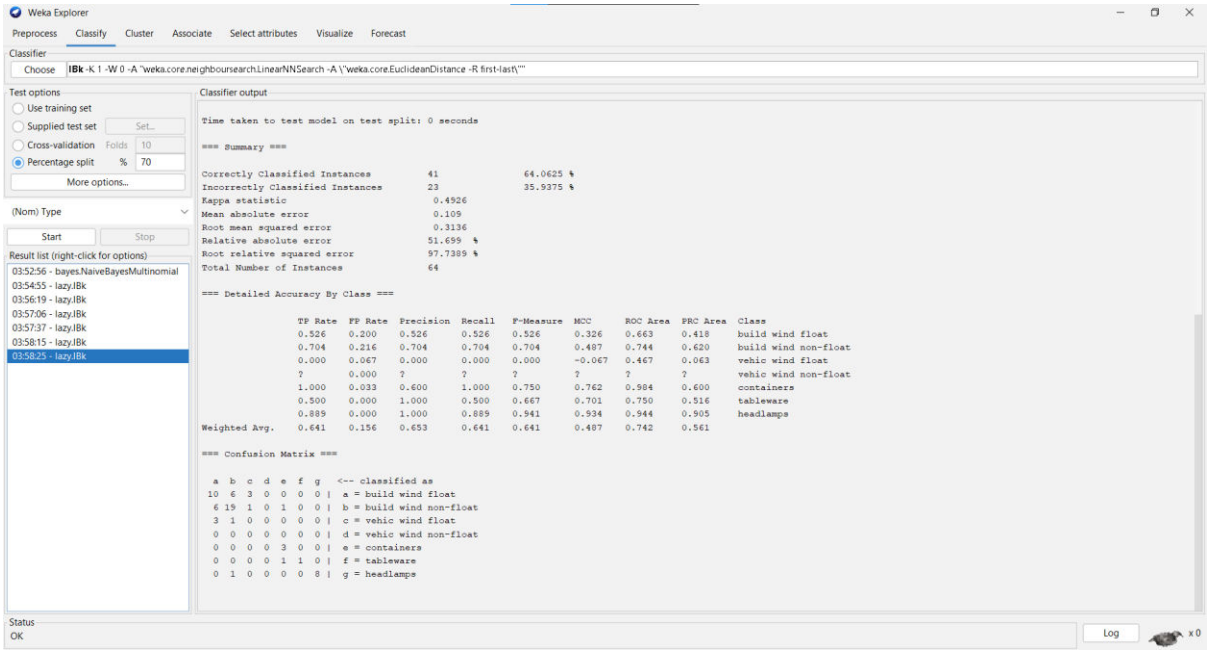
KNN (K=5) (Train-test split: 80%-20%)



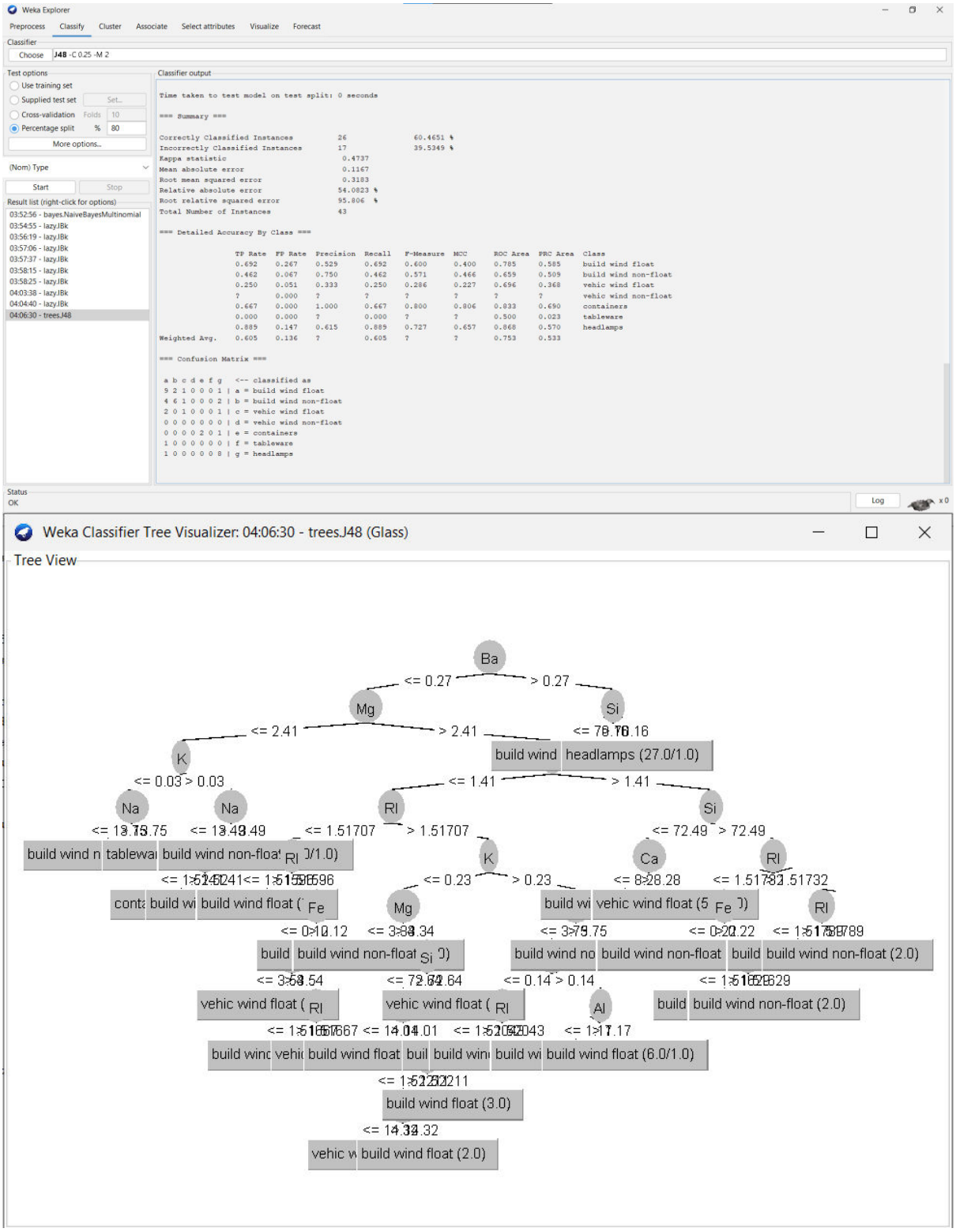
KNN (K=10) (Train-test split: 80%-20%)



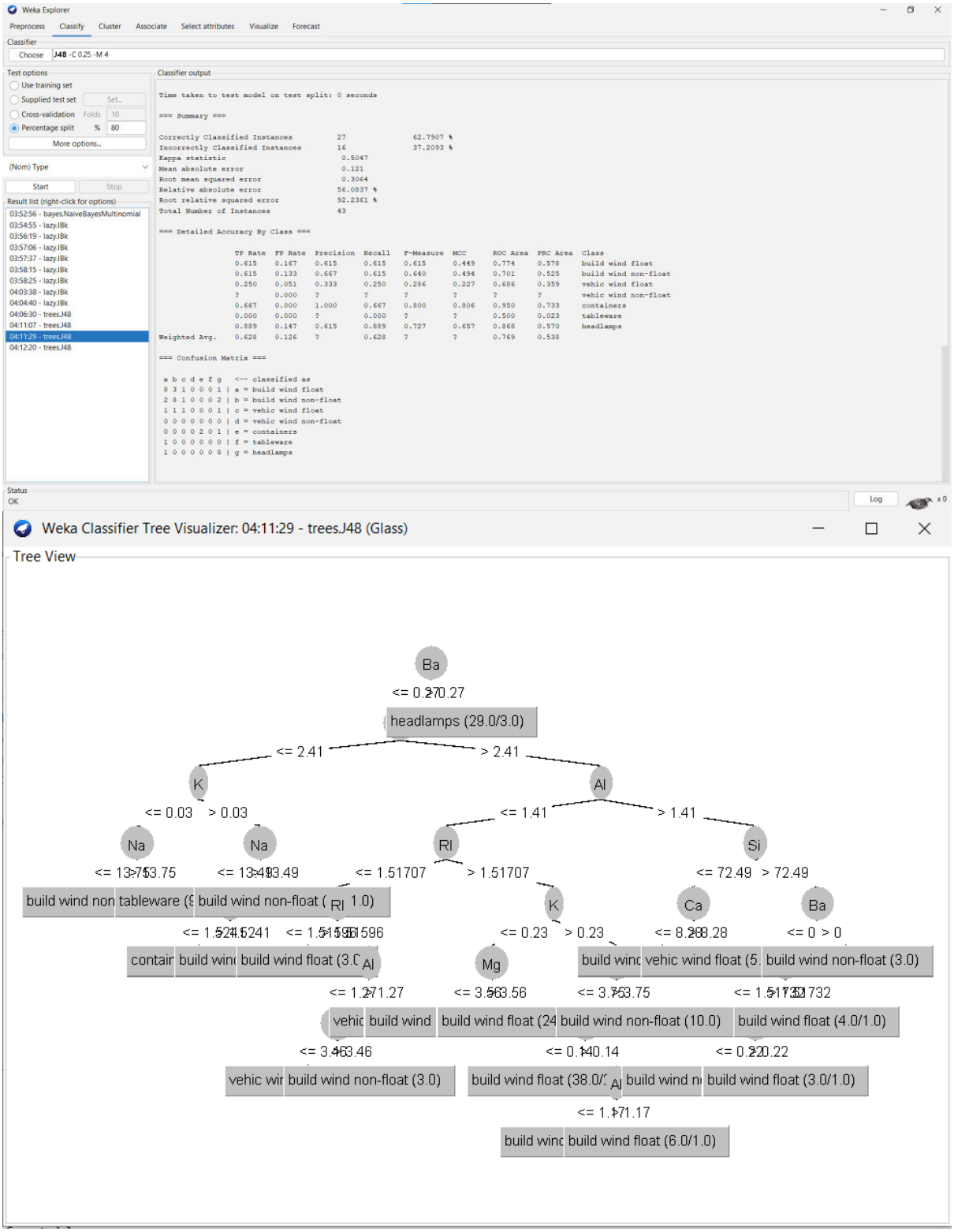
KNN (K=1) (Train-test split: 70%-30%)



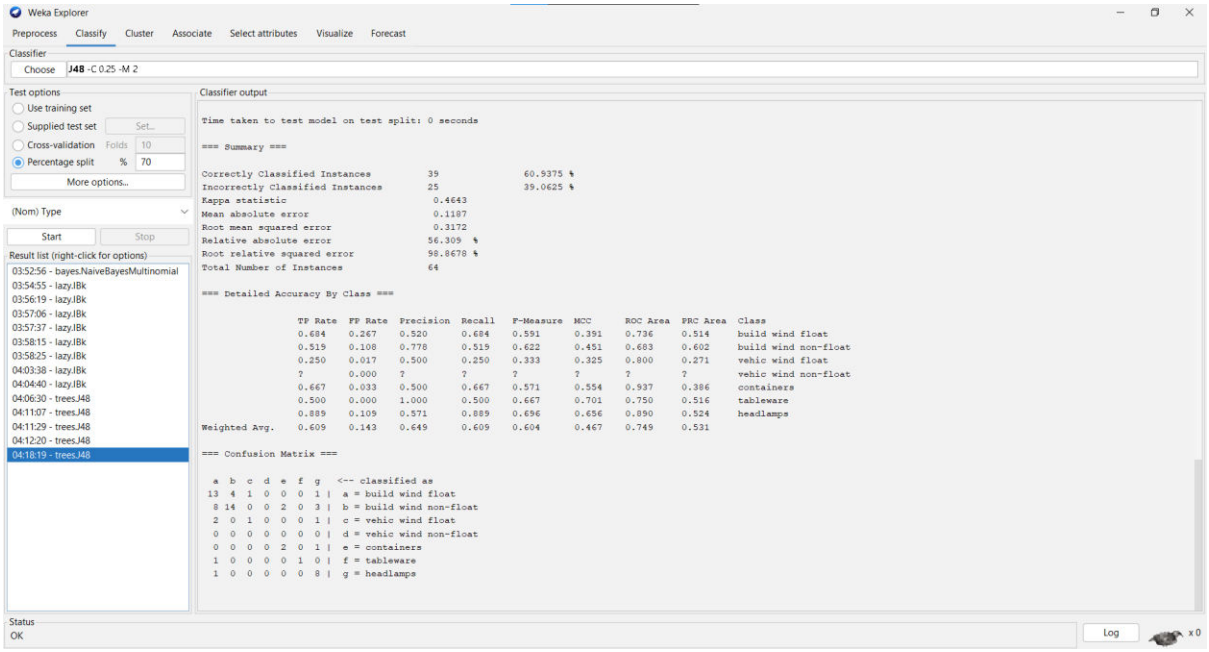
Decision Tree (minNumObj = 2) (Train-test split: 80%-20%)

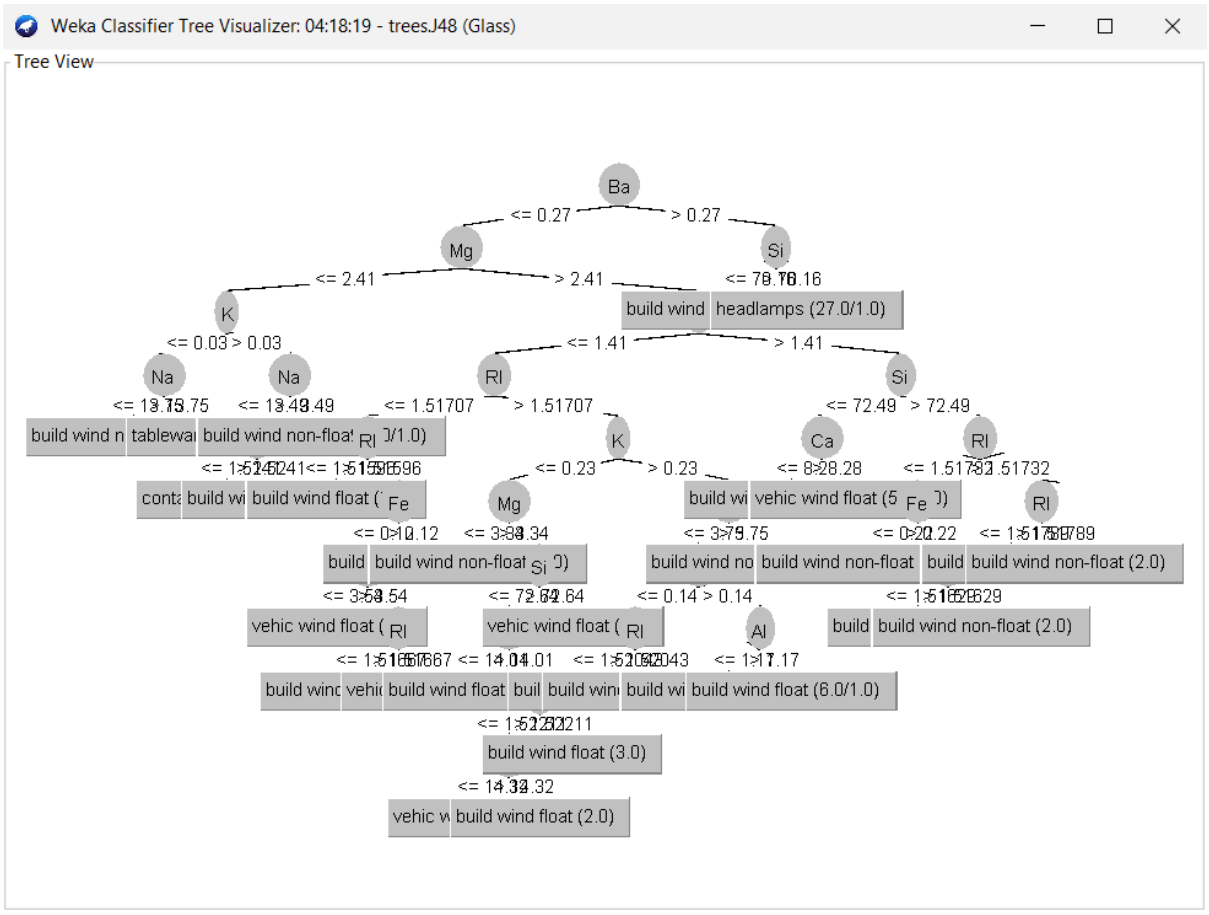


Decision Tree (minNumObj = 3) (Train-test split: 80%-20%)

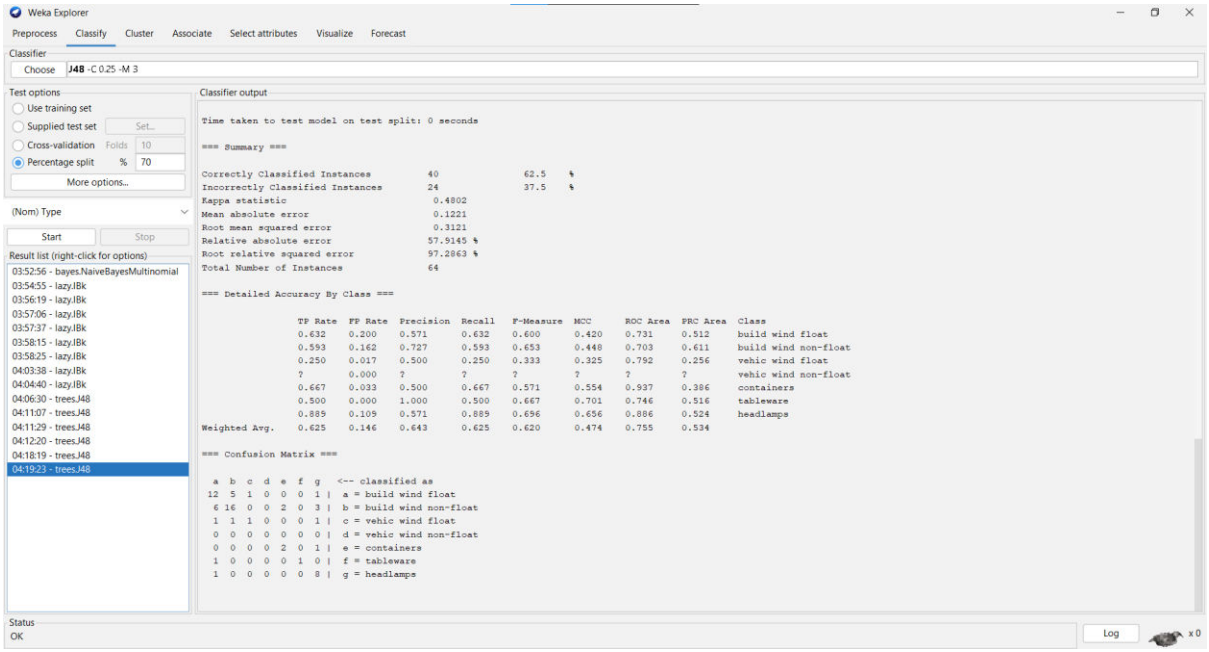


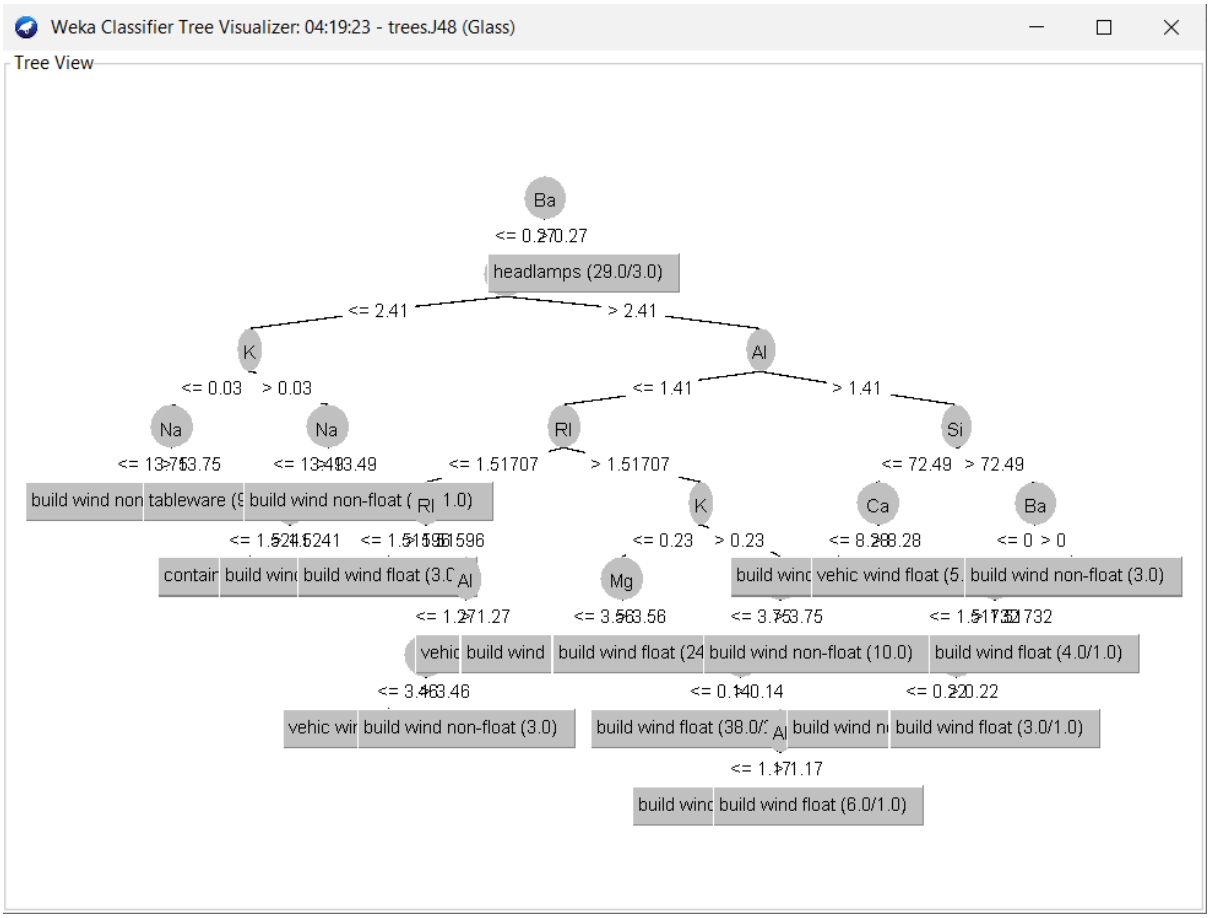
Decision Tree (minNumObj = 2) (Train-test split: 70%-30%)



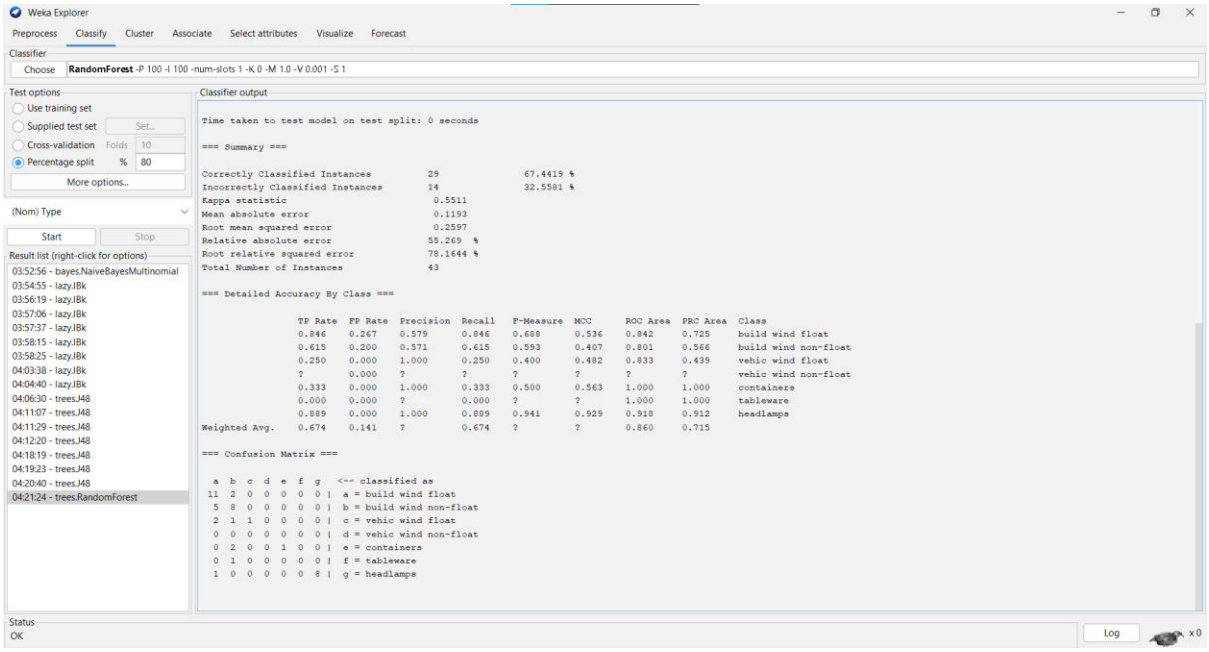


Decision Tree (minNumObj = 3) (Train-test split: 70%-30%)

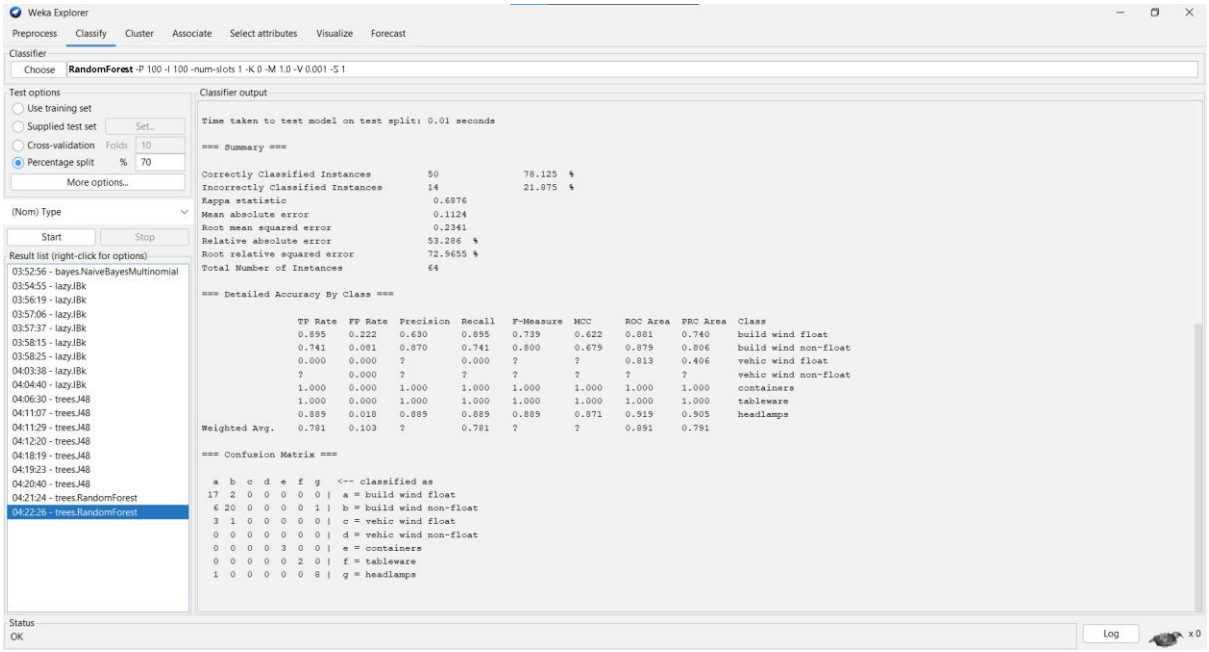




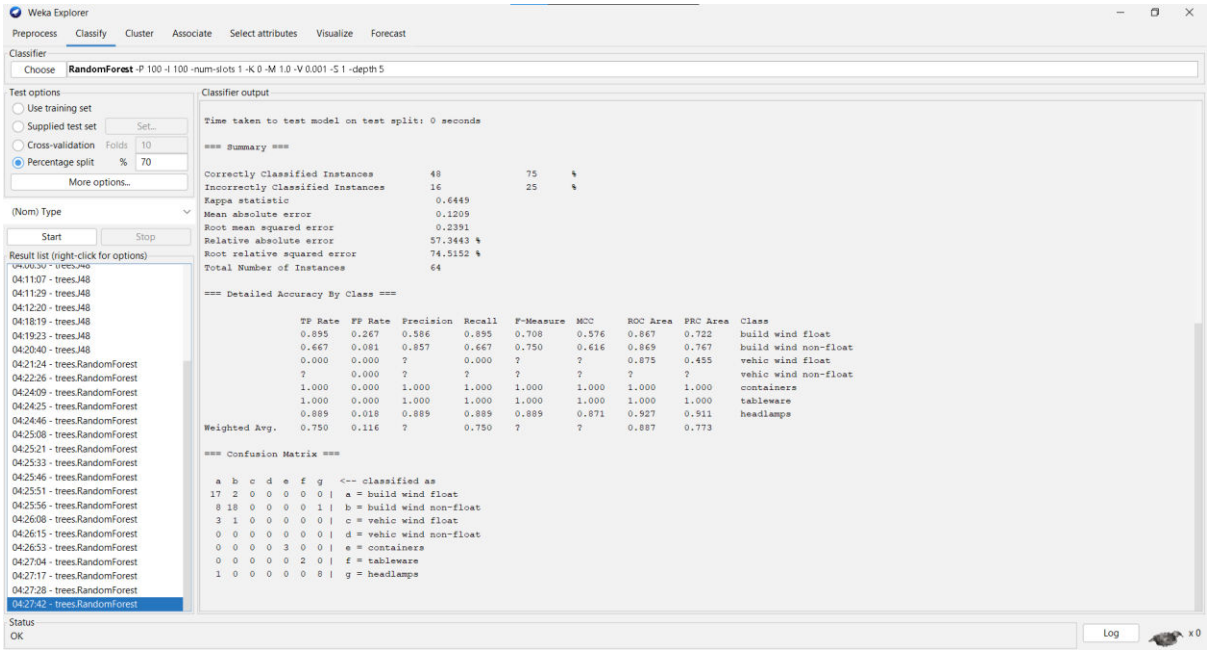
Random Forest (Default) (Train-test split: 80%-20%)



Random Forest (Default) (Train-test split: 70%-30%)



Random Forest (maxDepth=5) (Train-test split: 70%-30%)



Performance Comparison (The model with the highest accuracy is highlighted in the table):

The Name of the Machine Learning Technique	Accuracy percentage
Naïve Bayes Multinomial (Train-Test split 80:20)	53.49%
KNN (k=1) (Train-Test split 80:20)	62.79%
KNN (k=3) (Train-Test split 80:20)	62.79%
KNN (k=5) (Train-Test split 80:20)	53.49%
KNN (k=10) (Train-Test split 80:20)	51.16%
KNN (k=1) (Train-Test split 70:30)	64.06%
Decision Tree (minNumObj = 2) (Train-Test split 80:20)	60.47%
Decision Tree (minNumObj = 3) (Train-Test split 80:20)	62.79%
Decision Tree (minNumObj = 2) (Train-Test split 70:30)	60.94%
Decision Tree (minNumObj = 3) (Train-Test split 70:30)	62.50%
Random Forest (default) (Train-Test split 80:20)	67.44%
Random Forest (default) (Train-Test split 70:30)	78.13%
Random Forest (maxDepth = 5) (Train-Test split 70:30)	75.00%

R Studio

1. **Apply "Linear Regression" analysis on the "Air quality" dataset.**
   a. Installed the needed packages & libraries.

```
# Importing the required libraries and importing the dataset
library(readr)
library(visreg)
```

   b. Downloaded the "air_quality" dataset from BB
   c. Loaded the dataset in RStudio.

```
df1 = read_csv("air_quality.csv")
```

   d. Get the structure of the dataset, summarize the dataset, and display the first few and last columns.

```
# Dataset structure & summary
dim(df1)
df1
```

[1] 153 6
No. of columns = 6
No. of rows = 153

```
Summary:
summary(df1)
```

Ozone	Solar.R	Wind	Temp	Month	Day
Min. : 1.00	Min. : 7.0	Min. : 1.700	Min. :56.00	Min. :5.000	Min. : 1.00
1st Qu.: 18.00	1st Qu.:115.8	1st Qu.: 7.400	1st Qu.:72.75	1st Qu.:6.000	1st Qu.: 8.50
Median : 31.50	Median :205.0	Median : 9.700	Median :79.00	Median :7.000	Median :16.00
Mean : 42.13	Mean :185.9	Mean : 9.983	Mean :77.95	Mean :6.993	Mean :15.99
3rd Qu.: 63.25	3rd Qu.:258.8	3rd Qu.:11.750	3rd Qu.:85.00	3rd Qu.:8.000	3rd Qu.:23.50
Max. :168.00	Max. :334.0	Max. :20.700	Max. :97.00	Max. :9.000	Max. :31.00
NA's :37	NA's :7	NA's :2	NA's :1		NA's :2

   e. Pre-process the dataset by finding which attributes have missing values. You can either exclude or replace the missing values to be able to build the regression model.

```
# Missing value handling
colSums(is.na(df1))
```

Ozone	Solar.R	Wind	Temp	Month	Day
37	7	2	1	0	2

```
df1 = na.omit(df1)
colSums(is.na(df1))
```

Ozone	Solar.R	Wind	Temp	Month	Day
0	0	0	0	0	0

f. Apply scaling and then show the first few columns of the dataset after scaling it.

```
# Scaling the dataset
df1 = scale(df1, center = TRUE, scale = TRUE)
df1 = as.data.frame(df1)
df1
```

This is how the first 5 rows look after scaling:

	Ozone	Solar.R	Wind	Temp	Month	Day
1	-0.05624	0.035784	-0.70735	-1.11847	-1.46885	-1.76465
2	-0.20517	-0.75641	-0.53997	-0.59994	-1.46885	-1.64737
3	-0.92009	-0.41532	0.743333	-0.39253	-1.46885	-1.5301
4	-0.74136	1.389108	0.436457	-1.63699	-1.46885	-1.41282
5	-0.59242	1.235071	-0.37258	-1.32588	-1.46885	-1.06098

g. Apply the Liner Regression algorithm.

```
# Linear Regression Model
model = lm(Ozone~., data = df1)
```

h. Show the model output.

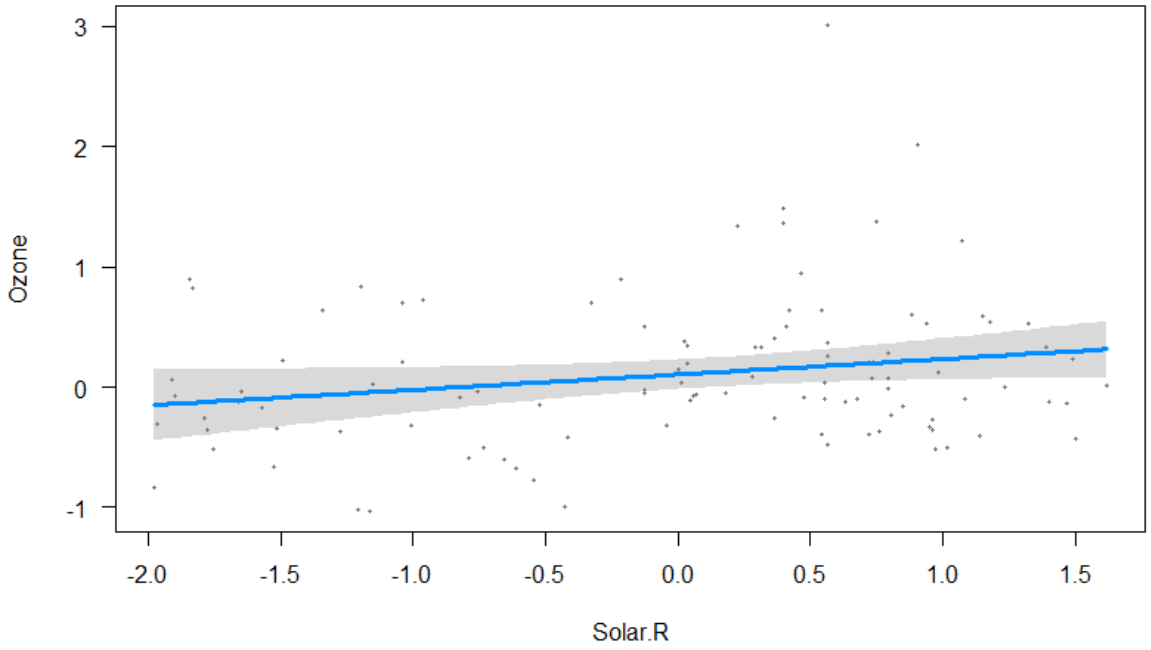
```
summary(model)
```

```
Call:

lm(formula = Ozone ~ ., data = df1)

Residuals:

   Min     1Q  Median     3Q    Max

-1.0406 -0.3988 -0.1142  0.2547  2.8255

Coefficients:

             Estimate Std. Error t value Pr(>|t|)

(Intercept)  2.666e-16  6.013e-02   0.000   1.0000

Solar.R      1.284e-01  6.580e-02   1.951   0.0538 .

Wind        -3.548e-01  6.987e-02  -5.078 1.75e-06 ***

Temp         5.474e-01  8.058e-02   6.793 7.75e-10 ***

Month       -1.200e-01  6.834e-02  -1.756   0.0822 .

Day          6.397e-02  6.089e-02   1.051   0.2959

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.622 on 101 degrees of freedom

Multiple R-squared:  0.6313,      Adjusted R-squared:  0.6131
```

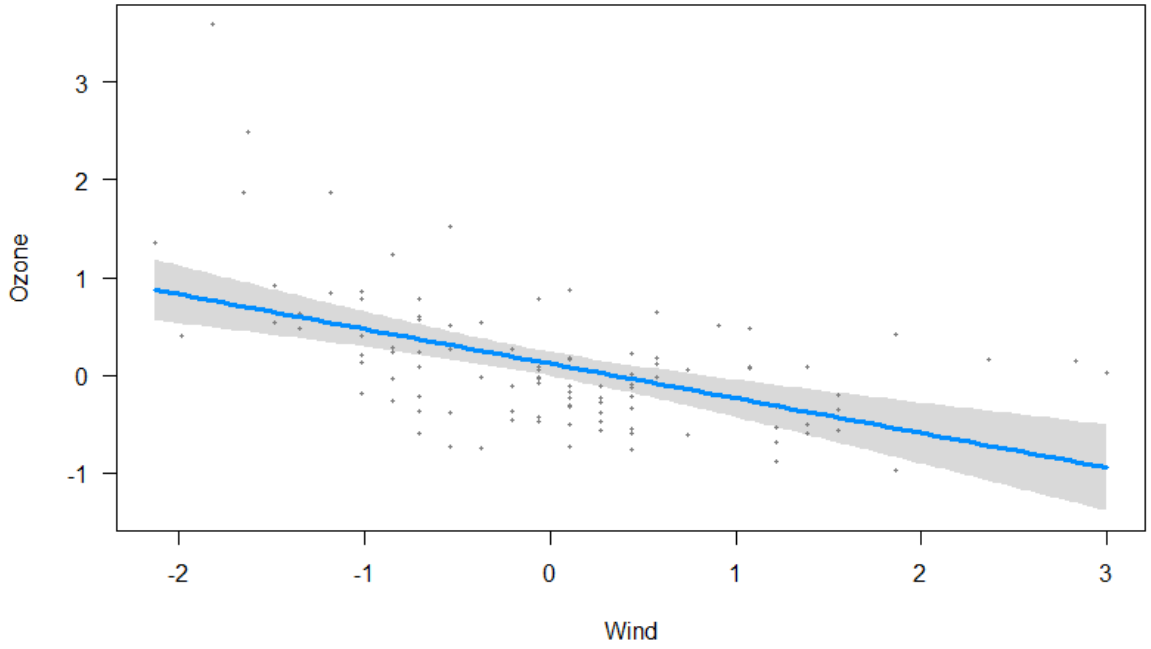
F-statistic: 34.59 on 5 and 101 DF, p-value: < 2.2e-16

i. Show your model (visualization) and explain it.

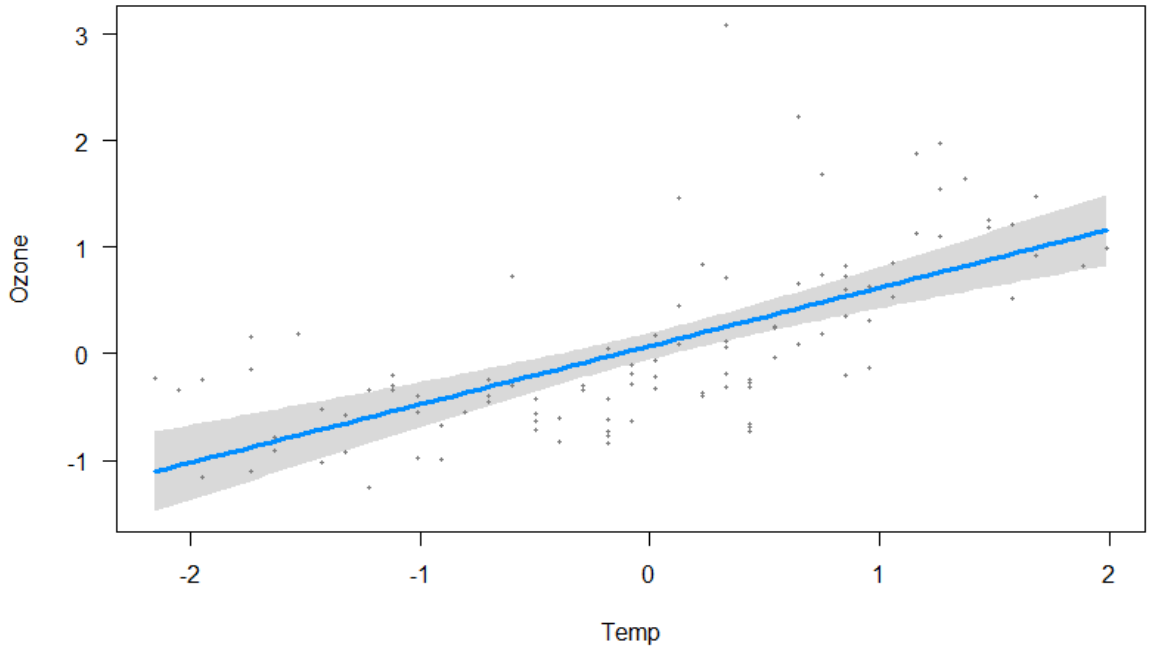
```
# Model Visualization
visreg(model)
```



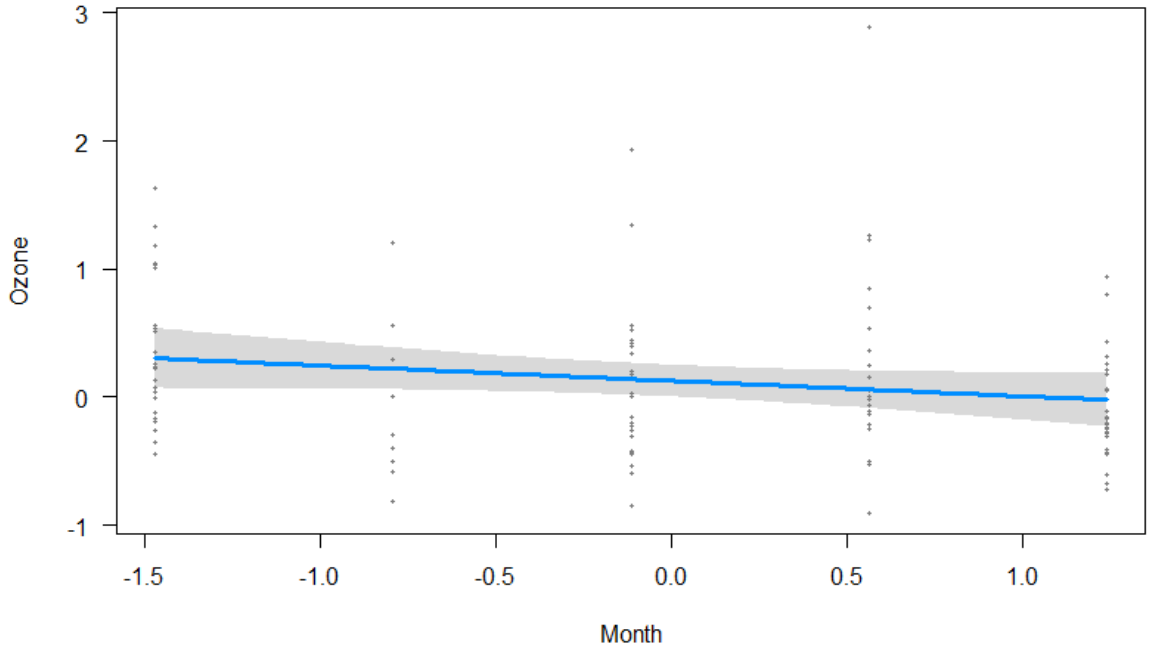
The Solar.R variable has very less correlation with ozone



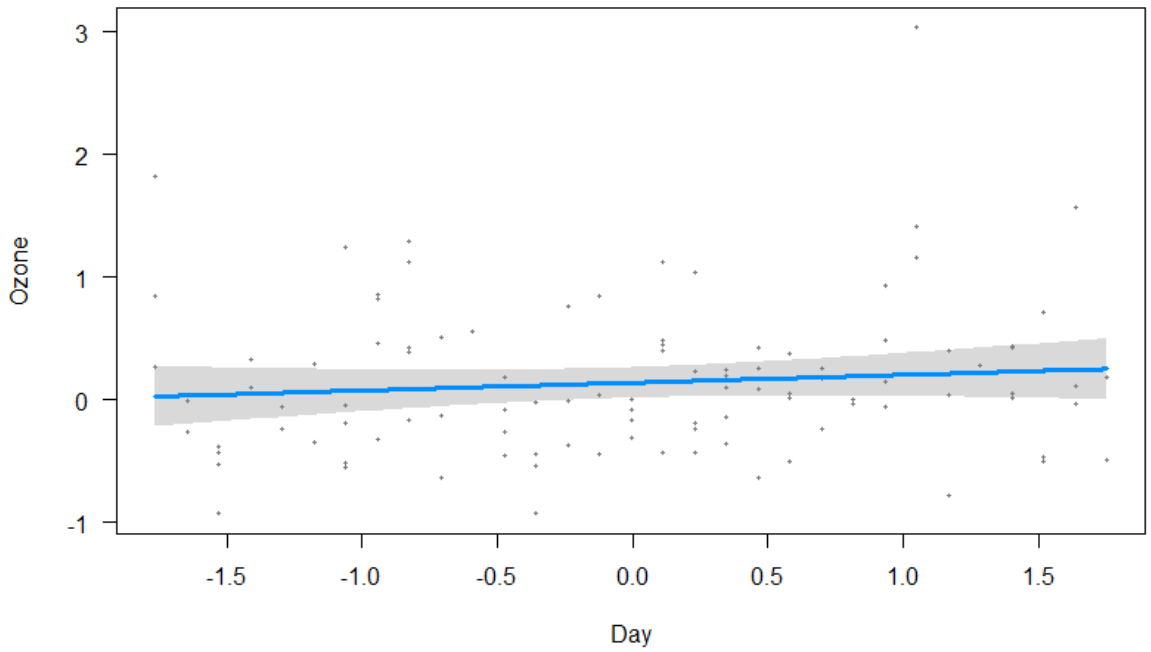
The Wind variable has negative correlation with ozone



The Temp variable has positive correlation with ozone



The Month variable has almost zero correlation with ozone



The Day variable has almost zero correlation with ozone

j. Calculate the correlation between any of the variables and the outcome variable.

```
# Correlation:
cor(df1)
```

	Ozone	Solar.R	Wind	Temp	Month	Day
Ozone	1	0.345336	-0.61728	0.710434	0.169246	-0.00875
Solar.R	0.345336	1	-0.12315	0.314873	-0.06859	-0.11479
Wind	-0.61728	-0.12315	1	-0.50043	-0.19994	0.05072
Temp	0.710434	0.314873	-0.50043	1	0.416801	-0.07664
Month	0.169246	-0.06859	-0.19994	0.416801	1	-0.01637
Day	-0.00875	-0.11479	0.05072	-0.07664	-0.01637	1

k. Build the linear regression model.

```
# Linear Regression Model with correlated variables
model_new = lm(Ozone~Wind+Temp, data = df1)
summary(model_new)
```

Call:

lm(formula = Ozone ~ Wind + Temp, data = df1)

Residuals:

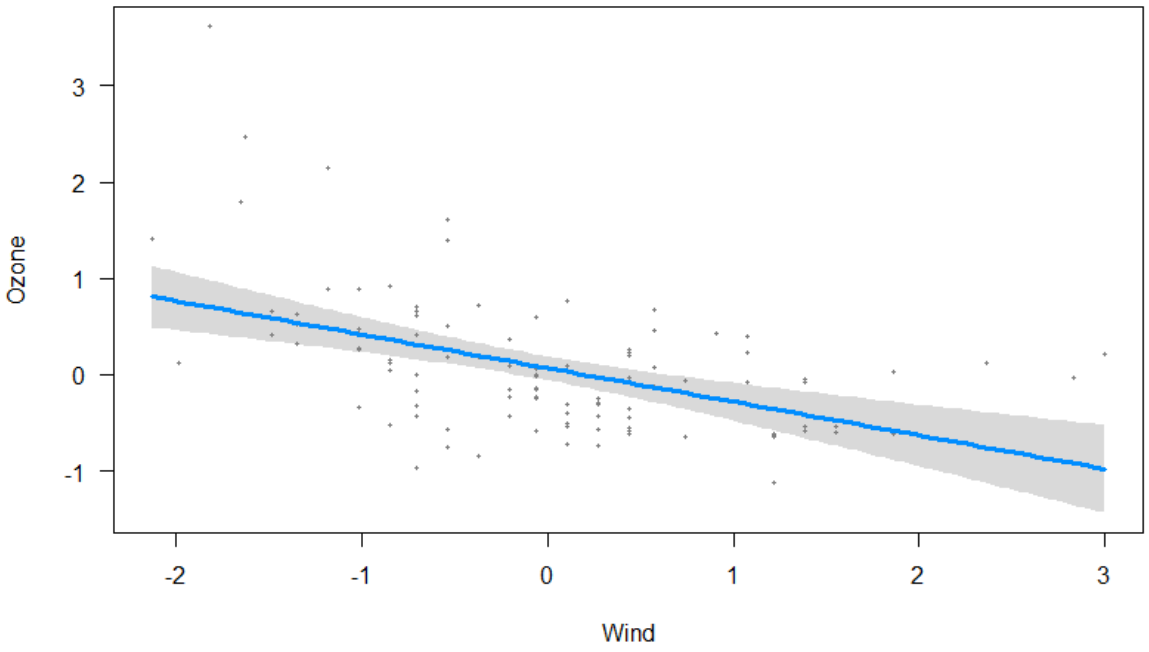
Min	1Q	Median	3Q	Max
-1.2821	-0.3639	-0.1041	0.3251	2.9115

Coefficients:

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.022e-16  6.203e-02   0.000        1
Wind        -3.492e-01  7.198e-02  -4.852 4.32e-06 ***
Temp         5.357e-01  7.198e-02   7.442 2.97e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6416 on 104 degrees of freedom
Multiple R-squared:  0.5961,      Adjusted R-squared:  0.5884
F-statistic: 76.75 on 2 and 104 DF,  p-value: < 2.2e-16
```

l. Show your model (visualization) and explain it.

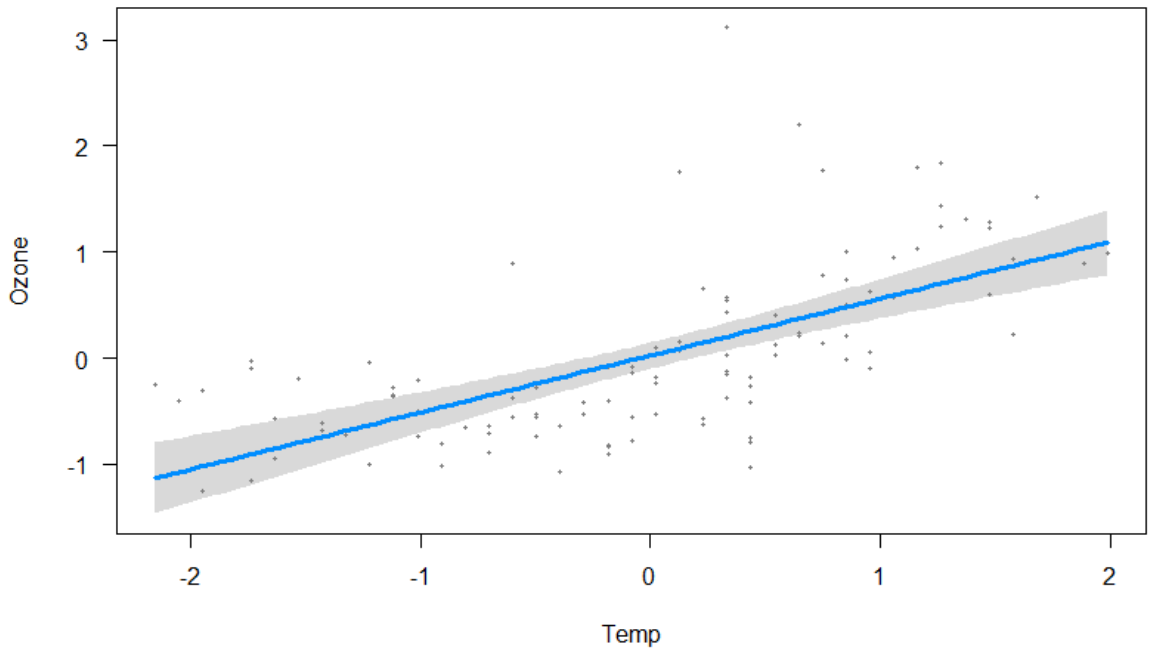
```
# Model Visualization
visreg(model_new)
```



The Wind variable has negative correlation with ozone

The Month variable has almost zero correlation with ozone

The Month variable has almost zero correlation with ozone



The Temp variable has positive correlation with ozone

2. Apply "Association Rule" Analysis on the "<u>Happiness dataset</u>".

a. Download the "Happiness" dataset from BB.

b. Set the Working Directory and install the "arules" and "arulesViz" package.

```
#Importing the required libraries
library(readr)
library(arules)
library(arulesViz)
```

c. Read in the Data for Modeling

```
# Importing the dataset
df2 = read_csv("happiness.csv")
```

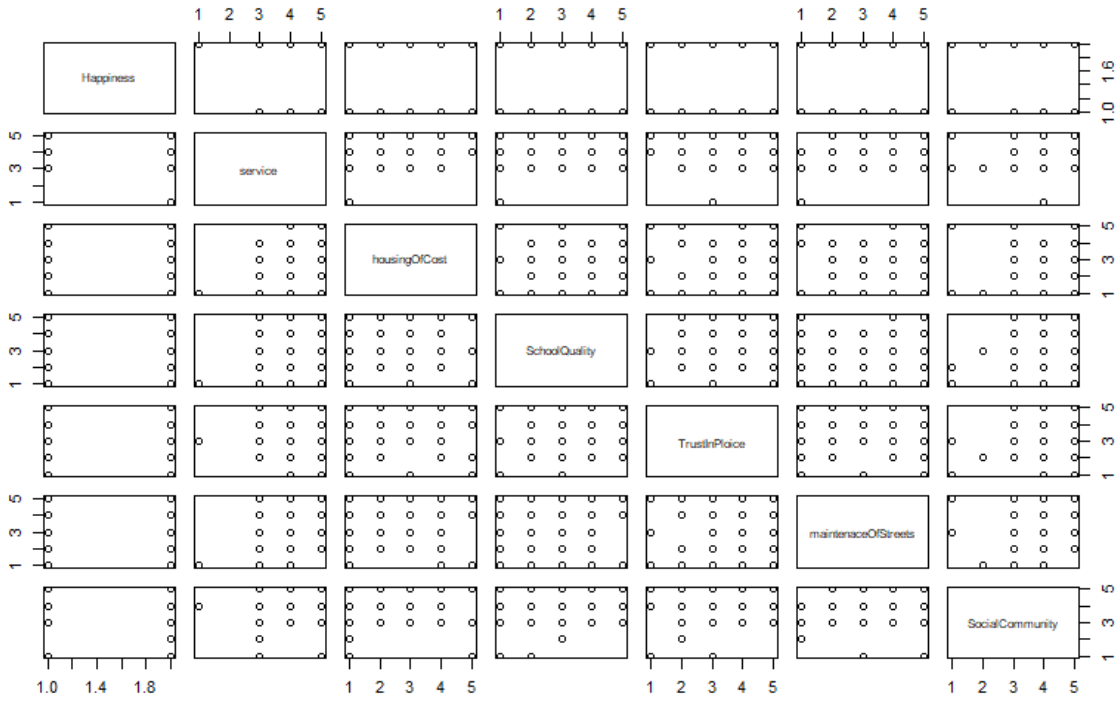
d. Review Transaction data

```
# show first 5 transactions
head(df2, 5)
```

Happiness	service	housingOfCost	SchoolQuality	TrustInPloice	maintenaceOfStreets	SocialCommunity
unhappy	3	3	3	4	2	4
unhappy	3	2	3	5	4	3
happy	5	3	3	3	3	5
unhappy	5	4	3	3	3	5
unhappy	5	4	3	3	3	5

e. Plot Transactions

```
# Plot transactions
Plot(df2)
```



f. Mine the Association Rules

```
# find association rules with default settings
rules.all = apriori(df2)
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
target  ext
       0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 14

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[18 item(s), 143 transaction(s)] done [0.00s].
sorting and recoding items ... [17 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [230 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
inspect(rules.all)

# rules with rhs containing "happy or unhappy" only
rulesHappy = apriori(df2,  parameter = list(minlen=2, supp=0.005, conf=0.8),
appearance = list(rhs=c("Happiness=unhappy", "Happiness=happy"), default="lhs"))
inspect(rulesHappy)
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
target  ext
       0.8    0.1    1 none FALSE          TRUE       5   0.005      2     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
   0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[2 item(s)] done [0.00s].
set transactions ...[18 item(s), 143 transaction(s)] done [0.00s].
sorting and recoding items ... [18 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 done [0.00s].
writing ... [687 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
quality(rulesHappy) <- round(quality(rulesHappy), digits=3)
rulesHappy.sorted <- sort(rulesHappy, by="lift")
inspect(rulesHappy.sorted)
```

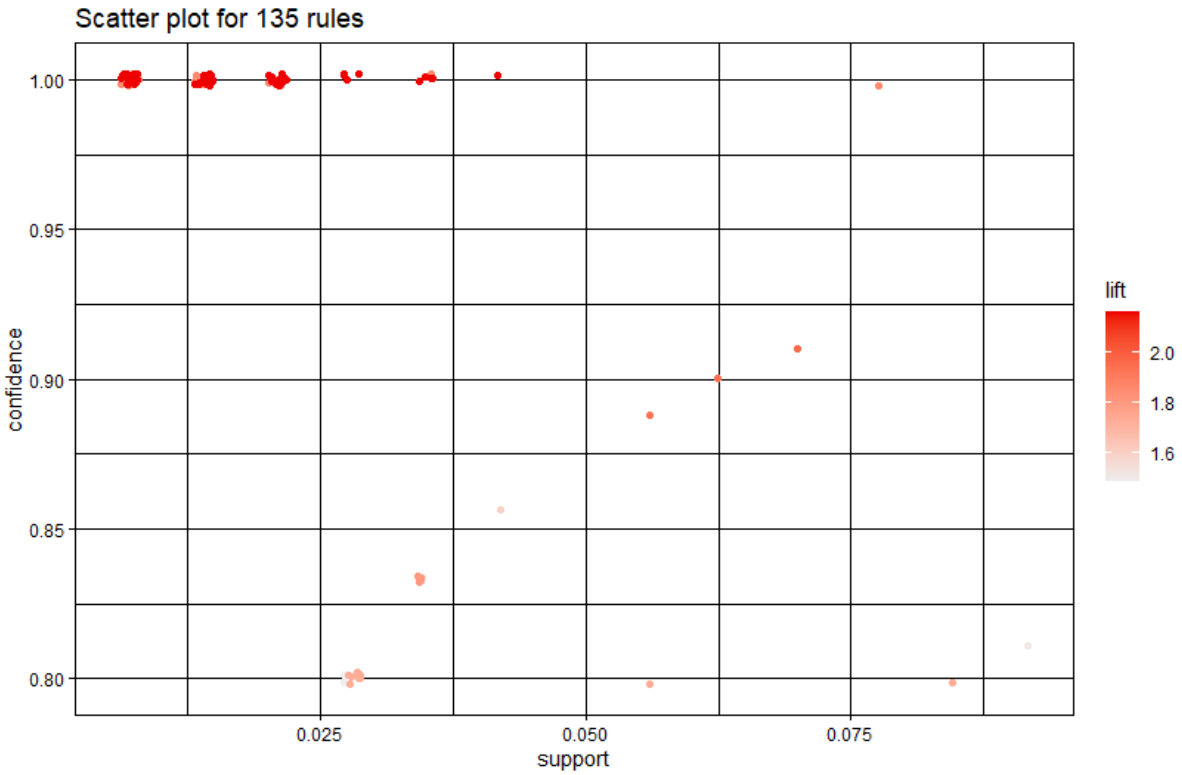
g. Remove redundancy from data

```
# Remove redundant rules
subset.matrix = is.subset(rulesHappy.sorted, rulesHappy.sorted, sparse = FALSE)
subset.matrix[lower.tri(subset.matrix, diag=T)] = NA
redundant = colSums(subset.matrix, na.rm=T) >= 1

rulesHappy.pruned = rulesHappy.sorted[!redundant]
inspect(rulesHappy.pruned)
```

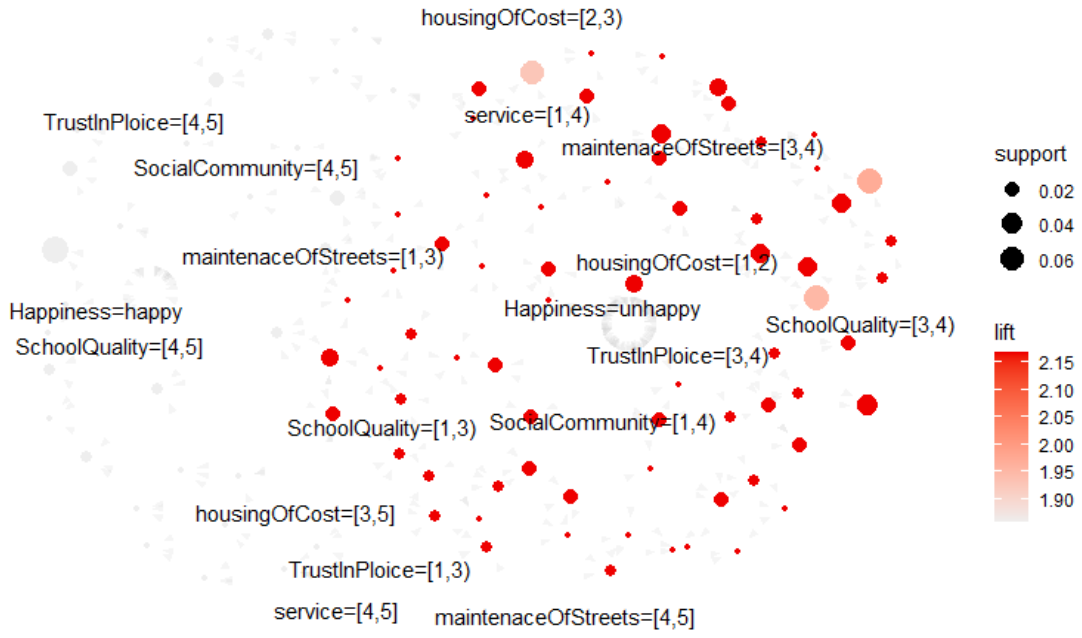
h. Visualize results

```
#  Visualizing Association Rules
```

```
plot(rulesHappy.pruned )
```



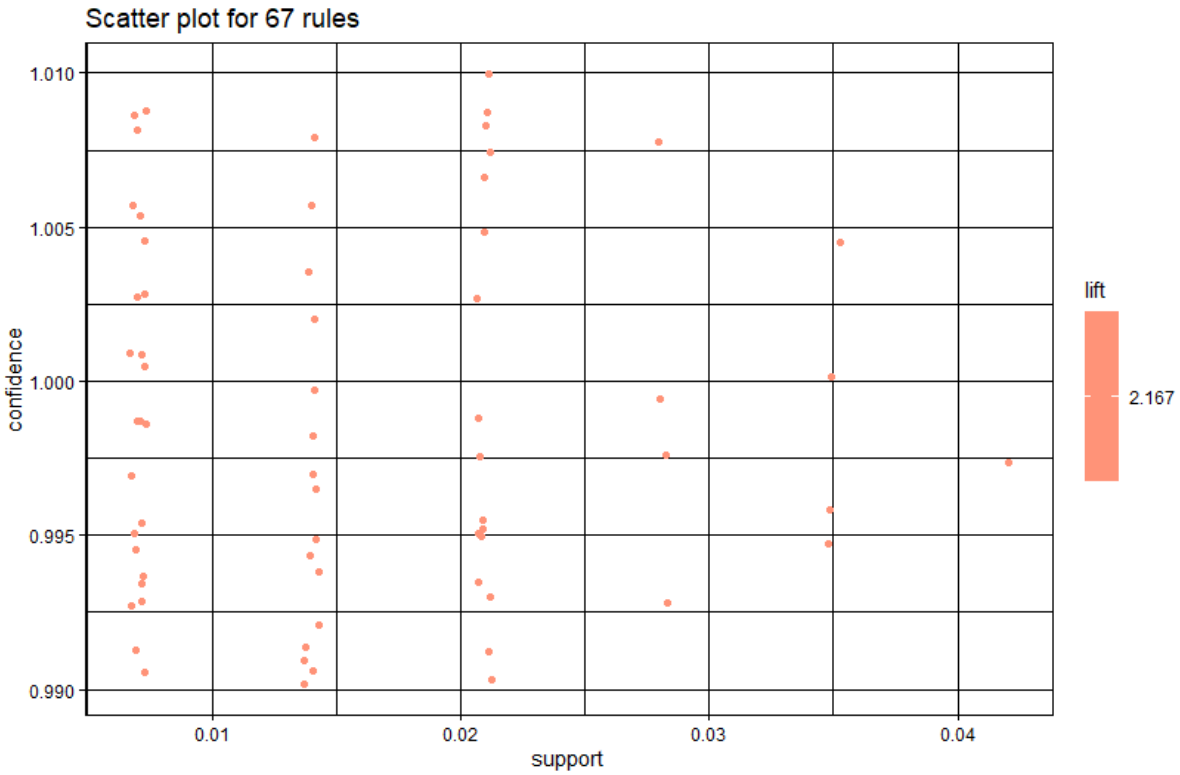

```
plot(rulesHappy.pruned , method="graph")
```



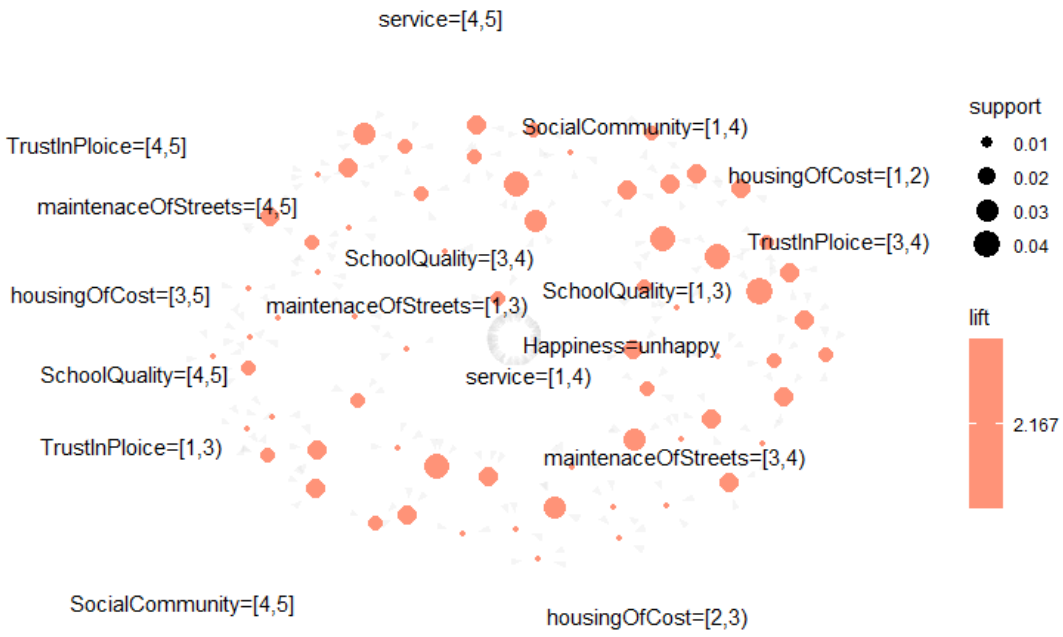
i. Extract the Rules in which the Confidence Value is >0.8 and high lift and visualize results and present your insight.

```
#Rules with Confidence Value >0.8 and high lift
rulesLift <- sort(subset(rulesHappy.pruned, subset = lift > 2), by="lift")
```

```
inspect(rulesLift)

# Visualizing Association Rules
plot(rulesLift)
```




```
plot(rulesLift , method="graph")
```

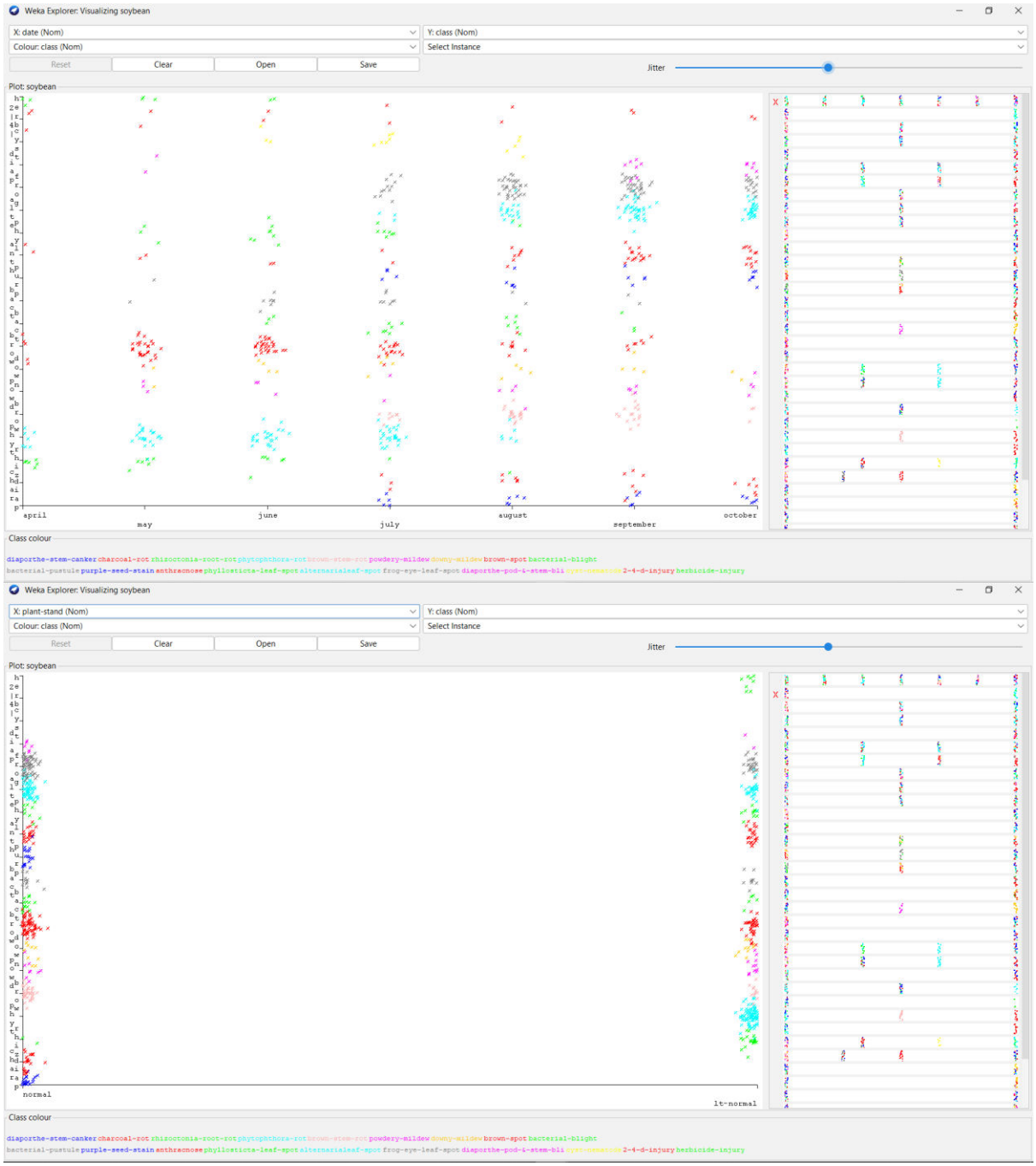


Here, we are only considering the rules with confidence > 0.8 and high lift (>2). We can see an improvement in the results.

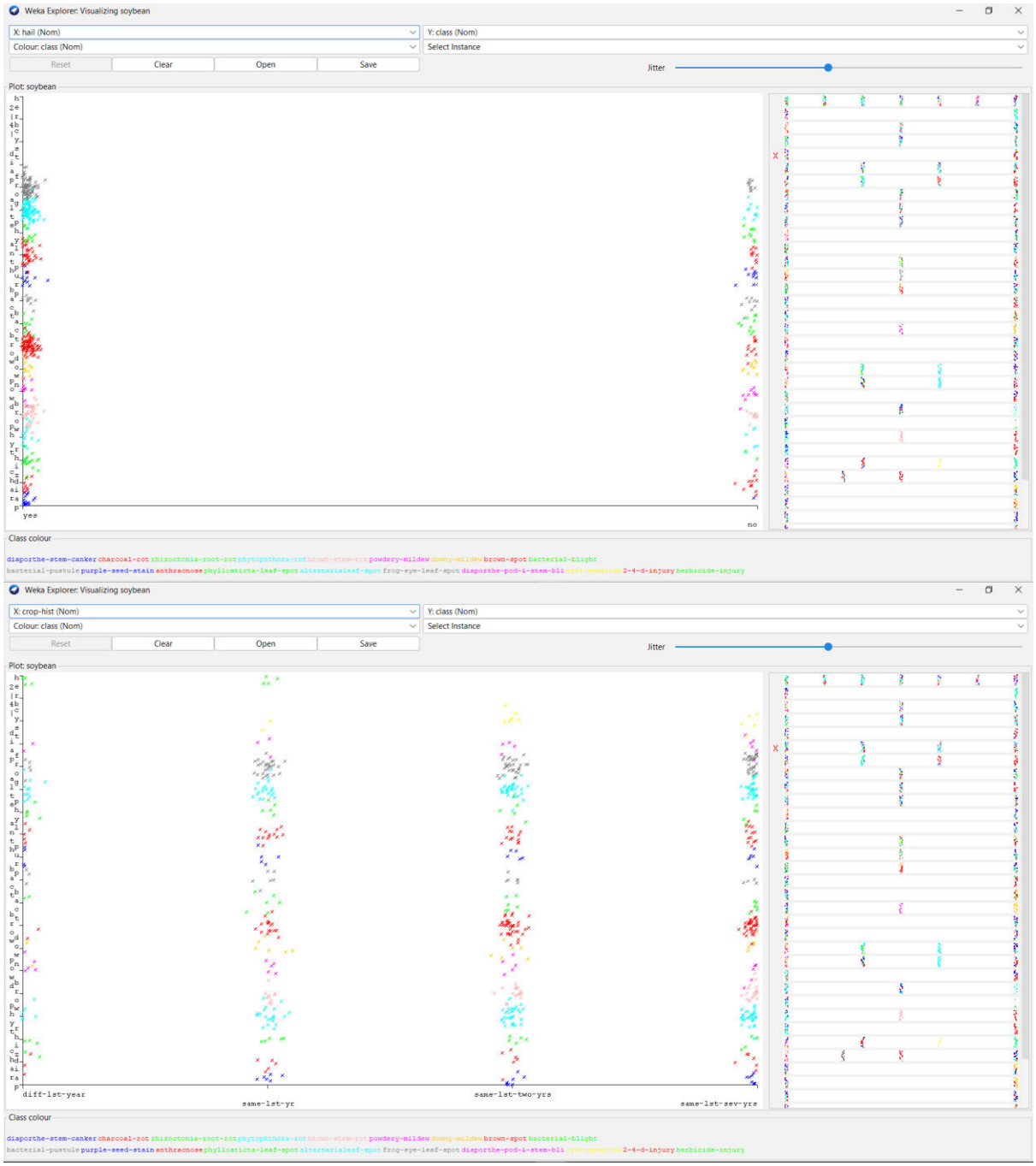
Weka

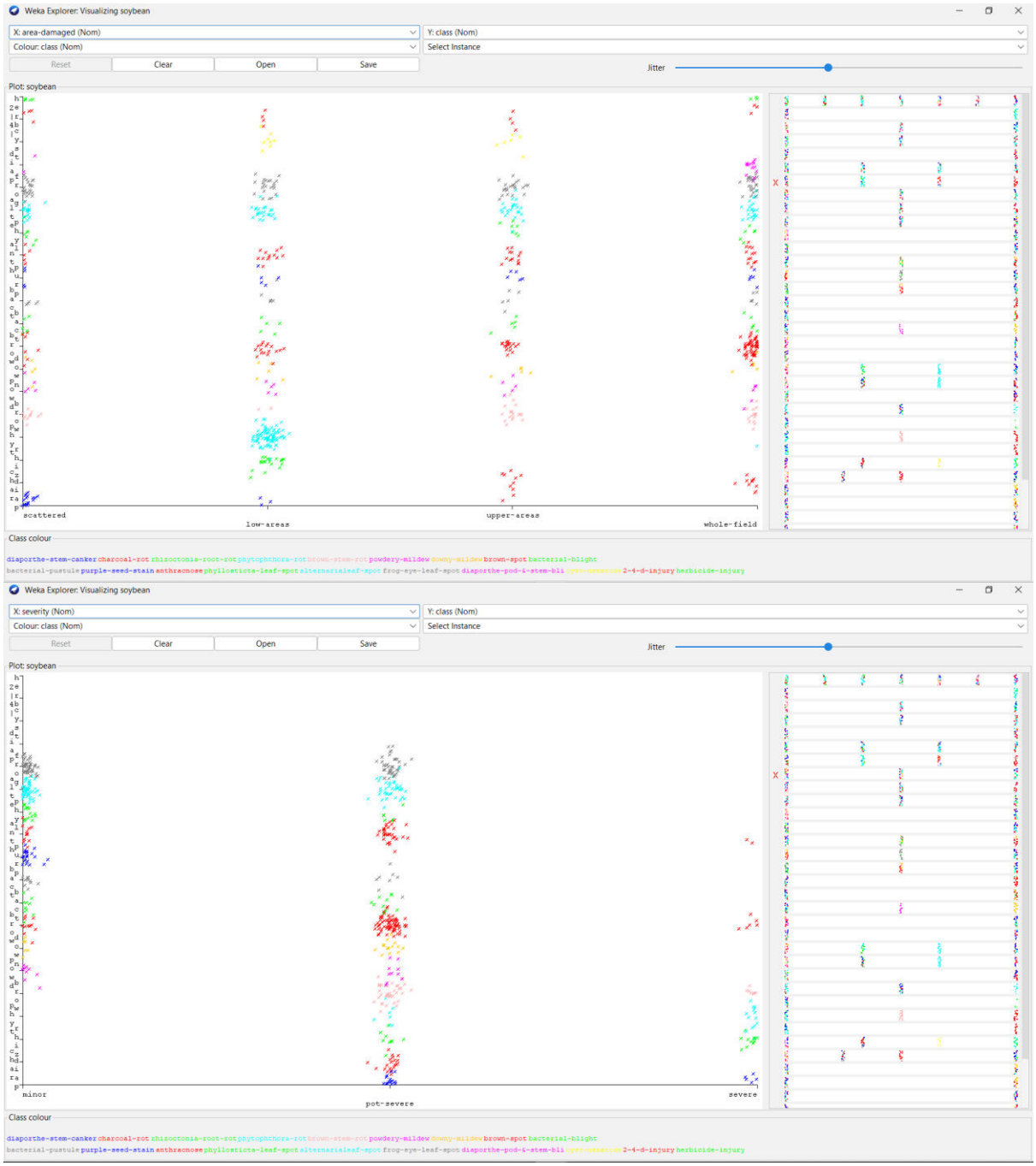
Q2. Analysis on "Soybean Dataset".

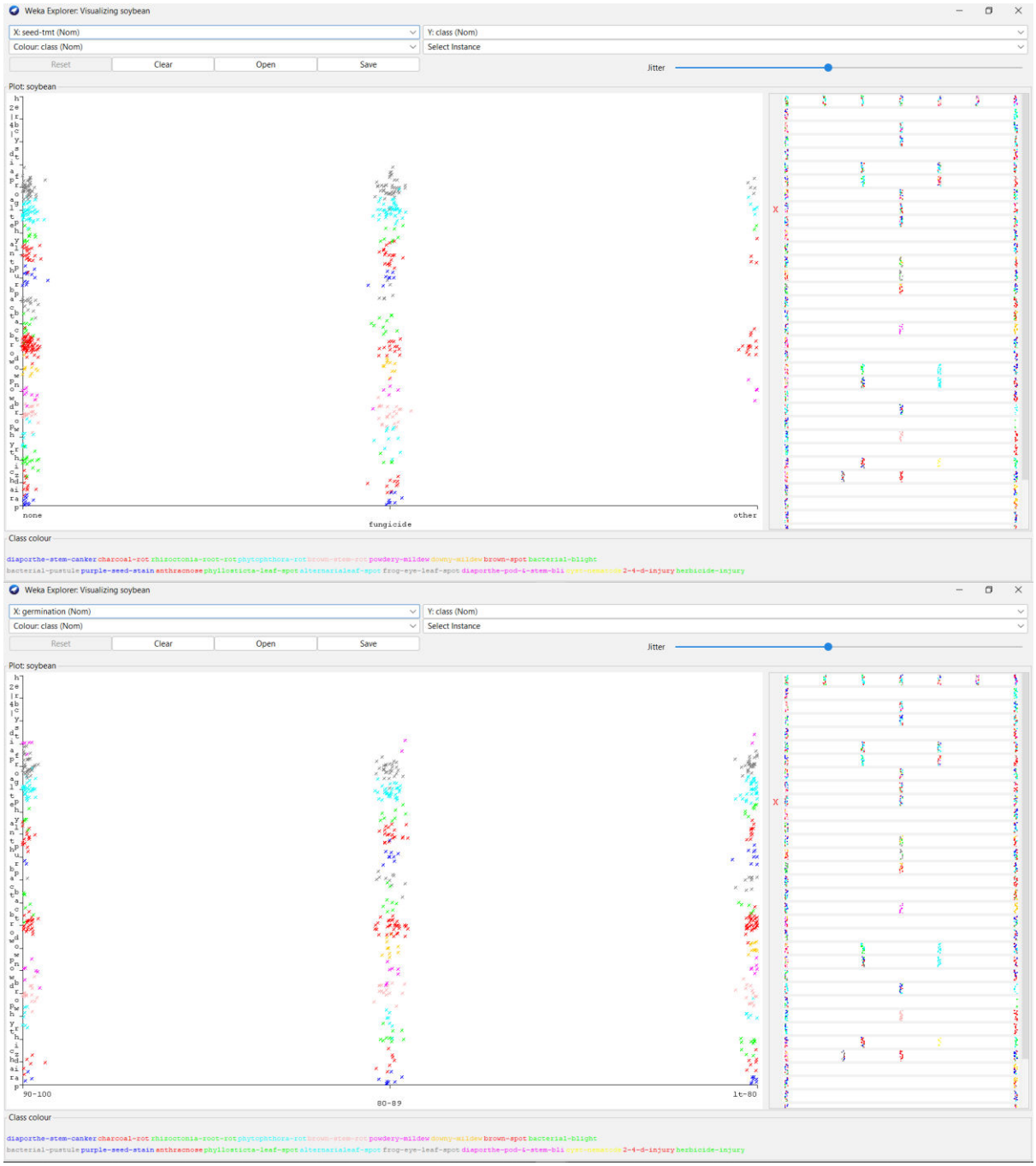
a. Downloaded the "soybean.arff" dataset.
b. Uploaded the dataset to WEKA explorer.
c. There is a total of 35 categorical attributes, some nominal and some ordered. Apart from that, the class attribute has a total of 19 distinct categories. The class attribute is stating different kinds of Soybean Diseases.
d. Distributions of data over classes:

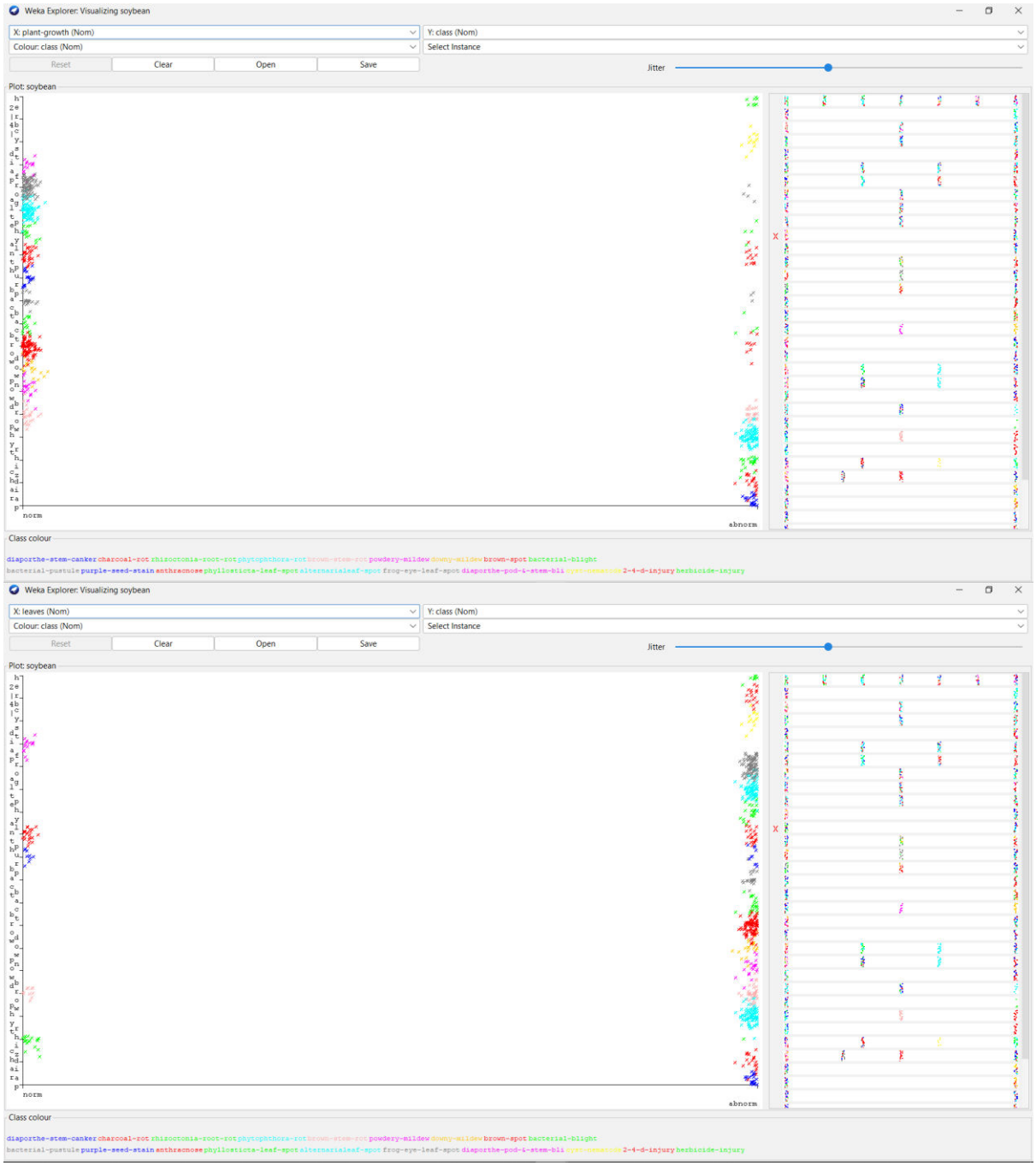

Weka Explorer: Visualizing soybean
X: date (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
april
may
june
july
august
september
october
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: plant-stand (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
normal
lt-normal
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

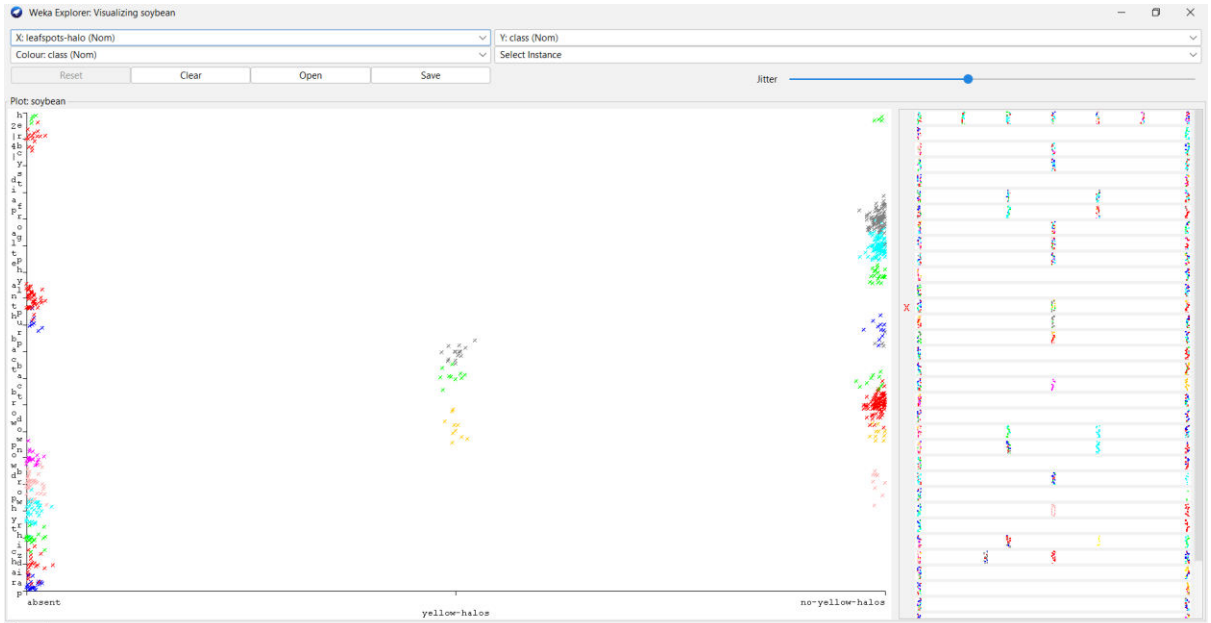

Weka Explorer: Visualizing soybean
X: precip (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
lt-norm
norm
gt-norm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: temp (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
lt-norm
norm
gt-norm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

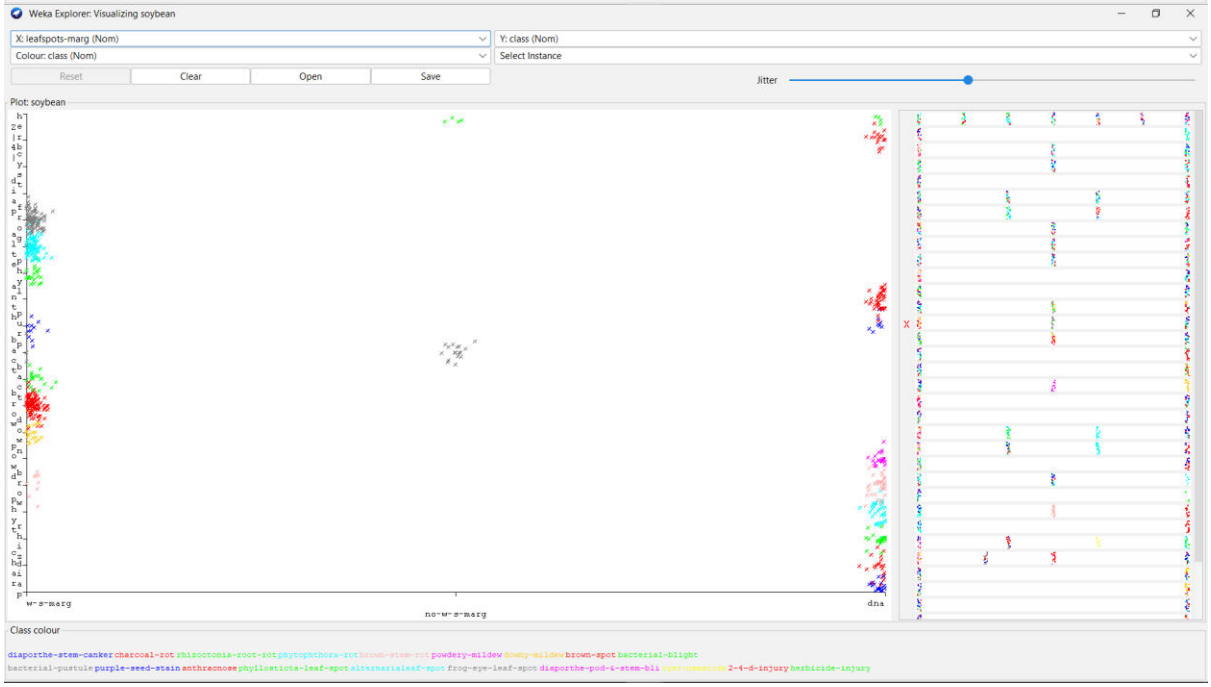

Weka Explorer: Visualizing soybean
X: hail (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
yes
no
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: crop-hist (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
diff-lst-year
same-lst-yr
same-lst-two-yrs
same-lst-sev-yrs
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

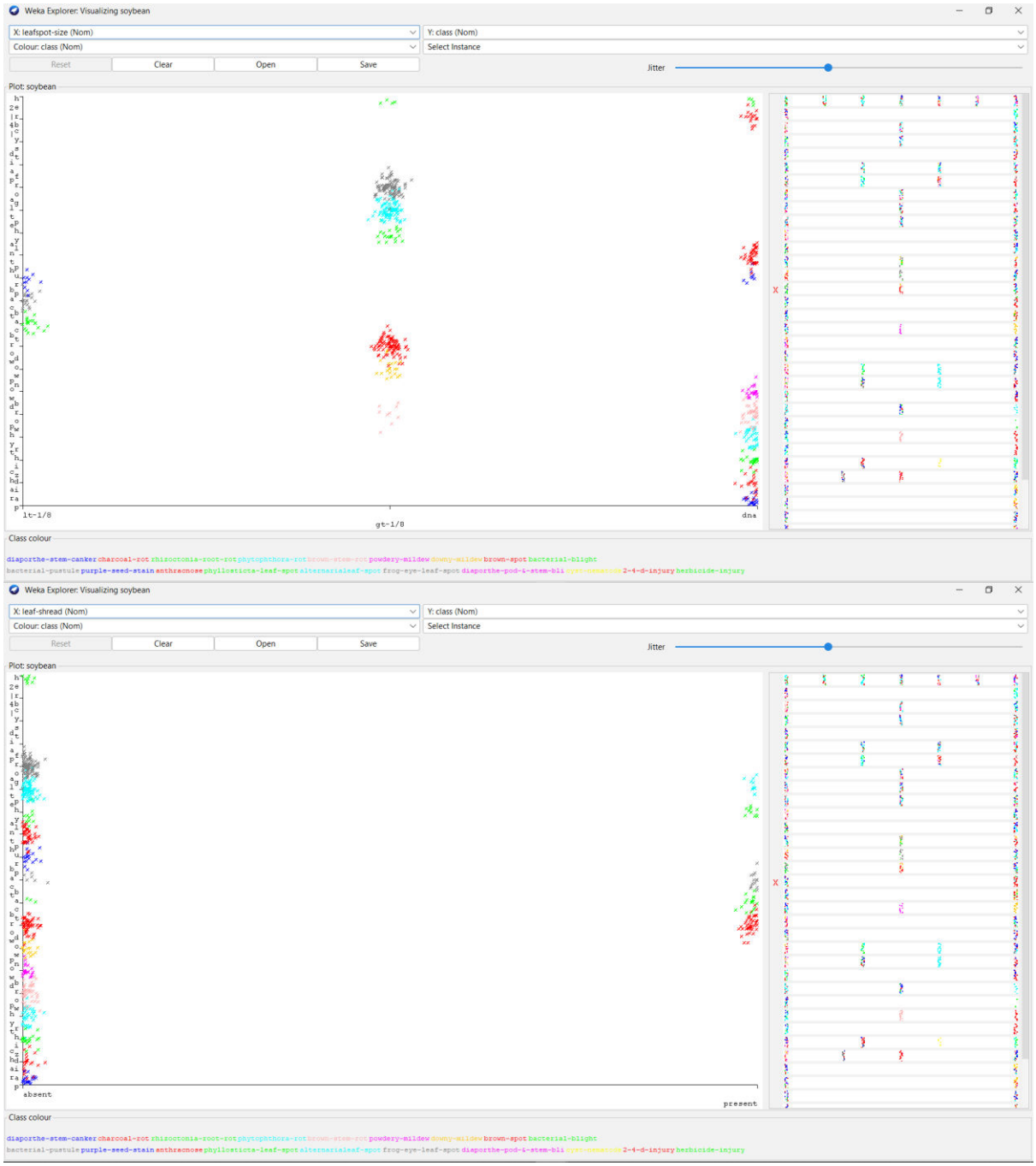

Weka Explorer: Visualizing soybean
X: area-damaged (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
scattered
low-areas
upper-areas
whole-field
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: severity (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
minor
pot-severe
severe
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

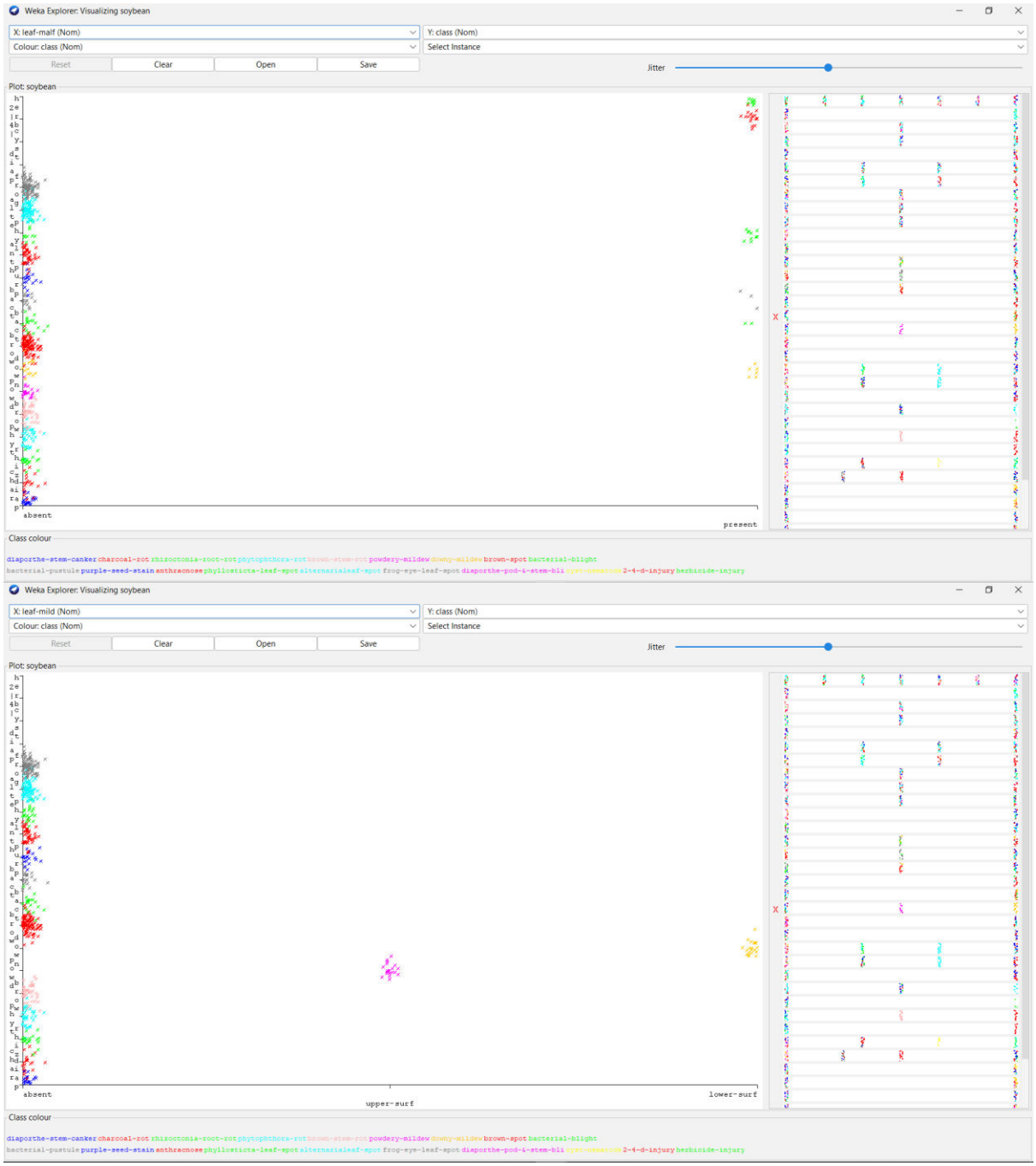

Weka Explorer: Visualizing soybean
X: seed-tmt (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
none
fungicide
other
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: germination (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
90-100
80-89
lt-80
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

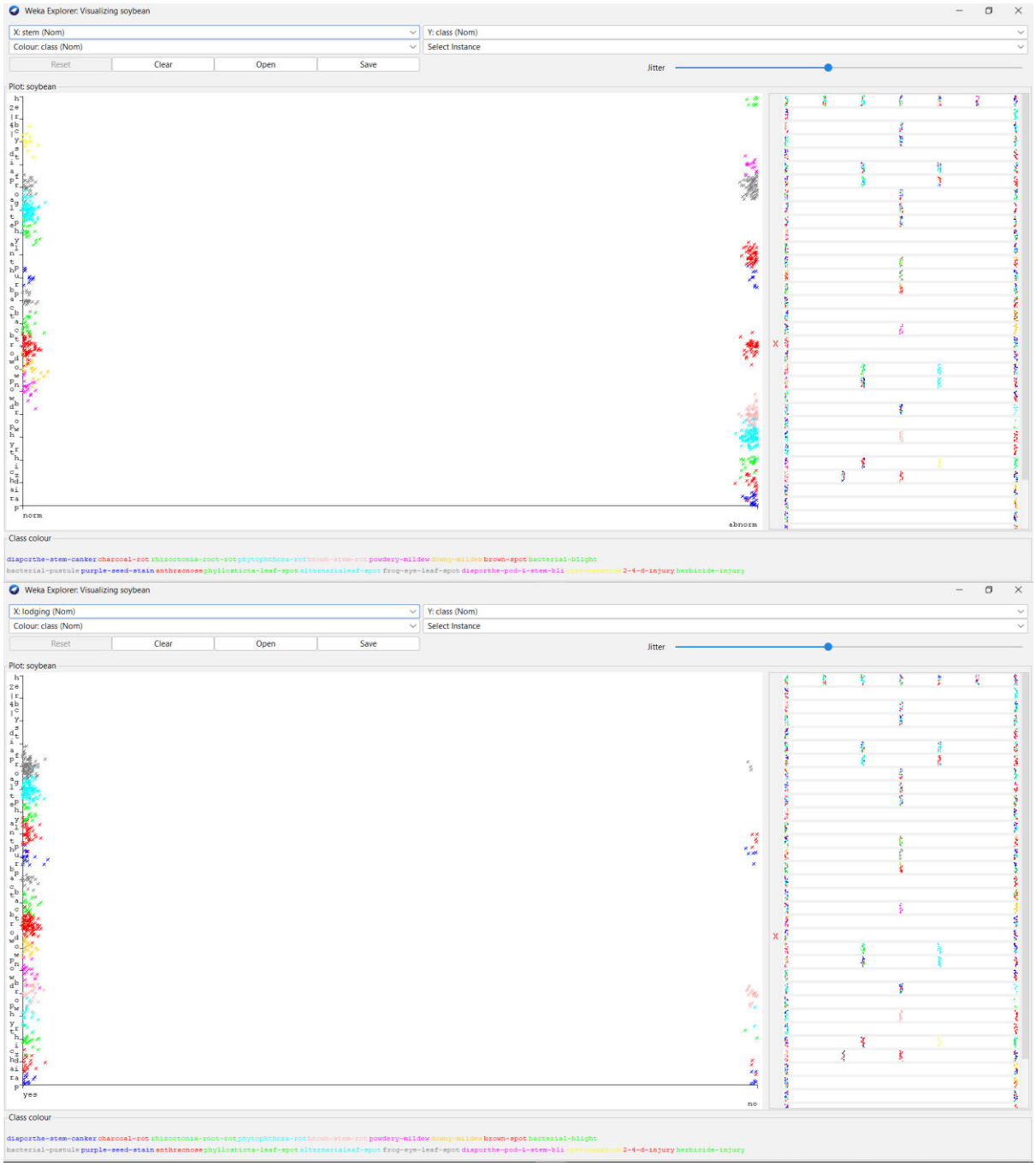

Weka Explorer: Visualizing soybean
X: plant-growth (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
abnorm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: leaves (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
abnorm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury


Weka Explorer: Visualizing soybean
X: leafspots-halo (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
yellow-halos
no-yellow-halos
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

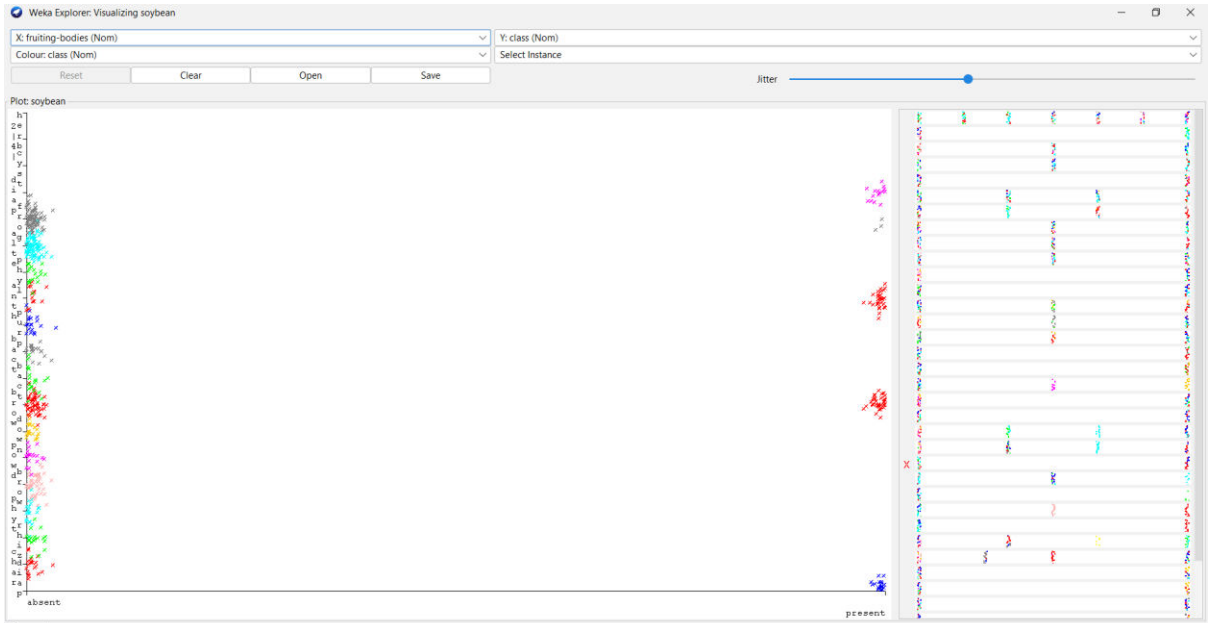

Weka Explorer: Visualizing soybean
X: leafspots-marg (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
w-s-marg
no-w-s-marg
dna
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

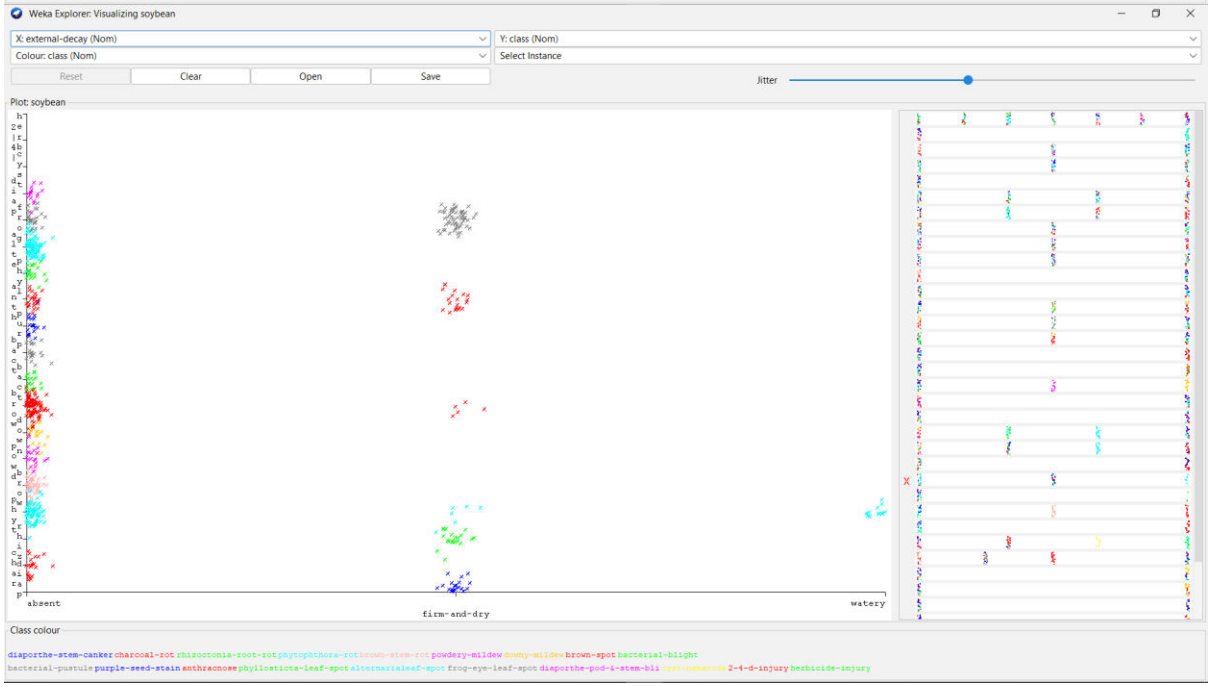

Weka Explorer: Visualizing soybean
X: leafspot-size (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
lt-1/8
gt-1/8
dna
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternaria-leaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: leaf-shread (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternaria-leaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

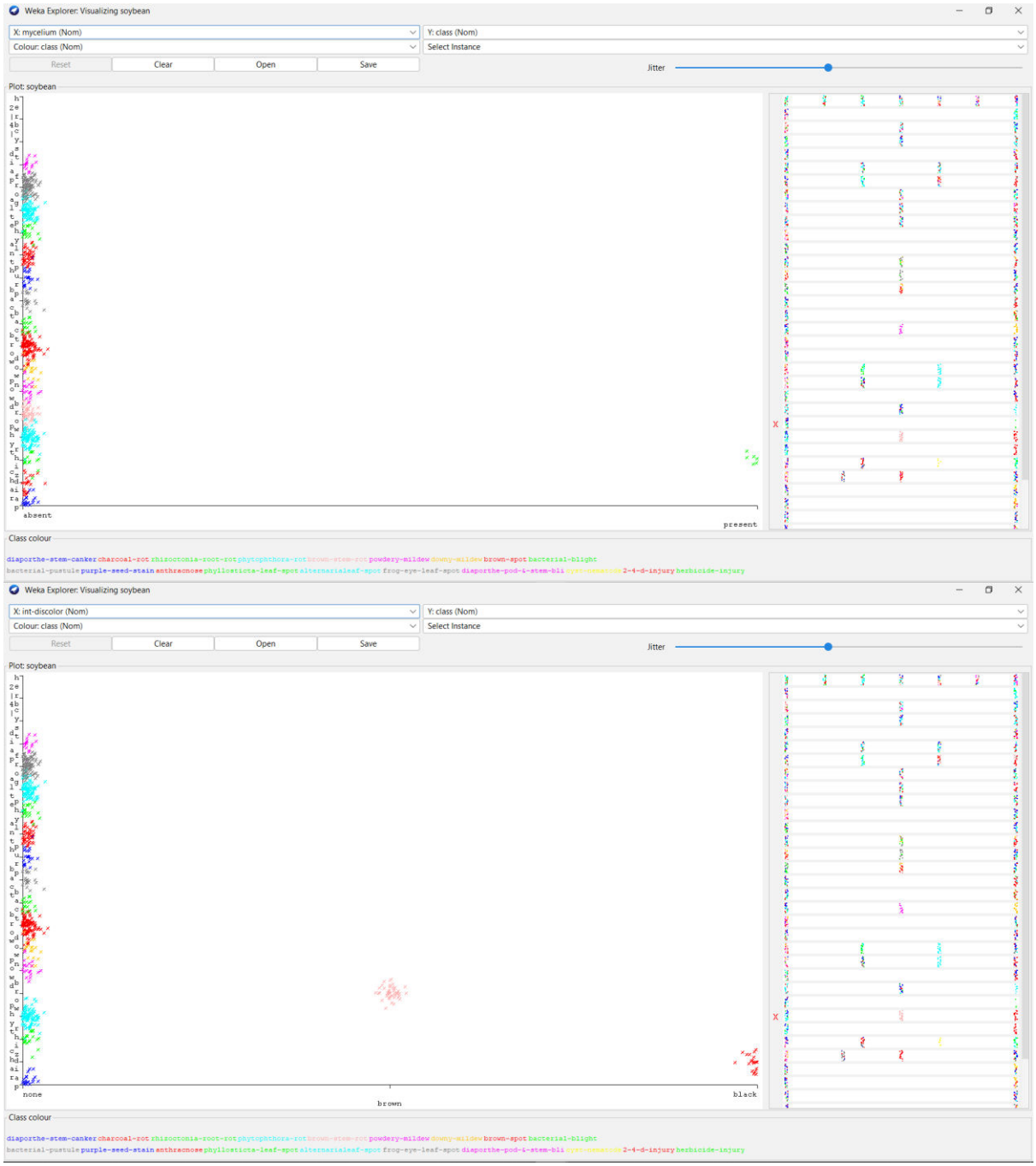

Weka Explorer: Visualizing soybean
X: leaf-malf (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: leaf-mild (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
upper-surf
lower-surf
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

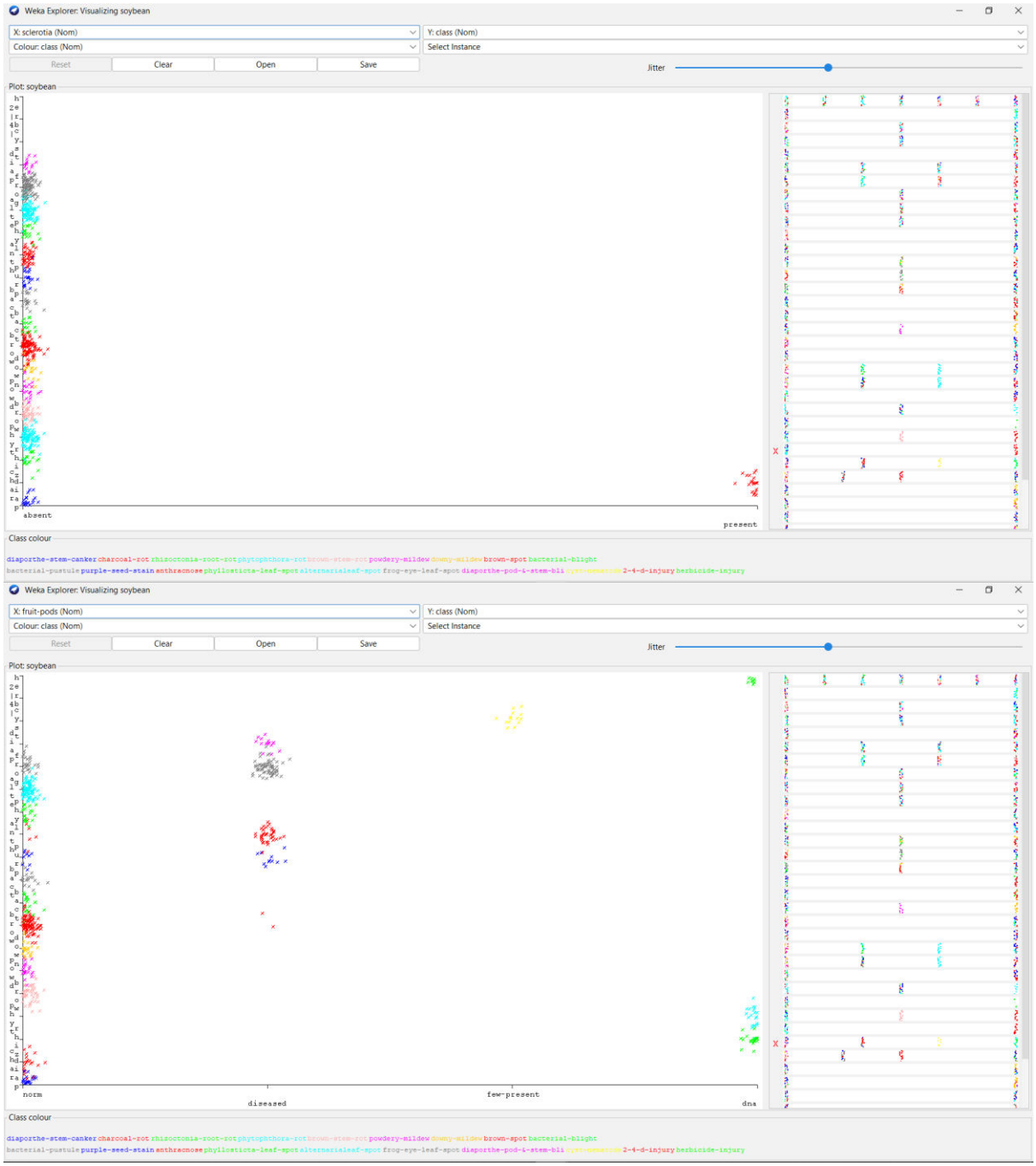

Weka Explorer: Visualizing soybean
X: stem (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
abnorm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: lodging (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
yes
no
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

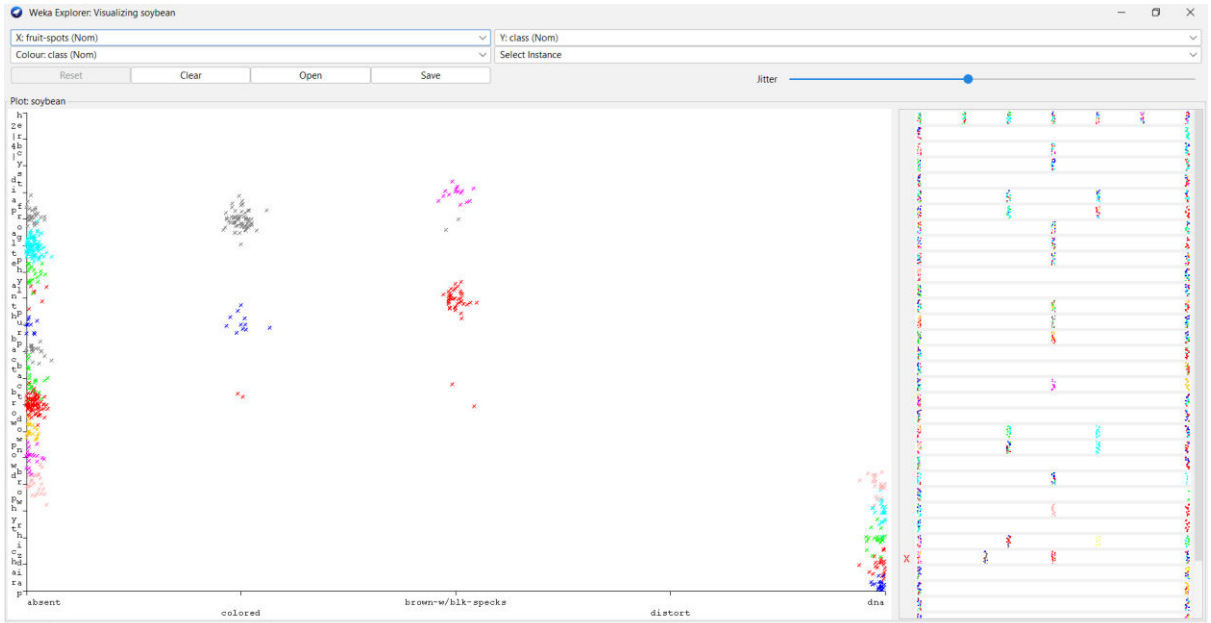

Weka Explorer: Visualizing soybean
X: stem-cankers (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
below-soil
above-soil
above-sec-nde
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: canker-lesion (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
dna
brown
dk-brown-blk
tan
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

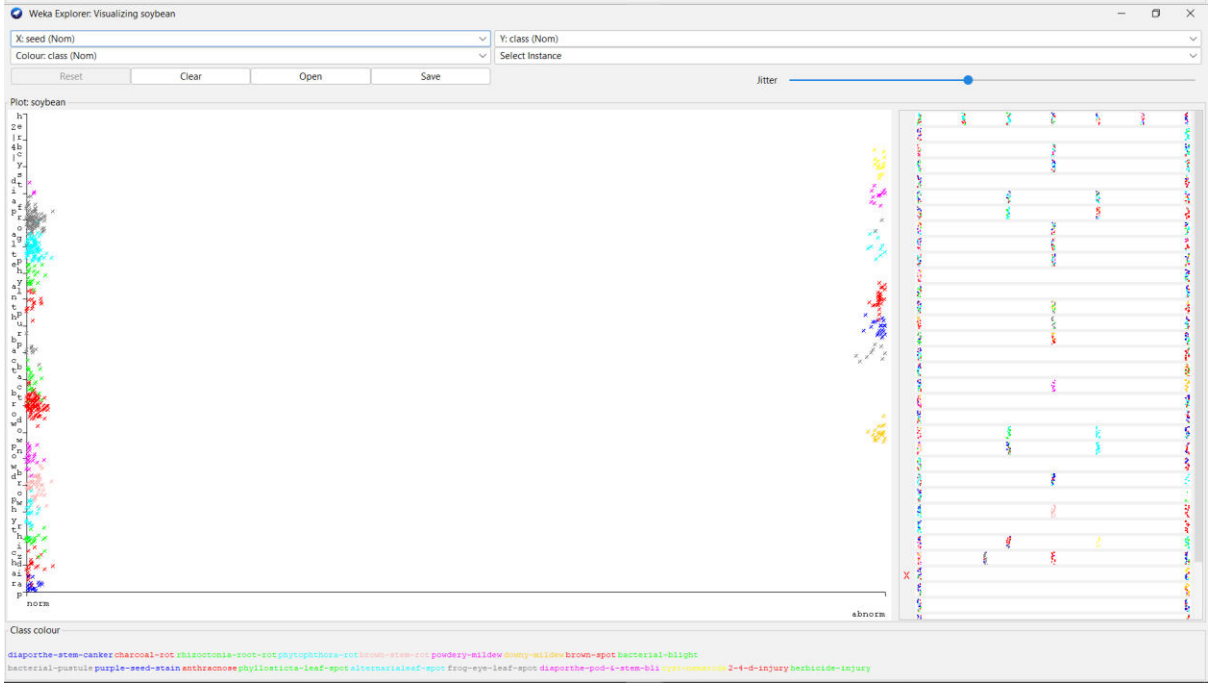

Weka Explorer: Visualizing soybean
X: fruiting-bodies (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

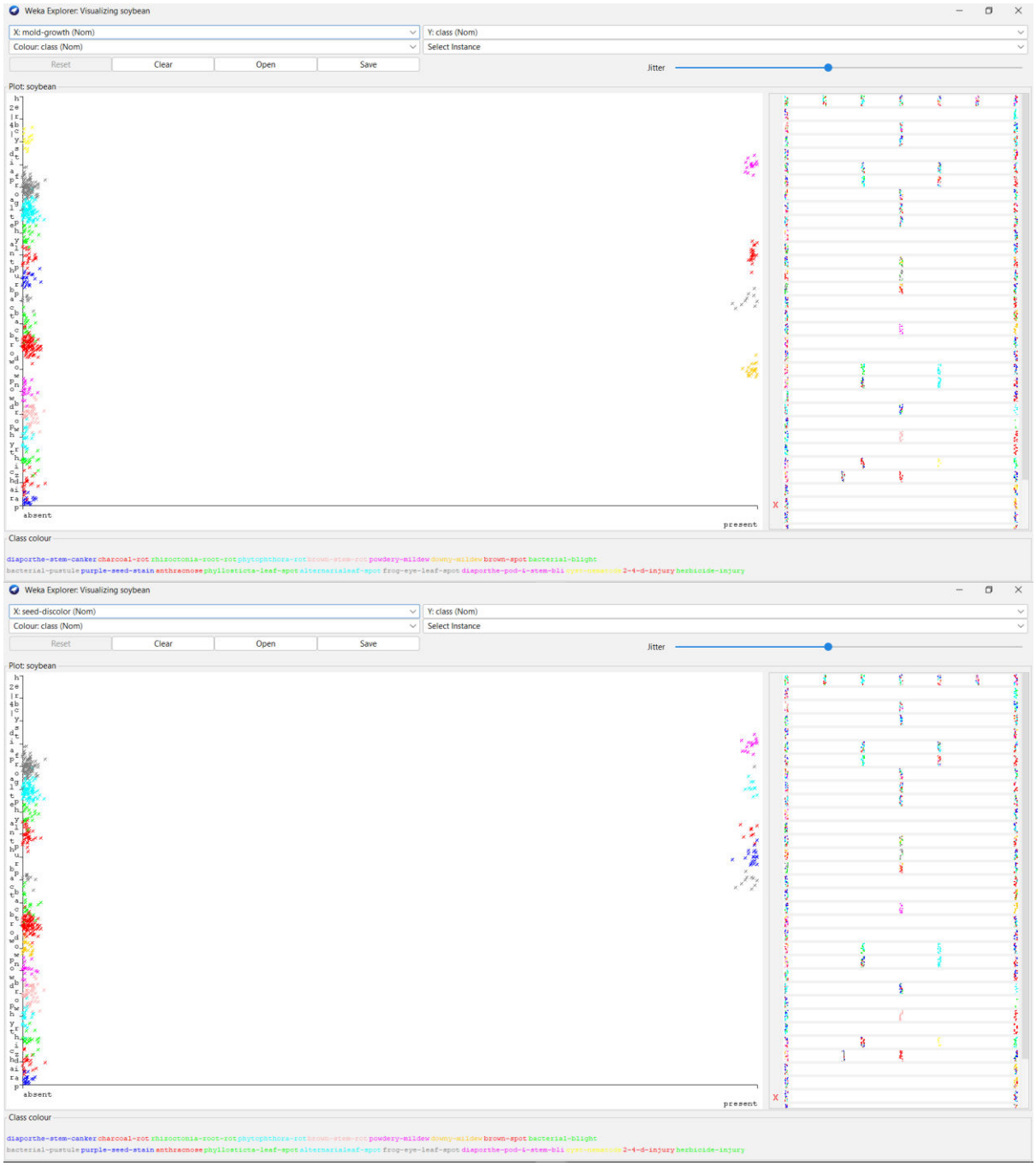

Weka Explorer: Visualizing soybean
X: external-decay (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
firm-and-dry
watery
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

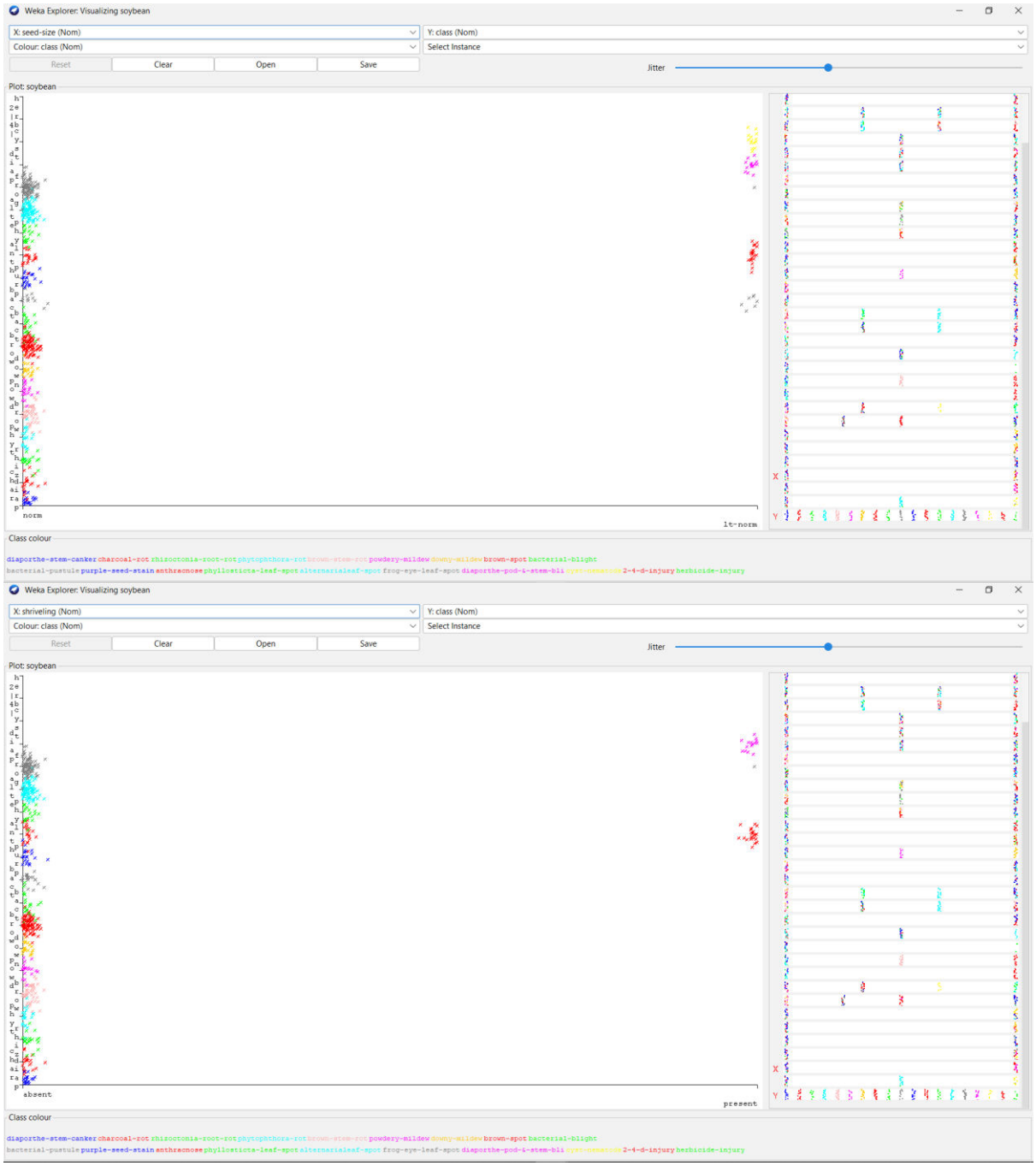

Weka Explorer: Visualizing soybean
X: mycelium (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
X
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: int-discolor (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
none
brown
black
X
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

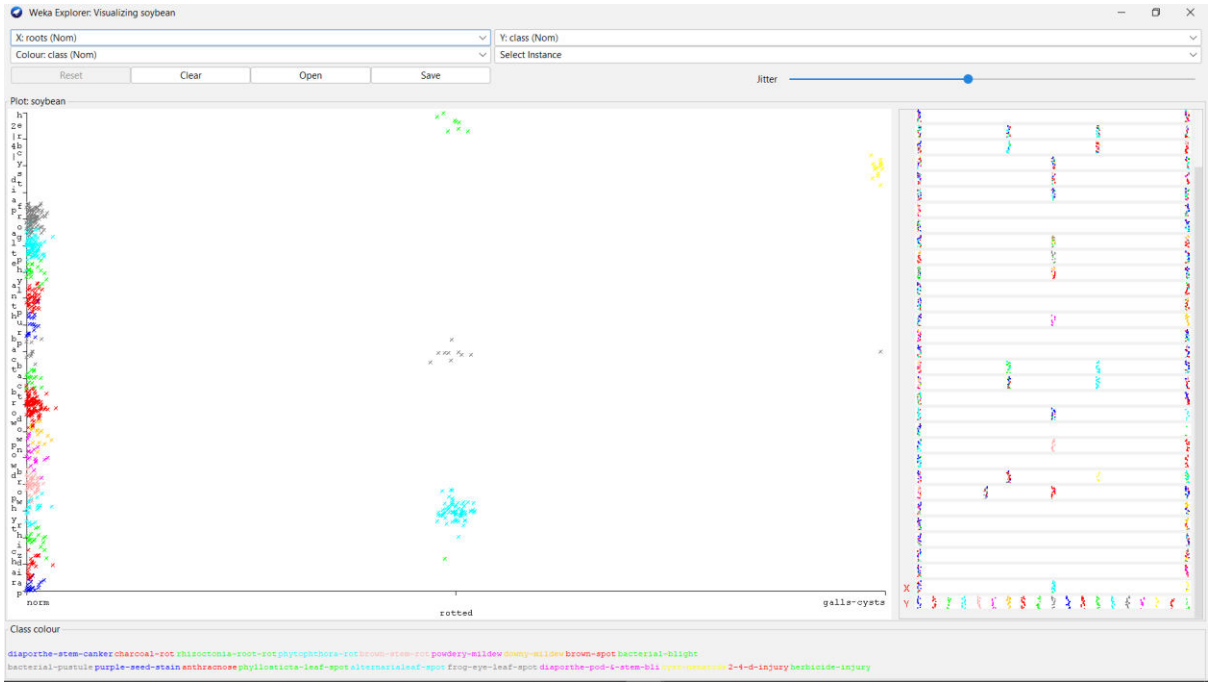

Weka Explorer: Visualizing soybean
X: sclerotia (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: fruit-pods (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
diseased
few-present
dna
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury


Weka Explorer: Visualizing soybean
X: fruit-spots (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
colored
brown-w/blk-specks
distort
dna
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury


Weka Explorer: Visualizing soybean
X: seed (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
abnorm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury


Weka Explorer: Visualizing soybean
X: mold-growth (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: seed-discolor (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury

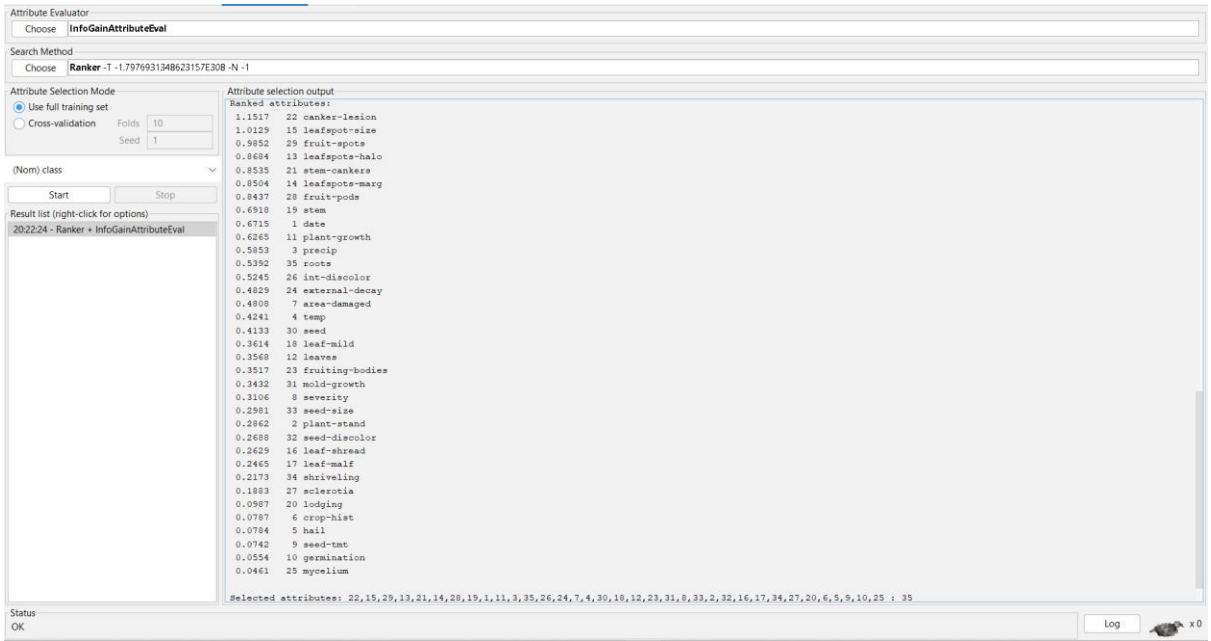

Weka Explorer: Visualizing soybean
X: seed-size (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
norm
lt-norm
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury
Weka Explorer: Visualizing soybean
X: shriveling (Nom)
Y: class (Nom)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: soybean
absent
present
Class colour
diaporthe-stem-canker charcoal-rot rhizoctonia-root-rot phytophthora-rot brown-stem-rot powdery-mildew downy-mildew brown-spot bacterial-blight
bacterial-pustule purple-seed-stain anthracnose phyllosticta-leaf-spot alternarialeaf-spot frog-eye-leaf-spot diaporthe-pod-&-stem-bli cyst-nematode 2-4-d-injury herbicide-injury



From the above plots, we can conclude:

1. In the month of April, there is a very less chance of occurring some of the diseases like – bacterial-purstule, frog-eye-leaf-spot, powdery-mildew, etc.
2. If the precip is lt-norm, there is a very less chance of occurring herbicide-injury, diaporthe-stem-canker, etc.
3. Most of the diseases occur at norm and gt-norm temperature.
4. There are very high chances of occurring different diseases when root is norm.
5. There are very high chances of occurring different diseases when int-discolor is none.
6. Etc.

e. Applied attribute selection classifier evaluator (InfoGainAttributeEval) to the dataset. The screenshot of the outcome is given below:



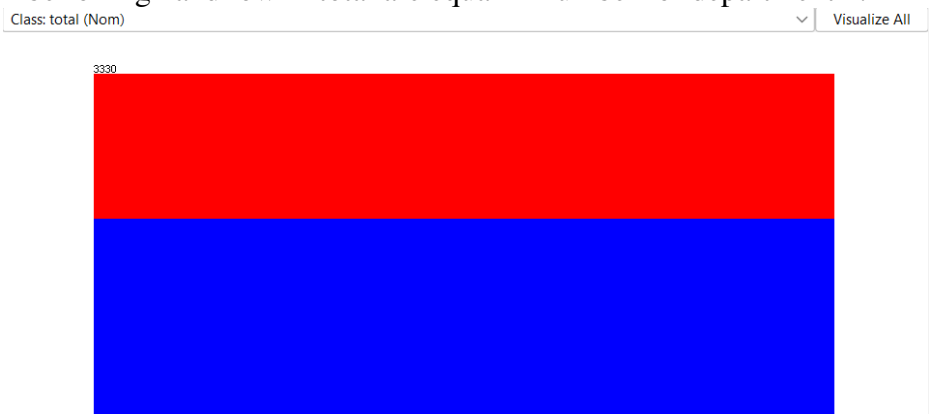
According to the above Information Gain values we can choose the following features to be important in building predictive models: canker-lesion, leafspot-size, fruit-spots, leafspots-halo, stem-cankers, leafspots-marg, fruit-pods, stem, date, plant-growth, precip, roots and int-discolor.

Q3. Analysis on "Supermarket Dataset".

a. Downloaded the "supermarket.arff" dataset.
b. Loaded the dataset into WEKA explorer.
c. Visualizing the dataset:

department 1

The number of high and low in total are equal in number for department 1.



bread and cake

The number of high and low in total are almost equal in number for bread and cake.

baking needs

The number of high and low in total are almost equal in number for baking needs.

d. Used Associate feature applying Apriori algorithm. metricType is set to 'Confidence' and numRules is set to 10. The screenshot of the output is given below:

Associator output

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

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

Best rules found:

 1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
 2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696    <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
 3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705    <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)
```

h. ii. From the above output we can predict that,

'When a person goes to the supermarket to buy biscuits, frozen foods and fruit, there is a high chance that the person will buy bread and cake.'

Part – 1

1. “Naïve Bayes” classifier using the “contact-lenses.arff” dataset:

I. Uploaded the dataset into the WEKA explorer.

II. Attributes in the dataset:

Name: age | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count	Weight
1	young	8	8
2	pre-presbyopic	8	8
3	presbyopic	8	8

Name: spectacle-prescrip | Type: Nominal
Missing: 0 (0%) | Distinct: 2 | Unique: 0 (0%)

No.	Label	Count	Weight
1	myope	12	12
2	hypermetrope	12	12

Name: astigmatism | Type: Nominal
Missing: 0 (0%) | Distinct: 2 | Unique: 0 (0%)

No.	Label	Count	Weight
1	no	12	12
2	yes	12	12

Name: tear-prod-rate | Type: Nominal
Missing: 0 (0%) | Distinct: 2 | Unique: 0 (0%)

No.	Label	Count	Weight
1	reduced	12	12
2	normal	12	12

Class attribute:

Name: contact-lenses | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count	Weight
1	soft	5	5
2	hard	4	4
3	none	15	15

Here class attribute is “contact-lenses”.
It represents if the required contact lenses are ‘soft’, ‘hard’ or ‘none’ type depending on the other attributes.

III. Applied Naïve Bayes Classifier with 80% train-test split.
Output predictions is set to “PlainText”.
Other settings are set to default.

IV. Information of the model:

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     contact-lenses
Instances:    24
Attributes:   5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses
Test mode:    split 80.0% train, remainder test

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

Naive Bayes Classifier

                     Class
Attribute              soft   hard   none
                     (0.22) (0.19) (0.59)
=========================================
age
  young                 3.0    3.0    5.0
  pre-presbyopic        3.0    2.0    6.0
  presbyopic            2.0    2.0    7.0
  [total]               8.0    7.0   18.0

spectacle-prescrip
  myope                 3.0    4.0    8.0
  hypermetrope          4.0    2.0    9.0
  [total]               7.0    6.0   17.0

astigmatism
  no                    6.0    1.0    8.0
  yes                   1.0    5.0    9.0
  [total]               7.0    6.0   17.0

tear-prod-rate
```

```
  reduced                1.0    1.0    13.0
  normal                 6.0    5.0     4.0
  [total]                7.0    6.0    17.0



Time taken to build model: 0 seconds

=== Predictions on test split ===

    inst#     actual  predicted error prediction
        1     3:none     3:none       0.877
        2     3:none     3:none       0.86
        3     2:hard     3:none   +   0.576
        4     1:soft     3:none   +   0.359
        5     1:soft     3:none   +   0.46

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances           2               40      %
Incorrectly Classified Instances         3               60      %
Kappa statistic                          0
Mean absolute error                      0.3052
Root mean squared error                  0.3889
Relative absolute error                 71.9467 %
Root relative squared error             78.1117 %
Total Number of Instances                5

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.000    0.000    ?          0.000    ?          ?        1.000     1.000     soft
               0.000    0.000    ?          0.000    ?          ?        1.000     1.000     hard
               1.000    1.000    0.400      1.000    0.571      ?        1.000     1.000     none

=== Confusion Matrix ===

 a b c   <-- classified as
 0 0 2 | a = soft
 0 0 1 | b = hard
 0 0 2 | c = none
```

Explanation:

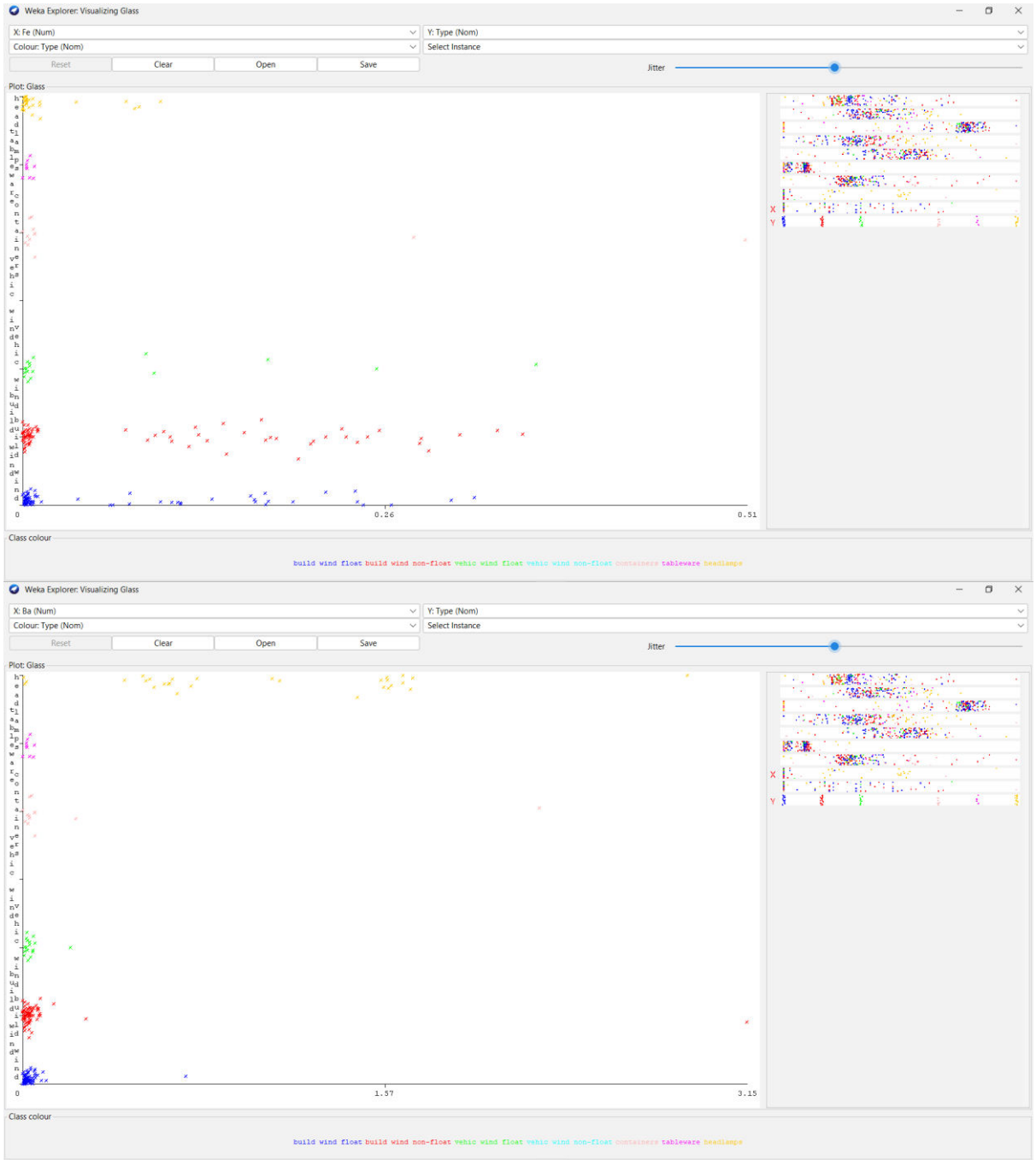
There are 5 instances in the test data, out of which 2 instances were predicted correctly by the model and other 3 were wrongly predicted. All the predictions by the model were 'none'. The model was not able to predict any other class.

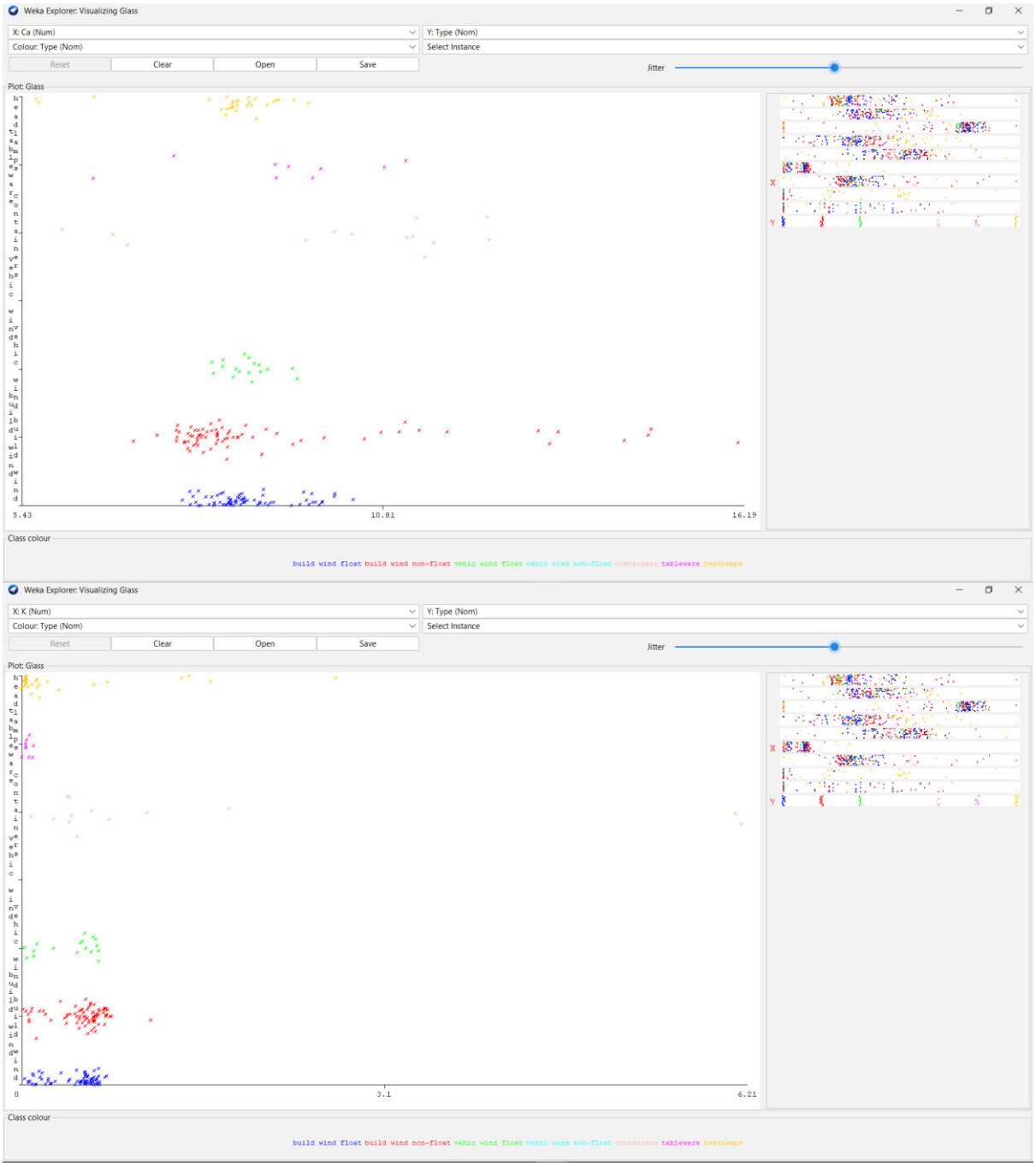
V. Accuracy percentage: 40%

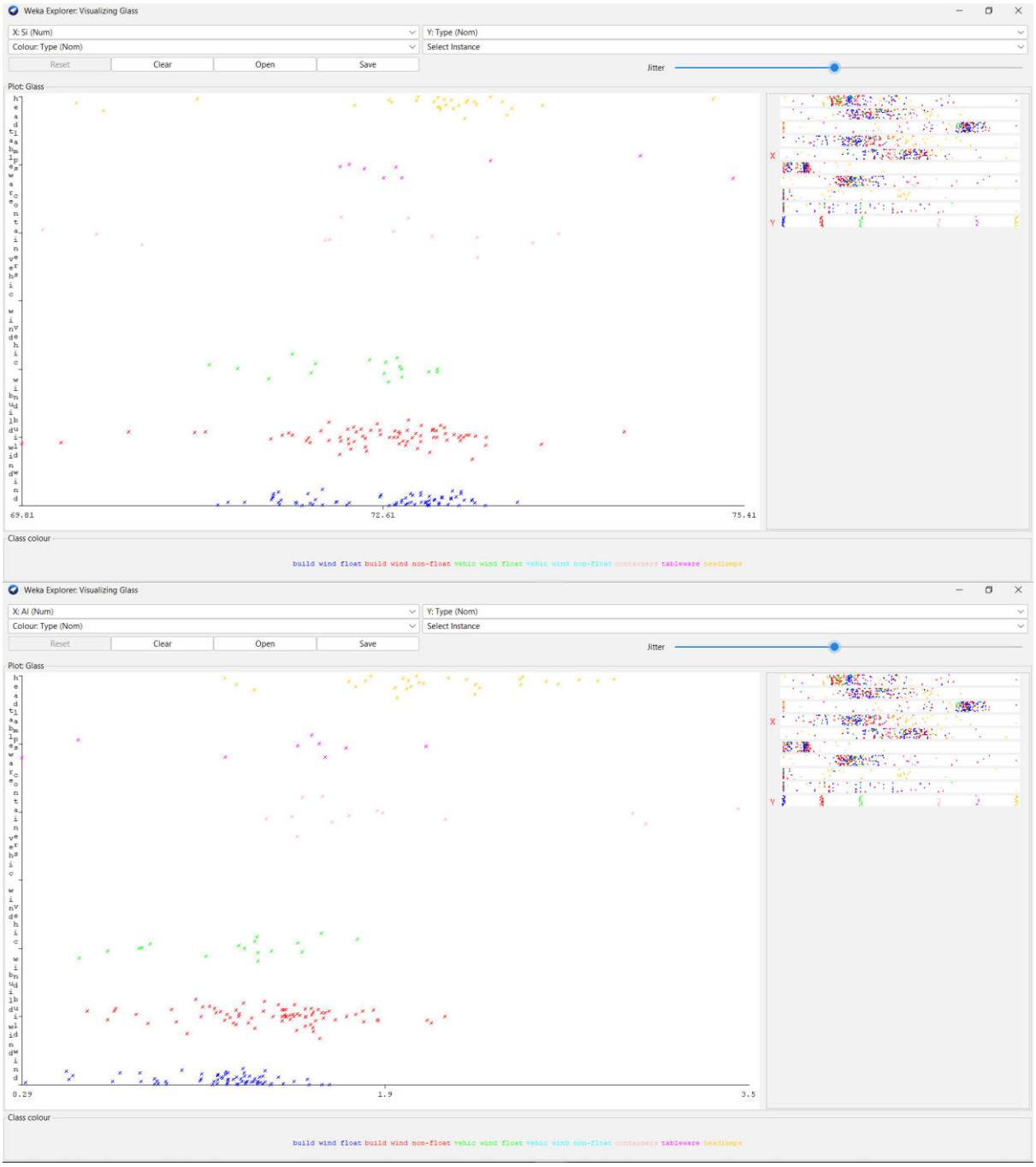
Part – 2

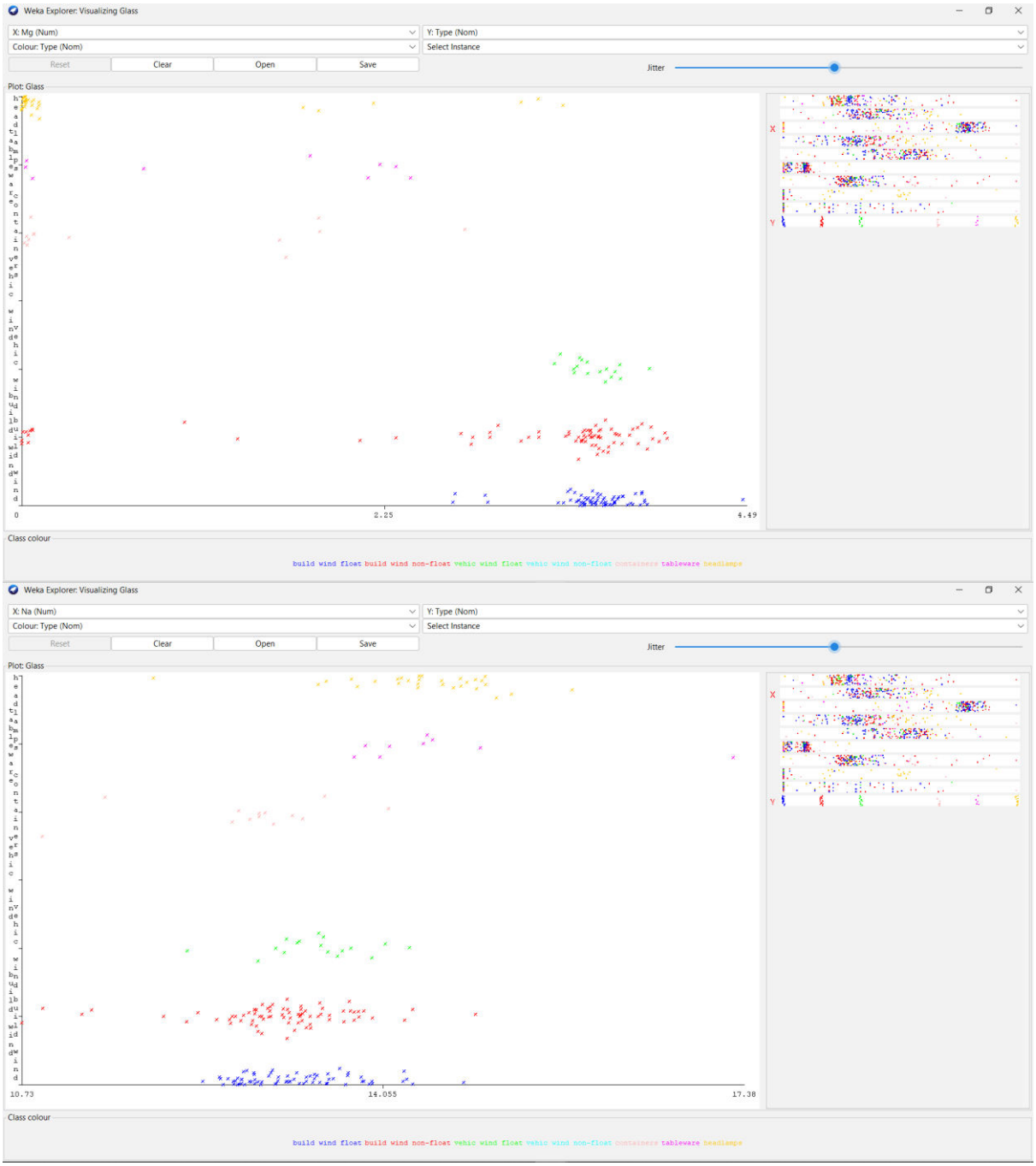
“RandomForest” analysis on “Glass.arff” dataset:

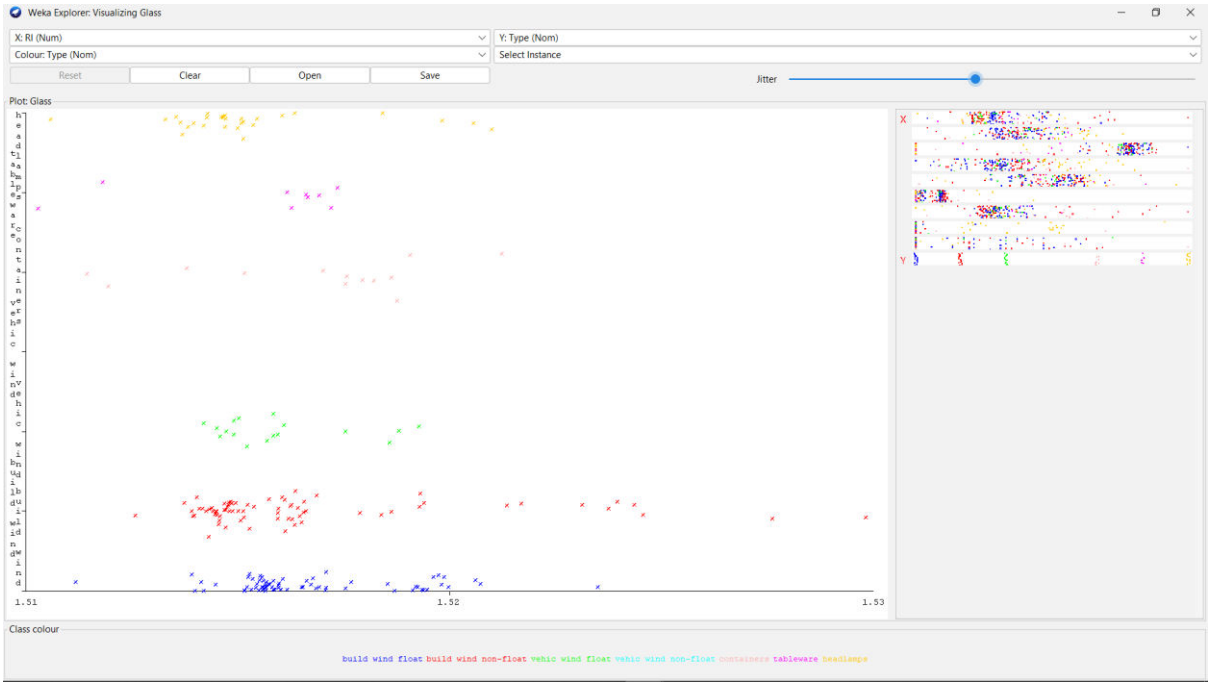
1. Loaded the dataset into WEKA explorer.
2. Class attribute in the dataset: “Type”, which indicates the type of the glass.
   Data visualizations:




Weka Explorer: Visualizing Glass
X: Ca (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
refractive index
sodium
magnesium
aluminum
silicon
potassium
calcium
barium
iron
5.43
10.81
16.19
X
Y
Class colour
build wind float
build wind non-float
vehic wind float
vehic wind non-float
containers
tableware
headlamps
Weka Explorer: Visualizing Glass
X: K (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
refractive index
sodium
magnesium
aluminum
silicon
potassium
calcium
barium
iron
0
3.1
6.21
X
Y
Class colour
build wind float
build wind non-float
vehic wind float
vehic wind non-float
containers
tableware
headlamps


Weka Explorer: Visualizing Glass
X: Si (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
69.81
72.61
75.41
Class colour
build wind float build wind non-float vehic wind float vehic wind non-float containers tableware headlamps
Weka Explorer: Visualizing Glass
X: Al (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
0.29
1.9
3.5
Class colour
build wind float build wind non-float vehic wind float vehic wind non-float containers tableware headlamps


Weka Explorer: Visualizing Glass
X: Mg (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
0
2.25
4.49
X
Y
Class colour
build wind float build wind non-float vehic wind float vehic wind non-float containers tableware headlamps
Weka Explorer: Visualizing Glass
X: Na (Num)
Y: Type (Nom)
Colour: Type (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Glass
10.73
14.055
17.38
X
Y
Class colour
build wind float build wind non-float vehic wind float vehic wind non-float containers tableware headlamps



3. Chose “RandomForest” classifier.
4. In the GenericObjectEditor window, following values are set:
   - Seed = 123
   - numIterations = 200
   - numFeatures = 4
5. Model output:

   === Run information ===

   Scheme: weka.classifiers.trees.RandomForest -P 100 -I 200 -num-slots 1 -K 4 -M 1.0 -V 0.001 -S 123
   Relation: Glass
   Instances: 214
   Attributes: 10
   RI
   Na
   Mg
   Al
   Si
   K
   Ca
   Ba
   Fe
   Type
   Test mode: 10-fold cross-validation

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

   RandomForest

   Bagging with 200 iterations and base learner

weka.classifiers.trees.RandomTree -K 4 -M 1.0 -V 0.001 -S 123 -do-not-check-capabilities

Time taken to build model: 0.21 seconds

=== Predictions on test data ===

inst#	actual	predicted	error	prediction
1	3:'vehic wind float'	1:build wind float	+	0.895
2	3:'vehic wind float'	3:vehic wind float		0.62
3	2:'build wind non-float'	2:build wind non-float		0.86
4	2:'build wind non-float'	2:build wind non-float		0.96
5	2:'build wind non-float'	1:build wind float	+	0.545
6	2:'build wind non-float'	2:build wind non-float		0.965
7	2:'build wind non-float'	2:build wind non-float		0.76
8	2:'build wind non-float'	2:build wind non-float		0.88
9	2:'build wind non-float'	2:build wind non-float		0.825
10	2:'build wind non-float'	3:vehic wind float	+	0.35
11	1:'build wind float'	1:build wind float		0.97
12	1:'build wind float'	1:build wind float		0.87
13	1:'build wind float'	1:build wind float		0.955
14	1:'build wind float'	1:build wind float		0.72
15	1:'build wind float'	1:build wind float		0.77
16	1:'build wind float'	1:build wind float		0.715
17	1:'build wind float'	1:build wind float		0.95
18	7:headlamps	2:build wind non-float	+	0.365
19	7:headlamps	7:headlamps		0.655
20	7:headlamps	2:build wind non-float	+	0.555
21	6:tableware	6:tableware		0.66
22	5:containers	5:containers		0.785
1	3:'vehic wind float'	3:vehic wind float		0.36
2	3:'vehic wind float'	2:build wind non-float	+	0.47
3	2:'build wind non-float'	2:build wind non-float		0.835
4	2:'build wind non-float'	2:build wind non-float		0.83
5	2:'build wind non-float'	1:build wind float	+	0.405
6	2:'build wind non-float'	2:build wind non-float		0.715
7	2:'build wind non-float'	2:build wind non-float		0.785
8	2:'build wind non-float'	2:build wind non-float		0.805
9	2:'build wind non-float'	2:build wind non-float		0.795
10	2:'build wind non-float'	2:build wind non-float		0.8
11	1:'build wind float'	2:build wind non-float	+	0.61
12	1:'build wind float'	1:build wind float		0.955
13	1:'build wind float'	1:build wind float		0.525
14	1:'build wind float'	1:build wind float		0.85
15	1:'build wind float'	1:build wind float		0.475
16	1:'build wind float'	1:build wind float		0.47
17	1:'build wind float'	2:build wind non-float	+	0.94
18	7:headlamps	7:headlamps		0.955

19 7:headlamps 7:headlamps 0.995
20 7:headlamps 7:headlamps 0.905
21 5:containers 5:containers 0.81
22 5:containers 5:containers 0.895
1 3:'vehic wind float' 1:build wind float + 0.34
2 3:'vehic wind float' 3:vehic wind float 0.495
3 2:'build wind non-float' 1:build wind float + 0.475
4 2:'build wind non-float' 2:build wind non-float 0.49
5 2:'build wind non-float' 2:build wind non-float 0.6
6 2:'build wind non-float' 2:build wind non-float 0.565
7 2:'build wind non-float' 6:tableware + 0.545
8 2:'build wind non-float' 2:build wind non-float 0.765
9 2:'build wind non-float' 2:build wind non-float 0.94
10 2:'build wind non-float' 1:build wind float + 0.725
11 1:'build wind float' 1:build wind float 0.845
12 1:'build wind float' 1:build wind float 0.72
13 1:'build wind float' 1:build wind float 0.965
14 1:'build wind float' 1:build wind float 0.91
15 1:'build wind float' 1:build wind float 0.895
16 1:'build wind float' 1:build wind float 0.61
17 1:'build wind float' 1:build wind float 0.565
18 7:headlamps 7:headlamps 0.995
19 7:headlamps 7:headlamps 0.845
20 6:tableware 6:tableware 0.84
21 5:containers 5:containers 0.765
22 5:containers 5:containers 0.835
1 3:'vehic wind float' 3:vehic wind float 0.71
2 3:'vehic wind float' 1:build wind float + 0.805
3 2:'build wind non-float' 6:tableware + 0.38
4 2:'build wind non-float' 2:build wind non-float 0.58
5 2:'build wind non-float' 2:build wind non-float 0.85
6 2:'build wind non-float' 2:build wind non-float 0.98
7 2:'build wind non-float' 2:build wind non-float 0.46
8 2:'build wind non-float' 2:build wind non-float 0.805
9 2:'build wind non-float' 2:build wind non-float 0.905
10 1:'build wind float' 2:build wind non-float + 0.54
11 1:'build wind float' 2:build wind non-float + 0.655
12 1:'build wind float' 1:build wind float 0.48
13 1:'build wind float' 1:build wind float 0.59
14 1:'build wind float' 1:build wind float 0.455
15 1:'build wind float' 1:build wind float 0.835
16 1:'build wind float' 1:build wind float 0.76
17 7:headlamps 7:headlamps 1
18 7:headlamps 7:headlamps 0.995
19 7:headlamps 7:headlamps 1
20 6:tableware 6:tableware 0.805
21 5:containers 5:containers 0.47
22 5:containers 7:headlamps + 0.655

1 3:'vehic wind float' 1:build wind float + 0.66
2 3:'vehic wind float' 1:build wind float + 0.72
3 2:'build wind non-float' 2:build wind non-float 0.825
4 2:'build wind non-float' 2:build wind non-float 0.38
5 2:'build wind non-float' 2:build wind non-float 0.9
6 2:'build wind non-float' 2:build wind non-float 0.975
7 2:'build wind non-float' 2:build wind non-float 0.82
8 2:'build wind non-float' 2:build wind non-float 0.51
9 2:'build wind non-float' 5:containers + 0.415
10 1:'build wind float' 1:build wind float 0.815
11 1:'build wind float' 1:build wind float 0.61
12 1:'build wind float' 1:build wind float 0.89
13 1:'build wind float' 1:build wind float 0.9
14 1:'build wind float' 1:build wind float 0.56
15 1:'build wind float' 1:build wind float 0.88
16 1:'build wind float' 1:build wind float 0.895
17 7:headlamps 7:headlamps 0.99
18 7:headlamps 7:headlamps 0.715
19 7:headlamps 7:headlamps 0.985
20 6:tableware 6:tableware 0.4
21 5:containers 5:containers 0.755
1 3:'vehic wind float' 3:vehic wind float 0.465
2 3:'vehic wind float' 2:build wind non-float + 0.83
3 2:'build wind non-float' 2:build wind non-float 0.97
4 2:'build wind non-float' 2:build wind non-float 0.565
5 2:'build wind non-float' 2:build wind non-float 0.81
6 2:'build wind non-float' 2:build wind non-float 0.95
7 2:'build wind non-float' 2:build wind non-float 0.84
8 2:'build wind non-float' 2:build wind non-float 0.97
9 2:'build wind non-float' 2:build wind non-float 0.635
10 1:'build wind float' 1:build wind float 0.855
11 1:'build wind float' 1:build wind float 0.76
12 1:'build wind float' 1:build wind float 0.81
13 1:'build wind float' 3:vehic wind float + 0.555
14 1:'build wind float' 1:build wind float 0.655
15 1:'build wind float' 1:build wind float 0.835
16 1:'build wind float' 2:build wind non-float + 0.75
17 7:headlamps 7:headlamps 1
18 7:headlamps 7:headlamps 0.395
19 7:headlamps 7:headlamps 0.71
20 6:tableware 6:tableware 0.695
21 5:containers 5:containers 0.47
1 3:'vehic wind float' 1:build wind float + 0.495
2 3:'vehic wind float' 1:build wind float + 0.445
3 2:'build wind non-float' 1:build wind float + 0.42
4 2:'build wind non-float' 2:build wind non-float 0.47
5 2:'build wind non-float' 2:build wind non-float 0.34
6 2:'build wind non-float' 1:build wind float + 0.81

7 2:'build wind non-float' 2:build wind non-float 0.69
8 2:'build wind non-float' 1:build wind float + 0.67
9 2:'build wind non-float' 1:build wind float + 0.62
10 1:'build wind float' 1:build wind float 0.89
11 1:'build wind float' 1:build wind float 0.985
12 1:'build wind float' 1:build wind float 0.54
13 1:'build wind float' 1:build wind float 0.825
14 1:'build wind float' 1:build wind float 0.765
15 1:'build wind float' 1:build wind float 0.7
16 1:'build wind float' 1:build wind float 0.585
17 7:headlamps 7:headlamps 0.845
18 7:headlamps 7:headlamps 0.965
19 7:headlamps 7:headlamps 0.755
20 6:tableware 6:tableware 0.47
21 5:containers 5:containers 0.76
1 3:'vehic wind float' 1:build wind float + 0.565
2 2:'build wind non-float' 2:build wind non-float 0.905
3 2:'build wind non-float' 1:build wind float + 0.615
4 2:'build wind non-float' 2:build wind non-float 0.77
5 2:'build wind non-float' 2:build wind non-float 0.915
6 2:'build wind non-float' 2:build wind non-float 0.92
7 2:'build wind non-float' 7:headlamps + 0.34
8 2:'build wind non-float' 2:build wind non-float 0.52
9 2:'build wind non-float' 1:build wind float + 0.555
10 1:'build wind float' 2:build wind non-float + 0.75
11 1:'build wind float' 1:build wind float 0.485
12 1:'build wind float' 1:build wind float 0.985
13 1:'build wind float' 1:build wind float 0.56
14 1:'build wind float' 1:build wind float 0.77
15 1:'build wind float' 1:build wind float 0.56
16 1:'build wind float' 1:build wind float 0.87
17 7:headlamps 7:headlamps 1
18 7:headlamps 7:headlamps 0.9
19 7:headlamps 7:headlamps 0.99
20 6:tableware 6:tableware 0.655
21 5:containers 5:containers 0.52
1 3:'vehic wind float' 3:vehic wind float 0.405
2 2:'build wind non-float' 2:build wind non-float 0.715
3 2:'build wind non-float' 5:containers + 0.595
4 2:'build wind non-float' 2:build wind non-float 0.9
5 2:'build wind non-float' 2:build wind non-float 0.605
6 2:'build wind non-float' 2:build wind non-float 0.725
7 2:'build wind non-float' 2:build wind non-float 0.805
8 2:'build wind non-float' 2:build wind non-float 0.49
9 2:'build wind non-float' 2:build wind non-float 0.615
10 1:'build wind float' 1:build wind float 0.61
11 1:'build wind float' 1:build wind float 0.755
12 1:'build wind float' 1:build wind float 0.815

13 1:'build wind float' 1:build wind float 0.505
14 1:'build wind float' 1:build wind float 0.69
15 1:'build wind float' 1:build wind float 0.705
16 1:'build wind float' 1:build wind float 0.735
17 7:headlamps 7:headlamps 0.975
18 7:headlamps 2:build wind non-float + 0.455
19 7:headlamps 7:headlamps 0.995
20 6:tableware 2:build wind non-float + 0.47
21 5:containers 5:containers 0.515
1 3:'vehic wind float' 1:build wind float + 0.75
2 2:'build wind non-float' 2:build wind non-float 0.87
3 2:'build wind non-float' 2:build wind non-float 0.655
4 2:'build wind non-float' 2:build wind non-float 0.77
5 2:'build wind non-float' 2:build wind non-float 0.925
6 2:'build wind non-float' 1:build wind float + 0.44
7 2:'build wind non-float' 2:build wind non-float 0.71
8 2:'build wind non-float' 3:vehic wind float + 0.365
9 2:'build wind non-float' 2:build wind non-float 0.375
10 1:'build wind float' 1:build wind float 0.865
11 1:'build wind float' 1:build wind float 0.925
12 1:'build wind float' 1:build wind float 0.56
13 1:'build wind float' 1:build wind float 0.635
14 1:'build wind float' 1:build wind float 0.93
15 1:'build wind float' 1:build wind float 0.94
16 1:'build wind float' 1:build wind float 0.96
17 7:headlamps 1:build wind float + 0.73
18 7:headlamps 7:headlamps 1
19 7:headlamps 2:build wind non-float + 0.445
20 6:tableware 6:tableware 0.765
21 5:containers 5:containers 0.755

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

Correctly Classified Instances	171	79.9065 %
Incorrectly Classified Instances	43	20.0935 %
Kappa statistic	0.724	
Mean absolute error	0.0996	
Root mean squared error	0.2112	
Relative absolute error	47.0438 %	
Root relative squared error	65.0775 %	
Total Number of Instances	214	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.900	0.146	0.750	0.900	0.818	0.725	0.936	0.875	build wind float

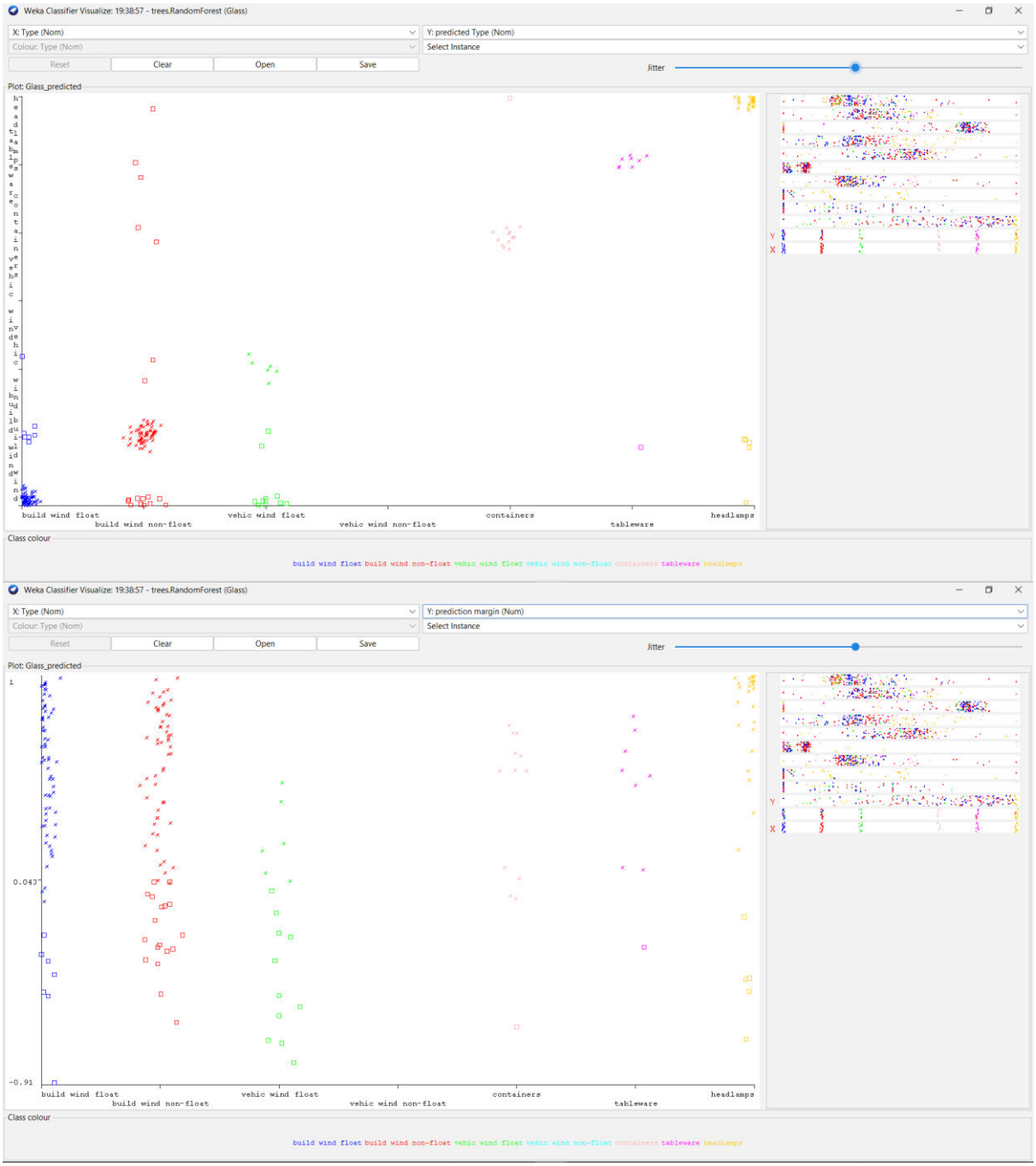
0.763	0.094	0.817	0.763	0.789	0.680	0.926	0.863	build wind non-float
0.353	0.015	0.667	0.353	0.462	0.455	0.930	0.544	vehic wind float
?	0.000	?	?	?	?	?	?	vehic wind non-float
0.923	0.010	0.857	0.923	0.889	0.882	0.981	0.913	containers
0.889	0.010	0.800	0.889	0.842	0.836	0.997	0.940	tableware
0.828	0.011	0.923	0.828	0.873	0.856	0.974	0.922	headlamps
Weighted Avg.	0.799	0.085	0.799	0.799	0.792	0.719	0.942	0.856

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
63	6	1	0	0	0	0	a = build wind float
11	58	2	0	2	2	1	b = build wind non-float
9	2	6	0	0	0	0	c = vehic wind float
0	0	0	0	0	0	0	d = vehic wind non-float
0	0	0	0	12	0	1	e = containers
0	1	0	0	0	8	0	f = tableware
1	4	0	0	0	0	24	g = headlamps

Accuracy of the model = 79.9065%

6. Visualizing classifier errors:



Here the squares represent the wrong predictions made by the Random Forest model.

7. Random Forest analysis using R:
   Confusion Matrix and Statistics

	Reference					
Prediction	build wind float	build wind non-float	vehic wind float	containers	tableware	headlamps
build wind float	62	9	8	0	0	1
build wind non-float	6	59	3	2	2	2
vehic wind float	2	3	6	0	0	0
containers	0	2	0	10	0	0
tableware	0	2	0	0	7	0
headlamps	0	1	0	1	0	26

Overall Statistics

Accuracy : 0.7944
95% CI : (0.734, 0.8465)
No Information Rate : 0.3551
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7175

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: build wind float	Class: build wind non-float	Class: vehic wind float	Class: containers	Class: tableware	Class: headlamps
Sensitivity	0.8857	0.7763	0.35294	0.76923	0.77778	0.8966
Specificity	0.8750	0.8913	0.97462	0.99005	0.99024	0.9892
Pos Pred Value	0.7750	0.7973	0.54545	0.83333	0.77778	0.9286
Neg Pred Value	0.9403	0.8786	0.94581	0.98515	0.99024	0.9839
Prevalence	0.3271	0.3551	0.07944	0.06075	0.04206	0.1355
Detection Rate	0.2897	0.2757	0.02804	0.04673	0.03271	0.1215
Detection Prevalence	0.3738	0.3458	0.05140	0.05607	0.04206	0.1308
Balanced Accuracy	0.8804	0.8338	0.66378	0.87964	0.88401	0.9429

Here we are getting an accuracy of 79.44% which slightly lower than that of WEKA.

Section I:

Part I:

```
#load your data

df <- data(iris)


# look into data structure

head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

```
str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
dim(iris)
[1] 150   5


# Generate a random sample of all data
# in this case 82% of the  dataset.
randSelection <- sample(1:nrow(iris), 0.82 * nrow(iris))
randSelection



# data normalization f
normalization <-function(x) {(x -min(x))/(max(x)-min(x))}
```

```
# Run nomalization on on coulumns which are the predictors
irisNormalized <- as.data.frame(lapply(iris[,c(1:4)], normalization))
summary(irisNormalized)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.00000
1st Qu.:0.2222	1st Qu.:0.3333	1st Qu.:0.1017	1st Qu.:0.08333
Median :0.4167	Median :0.4167	Median :0.5678	Median :0.50000
Mean :0.4287	Mean :0.4406	Mean :0.4675	Mean :0.45806
3rd Qu.:0.5833	3rd Qu.:0.5417	3rd Qu.:0.6949	3rd Qu.:0.70833
Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.00000

```
## seperate data into training and testing to #check model accuracy
#  get training data
training <- irisNormalized[randSelection,]
nrow(training)
```

[1] 122

```
# get testing data
testing <- irisNormalized[-randSelection,]
nrow(testing)
```

[1] 28

```
# obtain the class label of train dataset because as it will #be used as argument in knn classifier
targertClass <- iris[randSelection,5]
targertClass
summary(targertClass)
```

setosa	versicolor	virginica
39	43	40

```
# extract 5th column if test dataset to measure the #accuracy
testClass <- iris[-randSelection,5]
summary(testClass)
```

setosa	versicolor	virginica
11	7	10

```
library(class)
# building the model for classification
# run knn classifier
# here we use k = 10

classificationModel <-  knn(training,testing,cl=targertClass,k=10)
classificationModel
```

```
#create confusion matrix to check model # performance
ConfMatrix <- table(classificationModel,testClass)
ConfMatrix
```

	testClass		
classificationModel	setosa	versicolor	virginica
setosa	11	0	0
versicolor	0	7	2
virginica	0	0	8

```
#Calculate model accuracy
modelAccuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
modelAccuracy(ConfMatrix)
```

[1] 92.85714

Part II:

a) Download any dataset from BB or from the following websites:
   - https://archive.ics.uci.edu/ml/datasets.php
   - https://www.kaggle.com/datasets

I have downloaded the diabetes dataset from Kaggle.

b) Provide explanation of why you choose this specific dataset. And what are the predictors variables and the response variable? **(3 points)**

I have chosen this dataset as it gives us insights into the medical world, and also helps us to create a model that will help with the diagnostics of diabetes disease.

c) Load the dataset in RStudio. **(2 points)**

```
# Importing the required libraries and importing the dataset
library(readr)
df = read_csv("diabetes.csv")
```

d) Show the structure, first 6 rows, and the total number of instances and attributes in the dataset. Explain the output. **(2 points)**

```
# Dataset structure & summary
dim(df)
```

[1] 767 9

No. of columns = 9

No. of rows = 767

```
head(df, 6)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.29	33	1
6	5	116	74	0	0	25.6	0.201	30	0

```
summary(df)
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Min.: 0.000	Min.: 0.0	Min.: 0.0	Min.: 0.00	Min.: 0.0	Min.: 0.00	Min.: 0.0780	Min.: 21.00	Min.: 0.0000
1st Qu.: 1.000	1st Qu.: 99.0	1st Qu.: 62.0	1st Qu.: 0.00	1st Qu.: 0.0	1st Qu.:27.30	1st Qu.:0.2435	1st Qu.:24.00	1st Qu.:0.0000
Median: 3.000	Median :117.0	Median: 72.0	Median :23.00	Median :32.0	Median :32.00	Median :0.3740	Median :29.00	Median :0.0000
Mean: 3.849	Mean: 120.9	Mean: 69.1	Mean:20.52	Mean:79.9	Mean:31.99	Mean:0.4721	Mean:33.25	Mean:0.3494
3rd Qu.: 6.000	3rd Qu.:140.5	3rd Qu.: 80.0	3rd Qu.:32.00	3rd Qu.:127.5	3rd Qu.:36.60	3rd Qu.:0.6265	3rd Qu.:41.00	3rd Qu.:1.0000
Max. :17.000	Max. :199.0	Max. :122.0	Max. :99.00	Max. :846.0	Max. :67.10	Max. :2.4200	Max. :81.00	Max. :1.0000

e) Check for null values. Are there any null values? **(2 points)**

```
# Missing value handling
colSums(is.na(df))
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0	0	0	0	0	0	0	0

There are no null values in the dataset

f) Check if you need to change the datatype of any of the attributes. **(2 points)**

```
# Datatype of the attributes
str(df)
```

```
spec_tbl_df [767 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Pregnancies           : num [1:767] 6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose               : num [1:767] 148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure         : num [1:767] 72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness         : num [1:767] 35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin               : num [1:767] 0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                   : num [1:767] 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num [1:767] 0.627 0.351 0.672 0.167 2.288 ...
 $ Age                   : num [1:767] 50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome               : num [1:767] 1 0 1 0 1 0 1 0 1 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   Pregnancies = col_double(),
  ..   Glucose = col_double(),
  ..   BloodPressure = col_double(),
  ..   SkinThickness = col_double(),
  ..   Insulin = col_double(),
  ..   BMI = col_double(),
  ..   DiabetesPedigreeFunction = col_double(),
  ..   Age = col_double(),
  ..   Outcome = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

We do not need to change the data type of any variable

g) Generate a random sample of all data (82% of the dataset). Print the sample data. **(2 points)**

```
# Generate a random sample of 82%
set.seed(1)
randSelection = sample(1:nrow(df), 0.82 * nrow(df))
df[randSelection,]
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	3	121	52	0	0	36	0.127	25	1
2	1	117	88	24	145	34.5	0.403	40	1
3	2	84	50	23	76	30.4	0.968	21	0
4	1	144	82	40	0	41.3	0.607	28	0

5	14	100	78	25	184	36.6	0.412	46	1
6	2	146	0	0	0	27.5	0.24	28	1
7	8	181	68	36	495	30.1	0.615	60	1
8	10	161	68	23	132	25.5	0.326	47	1
9	0	67	76	0	0	45.3	0.194	46	0
10	7	106	60	24	0	26.5	0.296	29	1

h) Apply data normalization and show the first 6 rows after normalization. Explain what is the reason we use normalization? **(2 points)**

```
# data normalization function
normalization <-function(x) { (x -min(x))/(max(x)-min(x))   }
# Run nomalization on on columns which are the predictors
dfNormalized <- as.data.frame(lapply(df[,c(1:8)], normalization))
head(df, 6)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.29	33	1
6	5	116	74	0	0	25.6	0.201	30	0

i) Separate the data into training and testing sets. How many rows you have in the training and testing sets? **(2 points)**

```
#  get training data
training <- dfNormalized[randSelection,]
nrow(training)
```

There are 628 rows in the training set

```
# get testing data
testing <- dfNormalized[-randSelection,]
nrow(testing)
```

There are 139 rows in the training set

j) Obtain the class label of the training set because as it will be used as the argument in the KNN classifier. What is the column number that obtain the class label? **(2 points)**

```
# seperate target class
targetClass <- df[randSelection,9]
targetClass
```

```
summary(targetClass)
```
The 9th column represents the class labels

k) Extract the class label column for the testing set to measure the accuracy.

```
# extract 9th column of test dataset to measure the accuracy
testClass <- df[-randSelection,9]
summary(testClass)
```

l) Build the KNN model. Change the number of K each time as follows to see what k gives the best accuracy. Provide explanation of you work, which K shows best accuracy? **(2 points)**

- Use k = 8
- Use k = 10

```
library(class)
# Training knn with k=8
classificationModel_8 <-  knn(training,testing,cl=targetClass$Outcome,k=8)
classificationModel_8

# Training knn with k=10
classificationModel_10 <-  knn(training,testing,cl=targetClass$Outcome,k=10)
classificationModel_10
```

m) Create confusion matrix to check the model performance. Explain the confusion matrix in term of perfection and errors. **(2 points)**

```
# Confusion Matrix  for the model with k=8
ConfMatrix_8 <- table(classificationModel_8,testClass$Outcome)
```

	0	1
0	78	17
1	18	26

```
# Confusion Matrix for the model with k=10
ConfMatrix_10 <- table(classificationModel_10,testClass$Outcome)
```

	0	1
0	83	19
1	13	24

n) Calculate the model accuracy and compare the results. **(2 points)**

```
# Accuracy
modelAccuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
modelAccuracy(ConfMatrix_8)
```
74.82014%
```
modelAccuracy(ConfMatrix_10)
```
76.97842%

Section II:

Question 1: In brief, explain the requirement of the packages and libraries mentioned below? (5 points)

```
install.packages('caTools')
install.packages('dplyr')
install.packages('ggplot2')
install.packages('class')
install.packages('caret')
install.packages('corrplot')
library(class)
library(corrplot)
library(caTools)
library(dplyr)
library(ggplot2)
library(caret)
```

The library 'class' is required for applying the KNN algorithm on the dataset.

The library 'corrplot' is required for visualizing the correlation plots.

The library 'caTools' is required for converting binary data into string using the ASCII convention.

The library 'dplyr' is required for common data manipulations like null values, etc.

The library 'ggplot2' is required for data visualizations.

The library 'caret' is required for classification and regression.

Question 2: What do you understand by standardization in KNN? What is the need to scale the data in KNN? Present your inference and the results of scaling. (5 points)

The k-nearest neighbor algorithm relies on majority voting based on class membership of 'k' nearest samples for a given test point. The nearness of samples is typically based on Euclidean distance. Without normalization, all the nearest neighbors are aligned in the direction of the axis with the smaller range, i.e. x1 leading to incorrect classification. Normalization solves this problem!

Question 3: What does each line of the code below do after you run it? And what are the results? (5 points)

```
data <- cbind(standard.features,glass[10])
```

```
anyNA(data)
```

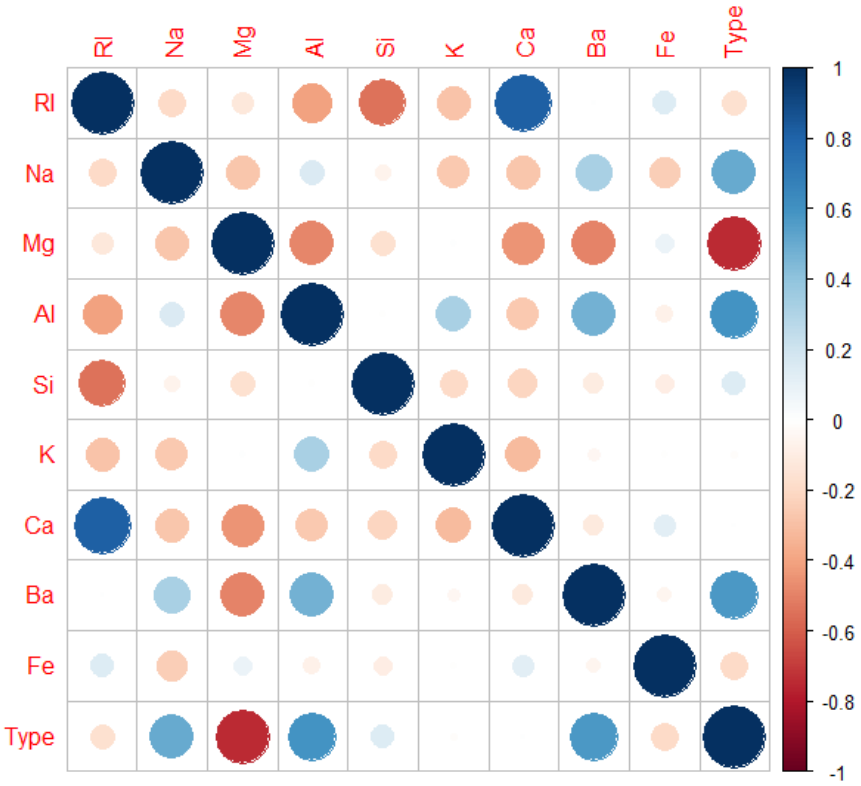
These lines tell us if the dataset has any null values or not. If there are no null values, it returns False

```
[1] FALSE
```

Question 4: Present the plot obtained. What do you understand from the plot? (5 points)

```
head(data)
```

```
corrplot(cor(data))
```



We understand from this plot that Na, Al, Ba have high positive corelation with Type, whereas Mg has high negative correlation.

Question 5: Execute the code to present the confusion matrix and model's accuracy. (5 points)

```
set.seed(101)
```

```
sample <- sample.split(data$Type,SplitRatio = 0.70)
```

```
train <- subset(data,sample==TRUE)
```

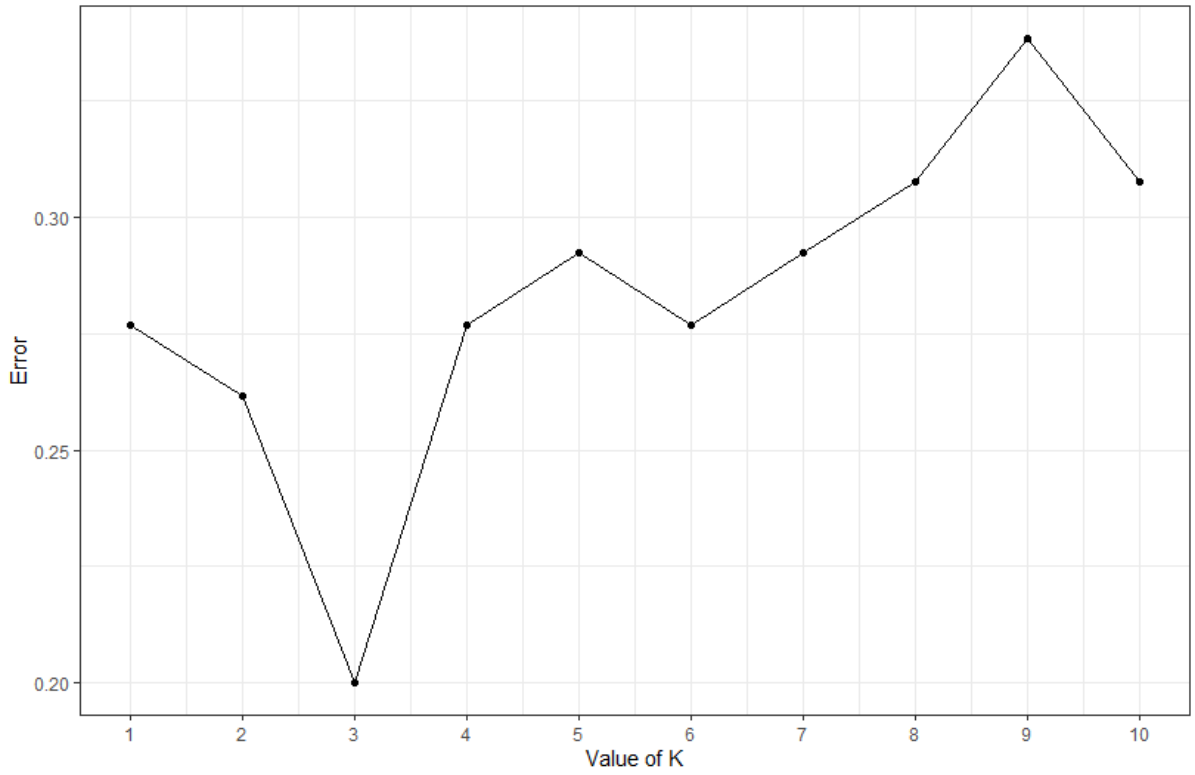
```
test <- subset(data,sample==FALSE)
```

```
predicted.type <- knn(train[1:9],test[1:9],train$Type,k=1)

error <- mean(predicted.type!=test$Type)
error
[1] 0.2769231

confusionMatrix(table(predicted.type,test$Type))

#Another method:
confusionMatrix <- table(predicted.type,test$Type)
confusionMatrix

predicted.type  1  2  3  5  6  7
             1 17  4  2  1  0  0
             2  0 16  3  0  0  1
             3  4  2  0  0  0  0
             5  0  0  0  3  0  0
             6  0  1  0  0  3  0
             7  0  0  0  0  0  8

modelAccuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
modelAccuracy(confusionMatrix)
[1] 72.30769
```

Question 6: Try different values of k and assess your model and present the accuracy obtained and compare it with previously obtained accuracy and confusion matrix. (5 points)

```
predicted.type <- NULL
error.rate <- NULL

for (i in 1:10) {
  predicted.type <- knn(train[1:9],test[1:9],train$Type,k=i)
  error.rate[i] <- mean(predicted.type!=test$Type)
```

```
}

knn.error <- as.data.frame(cbind(k=1:10,error.type =error.rate))

   ggplot(knn.error,aes(k,error.type))+ geom_point()+
geom_line() + scale_x_continuous(breaks=1:10)+ theme_bw() +
      xlab("Value of K") + ylab('Error')
```



Question 7: What is the value of K that shows the best accuracy? (5 points)

The value of K that shows the best accuracy is 3. Also observe from the graph above that the error is lowest when K=3.

Section III:

KNN Analysis on Weather Dataset in WEKA:

4. Comparison of the results:

Options	Default	K=4	K=20
Accuracy	57.1429 %	57.1429 %	64.2857 %

From the above comparison table, we can observe that, with the default settings and K=4 (Cross Validate=True, Debug=False, distanceWeighting=Weight by 1/Distance, meanSquared=True), the

performance of the KNN model remain the same, it is giving an accuracy of 57.1429 %. But after using K=20 (Cross Validate=True, Debug=False, distanceWeighting=Weight by 1/Distance, meanSquared=True), the accuracy has improved to 64.2857 %.

Section IV:

1. Analysis on “covid19.csv” dataset in RStudio:

```
library(readr)

df = read_csv("covid19.csv")


# Dataset structure & summary

str(df)
```

```
spec_tbl_df [187 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Country/Region     : chr [1:187] "Afghanistan" "Albania" "Algeria" "Andorra" ...
 $ Confirmed          : num [1:187] 36263 4880 27973 907 950 ...
 $ Deaths             : num [1:187] 1269 144 1163 52 41 ...
 $ Recovered          : num [1:187] 25198 2745 18837 803 242 ...
 $ Active             : num [1:187] 9796 1991 7973 52 667 ...
 $ New cases          : num [1:187] 106 117 616 10 18 4 4890 73 368 86 ...
 $ New deaths         : num [1:187] 10 6 8 0 1 0 120 6 6 1 ...
 $ New recovered      : num [1:187] 18 63 749 0 0 ...
 $ Confirmed last week: num [1:187] 35526 4171 23691 884 749 ...
 $ 1 week change      : num [1:187] 737 709 4282 23 201 ...
 $ 1 week % increase  : num [1:187] 2.07 17 18.07 2.6 26.84 ...
 $ WHO Region         : chr [1:187] "Eastern Mediterranean" "Europe" "Africa" "Europe" ...
 - attr(*, "spec")=
  .. cols(
  ..   `Country/Region` = col_character(),
  ..   Confirmed = col_double(),
  ..   Deaths = col_double(),
  ..   Recovered = col_double(),
  ..   Active = col_double(),
  ..   `New cases` = col_double(),
```

```
..   `New deaths` = col_double(),
..   `New recovered` = col_double(),
..   `Confirmed last week` = col_double(),
..   `1 week change` = col_double(),
..   `1 week % increase` = col_double(),
..   `WHO Region` = col_character()
.. )
 - attr(*, "problems")=<externalptr>
```

```
nrow(df)
```

```
[1] 187
```

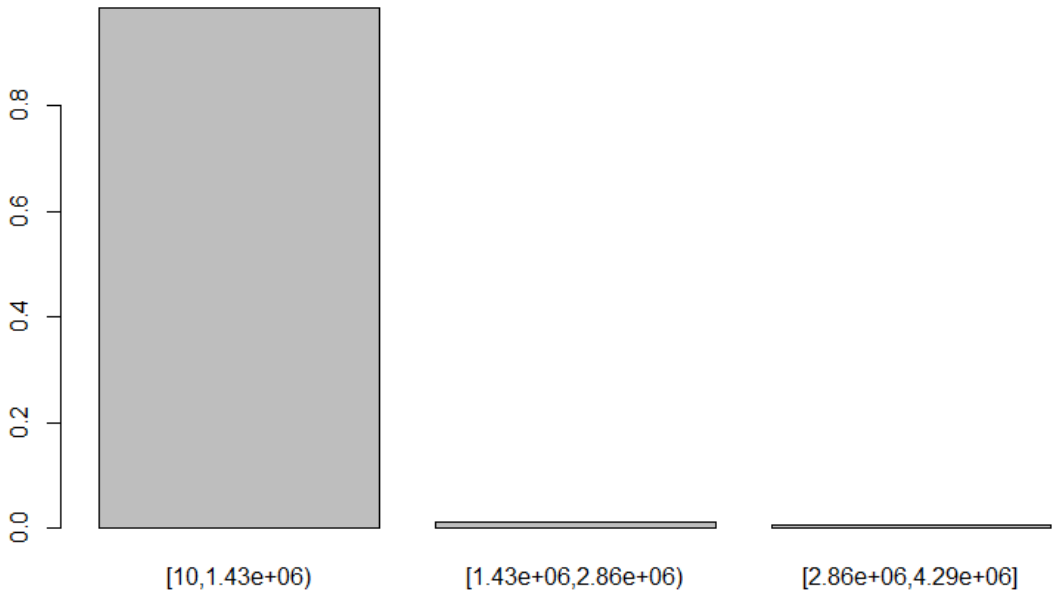
```
ncol(df)
```

```
[1] 12
```

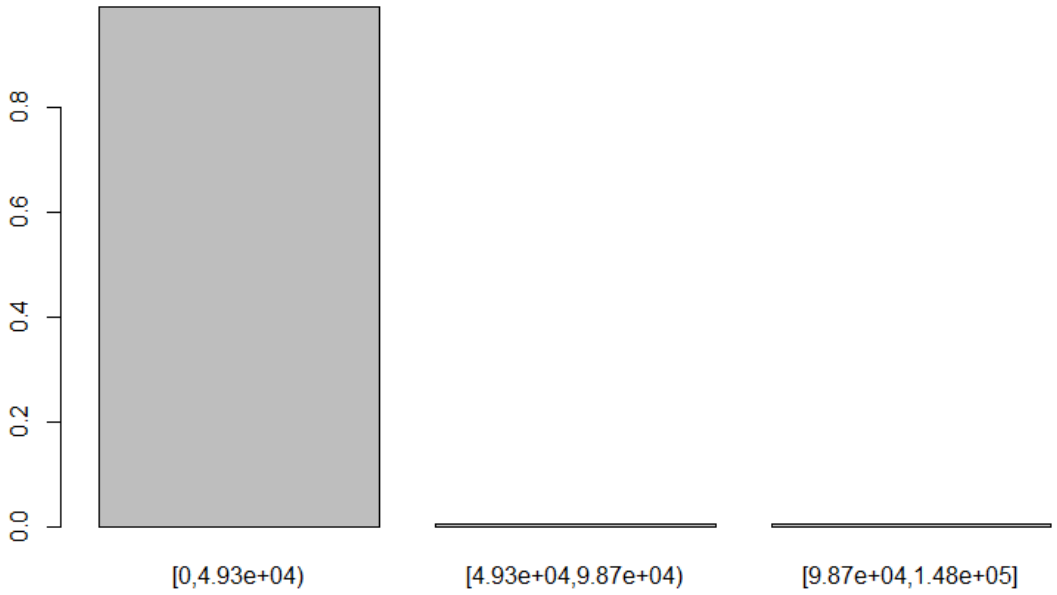
```
# Highest number of deaths
max(df$Deaths)
```

```
[1] 148011
```

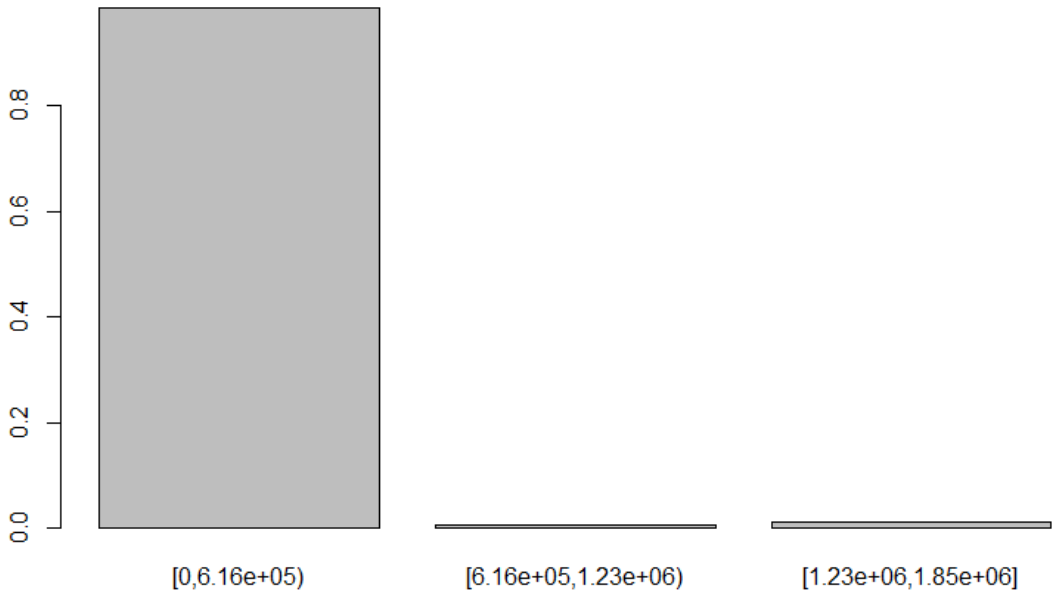
```
# Discretization
library(arules)
discretize(df$Confirmed, method = "frequency", breaks = 3)
```



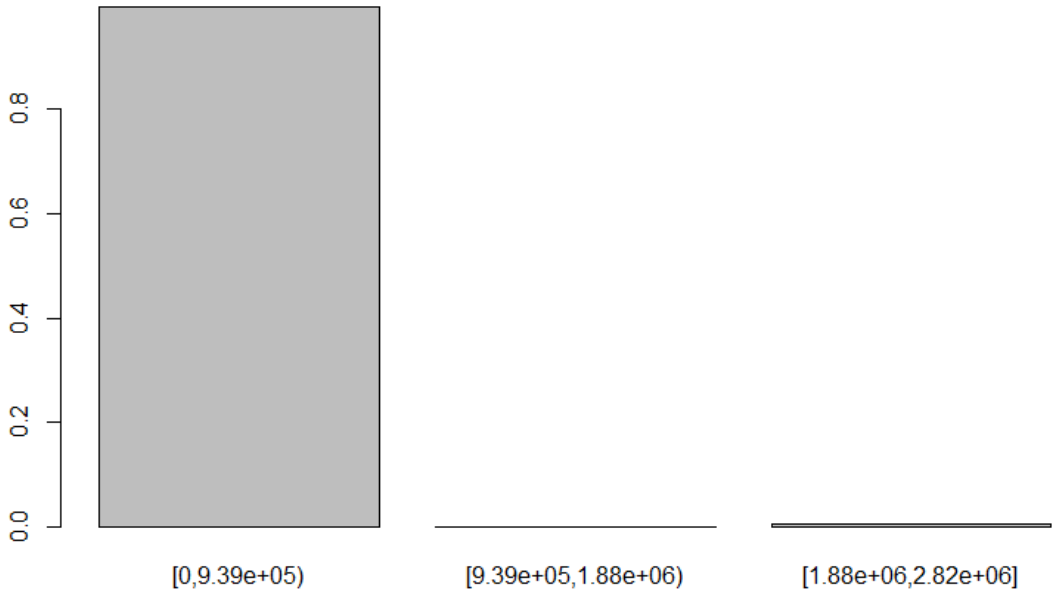
discretize(df$Deaths, method = "frequency", breaks = 3)



discretize(df$Recovered, method = "frequency", breaks = 3)

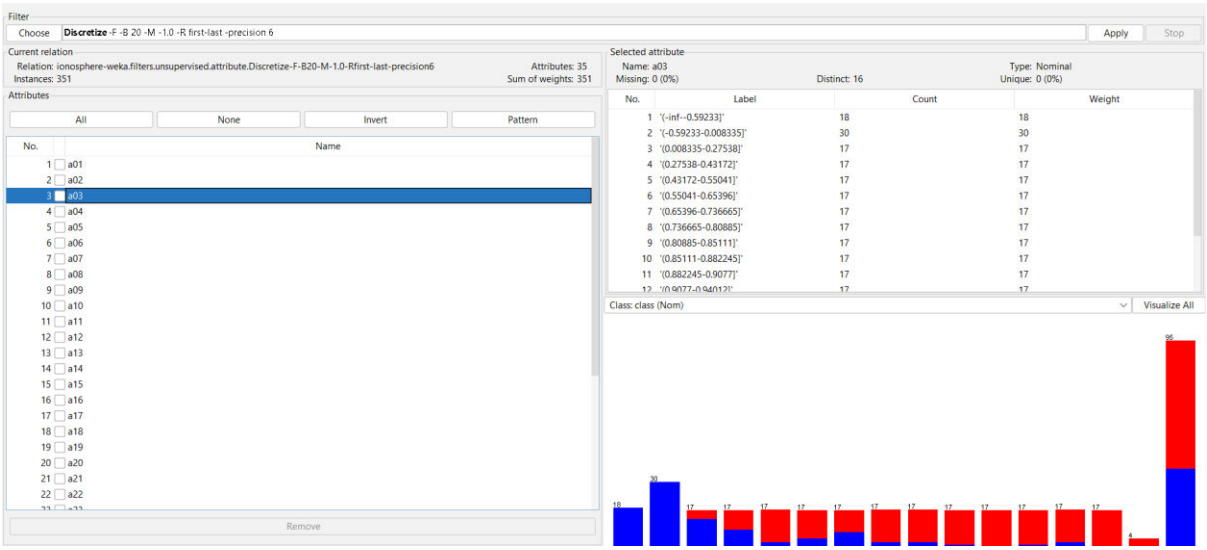


discretize(df$Active, method = "frequency", breaks = 3)



2. Analysis on “ionosphere.arff” dataset in WEKA:

After applying “discretize filter” with number of bins=20 and setting the useEqualFrequency as True, we get:



Importance of discretization:

Data discretization techniques can be used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Some machine learning algorithms that can handle both continuous and discrete attributes perform better with discrete-valued attributes.

Discretization techniques are often used by the classification algorithms.

Section 1:

1. Read the dataset in your RStudio. Explain how you created the above table into RStudio (5 points)

```
# Q1
df <- data.frame('YEAR' = character(),   #Create empty dataframe
                 '2001' = numeric(),
                 '2002' = numeric(),
                 '2003' = numeric(),
                 '2004' = numeric(),
                 '2005' = numeric(),
                 '2006' = numeric(),
                 '2007' = numeric(),
                 '2008' = numeric(),
                 '2009' = numeric(),
                 '2010' = numeric(),
                 check.names = FALSE, stringsAsFactors = FALSE)
df[1, ] <- list("Birthrate (Region 1)", 28, 30, 29, 32, 33, 40, 39, 38, 40, 35)  # Add first row
df[2, ] <- list("Birthrate (Region 2)", 23, 25, 26, 30, 32, 29, 28, 28, 31, 28)  # Add second row
df <- data.frame(df[,-1], row.names = df[,1], check.names = FALSE)
df
```

	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Birthrate (Region 1)	28	30	29	32	33	40	39	38	40	35
Birthrate (Region 2)	23	25	26	30	32	29	28	28	31	28

First, I have created a empty dataframe with column names 2001 to 2010. Then I have assigned the first row of the dataframe to be the birthrate values for region 1, and then assigned the second row of the dataframe to be the birthrate values for region 2.

2. Convert the dataframe that you created in the previous step into matrix. (5 points)

```
#Q2
df_matrix <- data.matrix(df)
df_matrix
```

	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Birthrate (Region 1)	28	30	29	32	33	40	39	38	40	35
Birthrate (Region 2)	23	25	26	30	32	29	28	28	31	28

3. Convert the matrix into a ***time series object.*** (5 points)

```
#Q3
```

```
df_timeseries <- ts(t(df_matrix), start=2001, frequency=1)
```

4. Print the time series data and explain the output. (5 points)

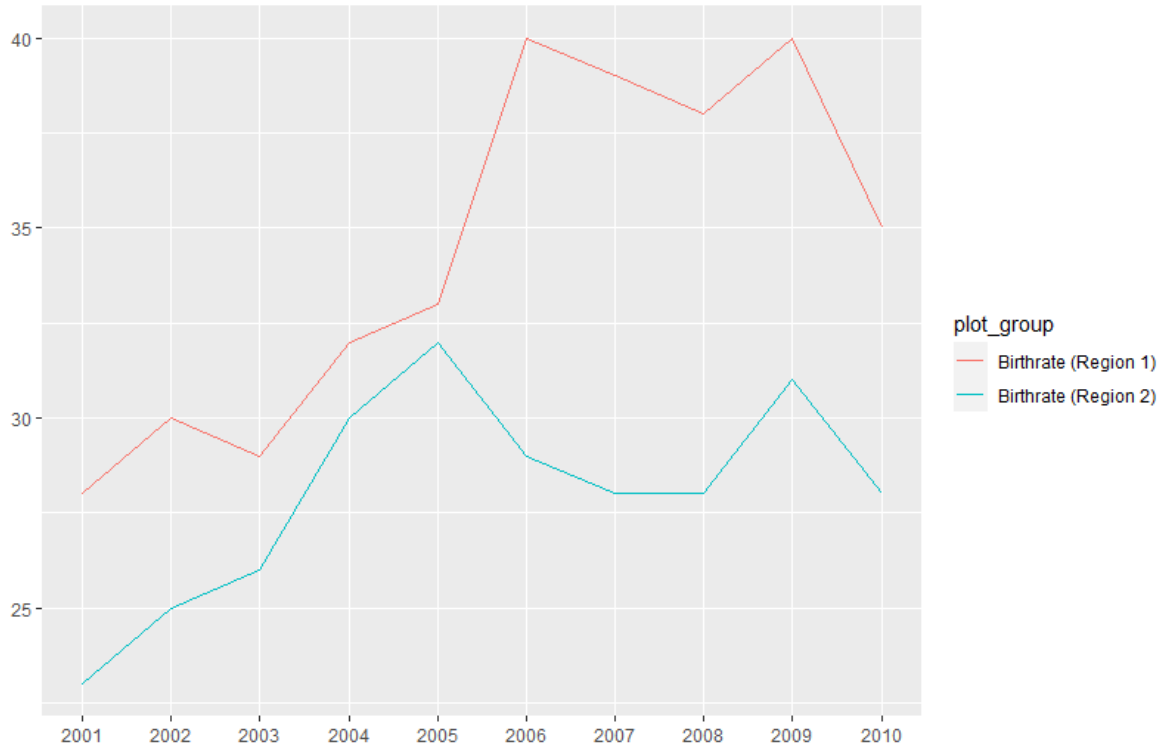
```
#Q4
df_timeseries
```

Time Series:
Start = 2001
End = 2010
Frequency = 1

	Birthrate (Region 1)	Birthrate (Region 2)
2001	28	23
2002	30	25
2003	29	26
2004	32	30
2005	33	32
2006	40	29
2007	39	28
2008	38	28
2009	40	31
2010	35	28

5. Plot the graph of the time series and present your analysis of the plot results. (5 points)

```
#Q5
library(ggplot2)
library(ggfortify)
library(fpp3)
autoplot(df_timeseries, facets = FALSE) + scale_x_continuous(breaks=c(seq(2001, 2010,1)), minor_breaks = NULL)
```



From the plot, we can see that the birthrate for region 1 is higher than the birthrate for region 2 in 2001-2010. In 2005-2010, the difference between the birthrates of these 2 regions was very high.

Section 2:

```
Library (astsa, quietly=TRUE, warn.conflicts=FALSE)
require (knitr)
```

Q.1 Why do you require knitr and astsa packages? (3 points)

The package astsa stands for Applied Statistical Time Series Analysis, and is used for dealing with time series data. Knitr package provides a general-purpose tool for dynamic report generation in R using Literate Programming techniques.

```
library(ggplot2)
kings<-scan('http://robjhyndman.com/tsdldata/misc/kings.dat', skip=3)
```

Q.2 What is your inference? What is skip = 3 here? (3 points)

Skip = 3 indicates that we have to skip the first 3 rows while reading the data. We re skipping the first 3 rows because they contain the information about the dataset, not the actual data.

```
kings
```

```
kings <- ts(kings)
```

Q.3 Explain this line of code. (3 points)

This line shows us the kings list. The output is as follows:

[1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56

Then we are converting this list into a time series object by using the ts() function.

```
kings
```

Q.4 Present the start, end and frequency of the time series obtained. (3 points)

The output of the above code is the following time series object:

Time Series:

Start = 1

End = 42

Frequency = 1

[1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56

```
births <-
scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
```

```
births <- ts(births, frequency = 12, start = c(1946, 1))
```

```
births
```

Q.5 What is your analysis on above code and explain the plot obtained. (3 points)

We load a dataset of number of births per month in New York city, from January 1946 to December 1959.

The dataset looks as follows:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1946	26.663	23.598	26.931	24.740	25.806	24.364	24.477	23.901	23.175	23.227	21.672	21.870
1947	21.439	21.089	23.709	21.669	21.752	20.761	23.479	23.824	23.105	23.110	21.759	22.073
1948	21.937	20.035	23.590	21.672	22.222	22.123	23.950	23.504	22.238	23.142	21.059	21.573

1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907 21.519 22.025

1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252 22.084 22.991

1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110 22.964 23.981

1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199 23.162 24.707

1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462 25.246 25.180

1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379 24.712 25.688

1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784 25.693 26.881

1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136 26.291 26.987

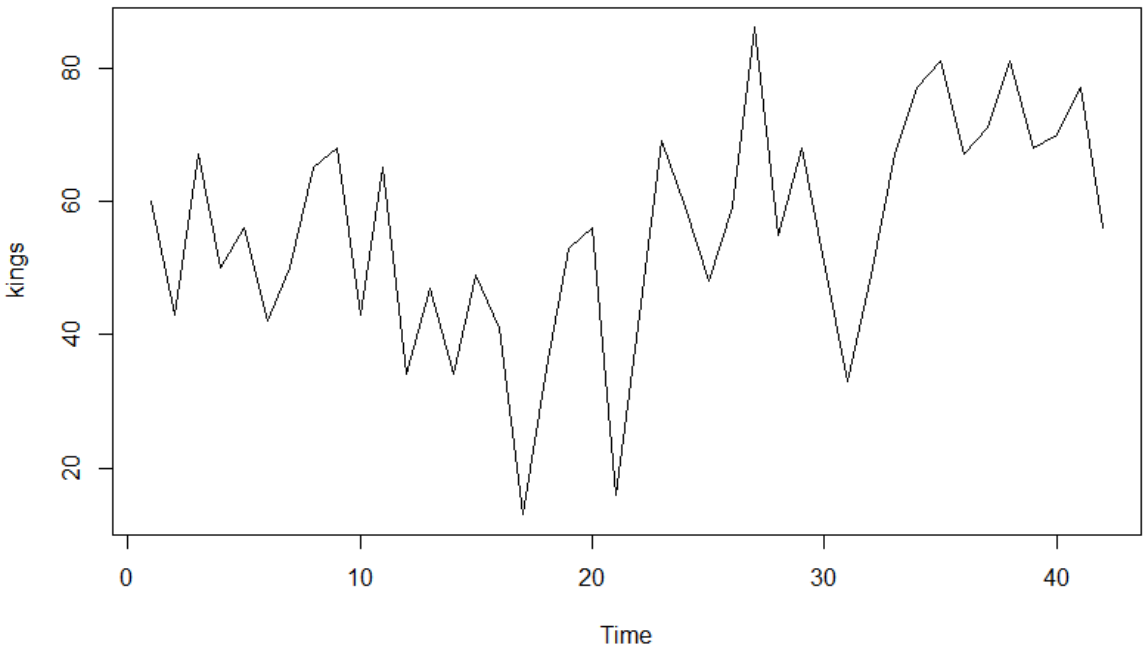
1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484 26.634 27.735

1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945 25.912 26.619

1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012 26.992 27.897

The graph is:

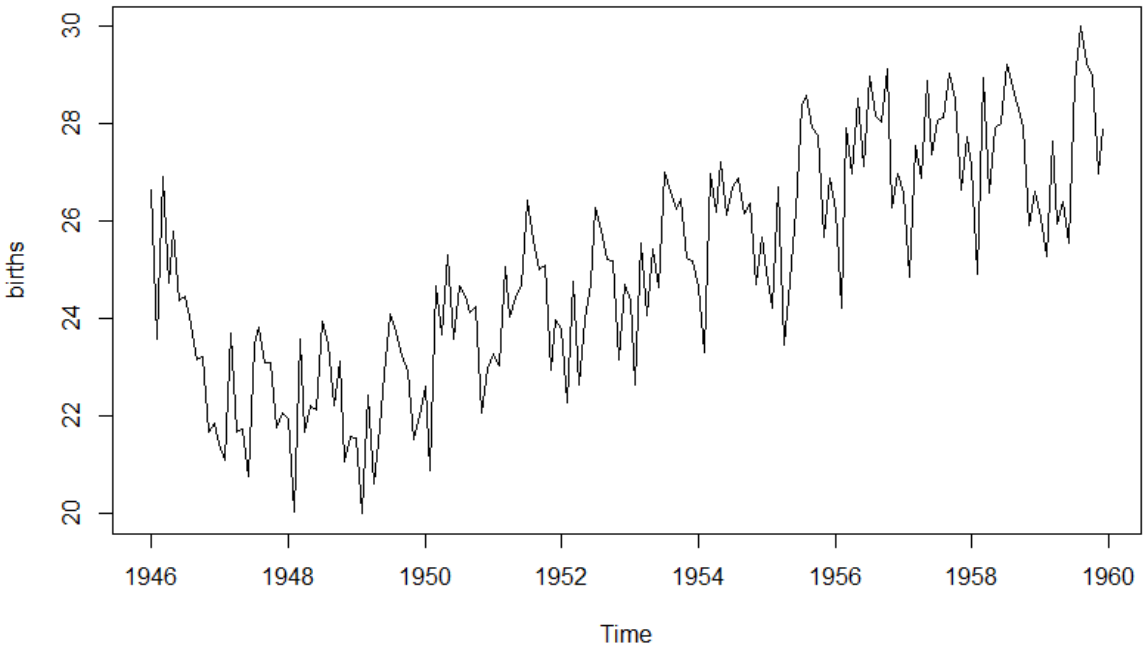
```
plot.ts(kings)
```




```
plot.ts(births)
```

Q.6 What kind of variation do you find here. (3 points)

It is a combination of seasonal and cyclic variation.



Q.7 (4 points)

(a) Explain the below lines of code and plot obtained.

(b)What do you understand by decomposition of data?

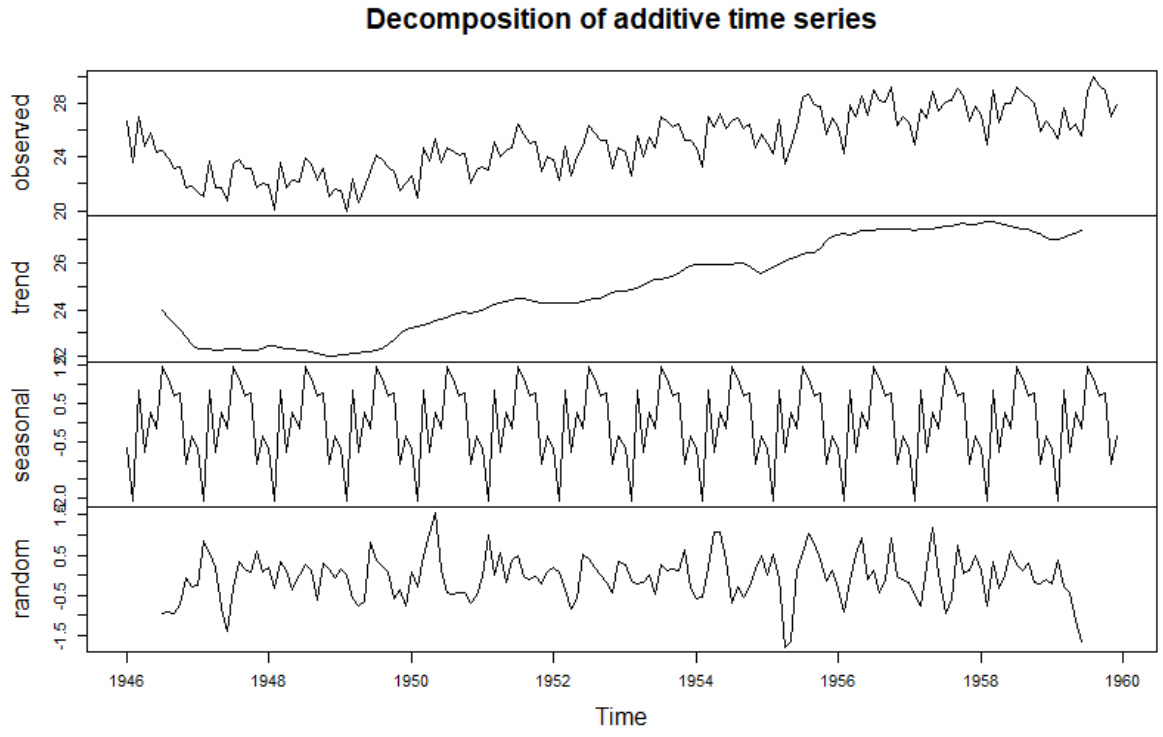
(c)What kind of data is this (i.e., seasonal, trend, or cyclic)?

```
birthsComp <- decompose(births)
plot(birthsComp)
```

These lines decompose the data and then plot it.

Decomposition of time series data means splitting it into seasonal, trend and cyclic data.

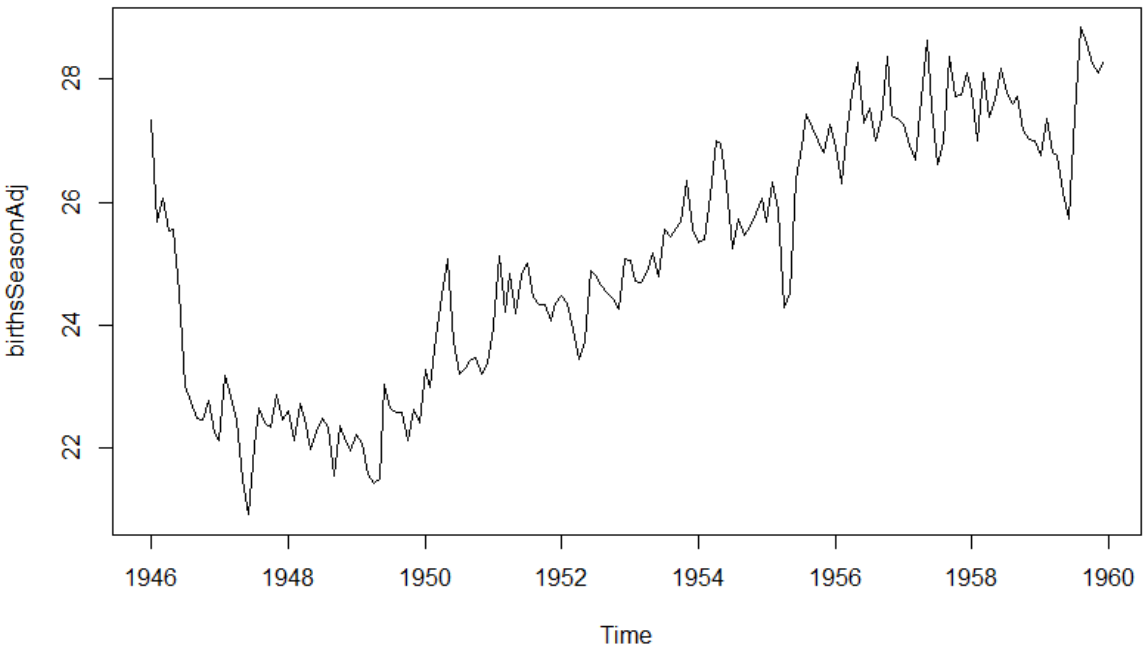
This data is seasonal in nature.



Q.8 Explain the below lines of code and plot obtained. (3 points)

```
birthsSeasonAdj <- births - birthsComp$seasonal
plot(birthsSeasonAdj)
```

These line show the plot of the difference between births and the seasonal effect.



Section 3:

1. Downloaded 'Twitter Stock Market Analysis Dataset' from Kaggle (Twitter | Stock Market Analysis | Founding Years | Kaggle).

2. Dataset description:

Stock Market Analysis of Twitter Inc from its Founding / Listing Years which is 2013 to 2022.

Data Dictionary:

Columns	Description
Date	Date of Listing (YYYY-MM-DD)
Open	Price when the market opens
High	Highest recorded price for the day
Low	Lowest recorded price for the day
Close	Price when the market closes
Adj Close	Modified closing price based on corporate actions

Columns	Description
Volume	Number of stocks sold in a day

Reason for choosing the dataset:

Daily price and the changes within the day can be seen here. The highest and lowest prices for every single day help in identifying patterns at the minuscule level. Volume per day is also recorded.

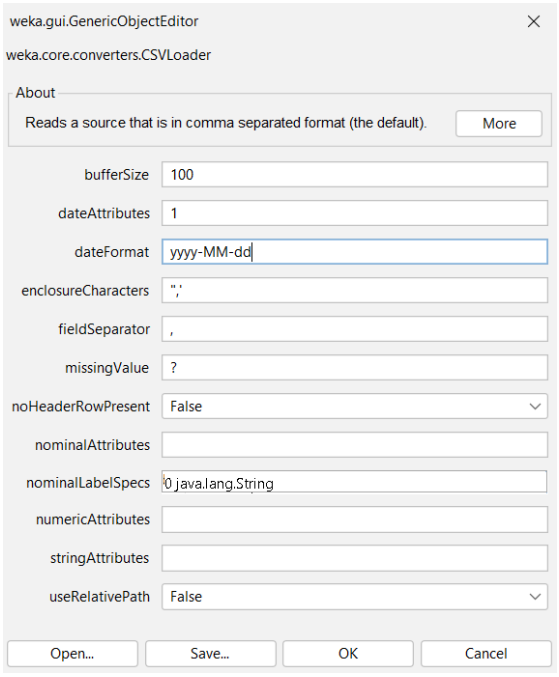
3. For this project purpose the 'Close' variable is chosen as the target variable and the 'Date' column is chosen for the corresponding dates. So, the closing prices can be predicted by the trained model.

4. Loaded the TWTR.csv dataset in WEKA.

While importing the dataset, the following steps have been followed:

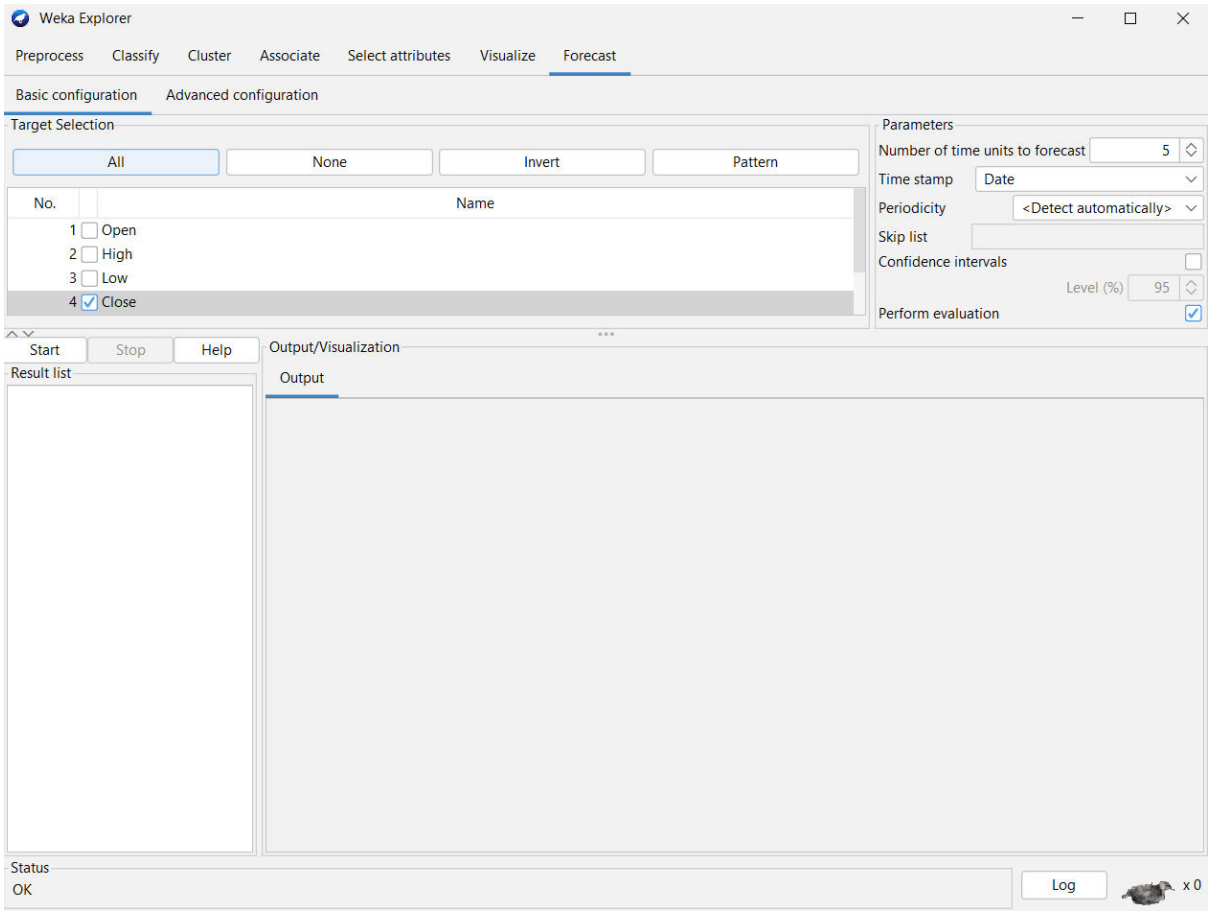
In the 'Open' dialog box 'Invoke options dialog' check box has been selected.

After clicking on 'open', the following dialog box has opened and set the options as shown in the screen shot below (These steps are taken for converting the 'Date' column into date format):

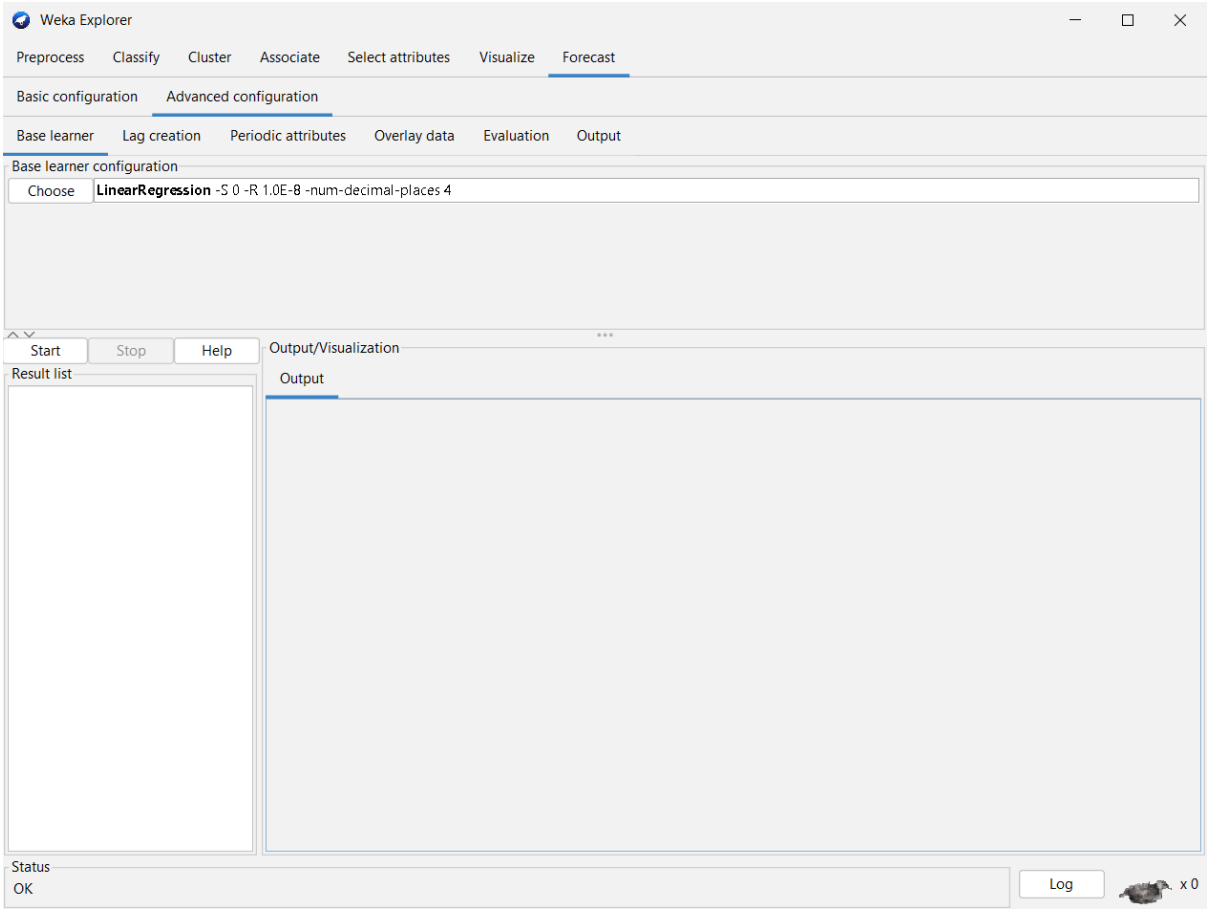


5. Applied Time Series Forecasting. This feature was activated by installing timeSeriesForecasting package from Tools>Package Manager option.

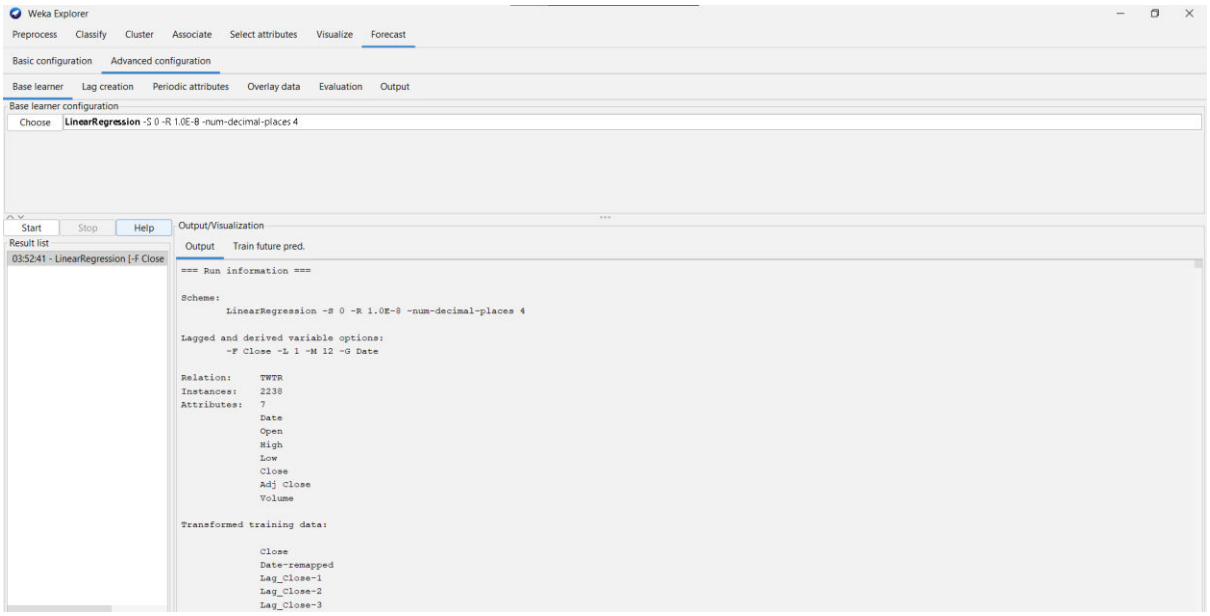
6. Set the options as following:

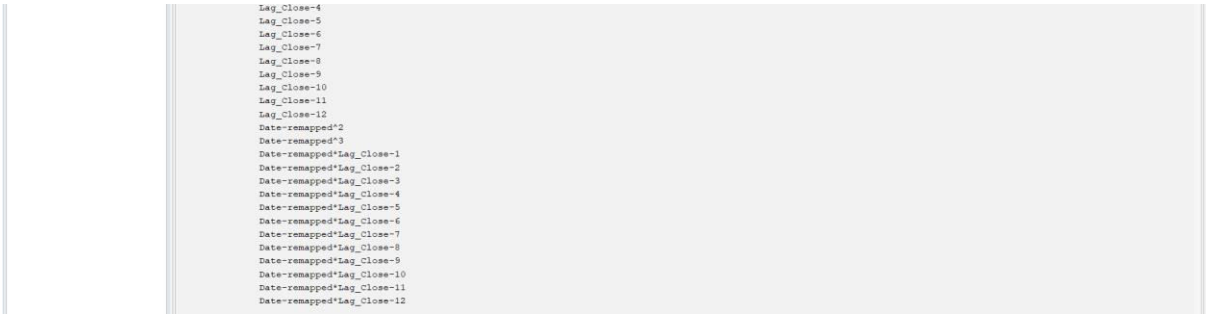


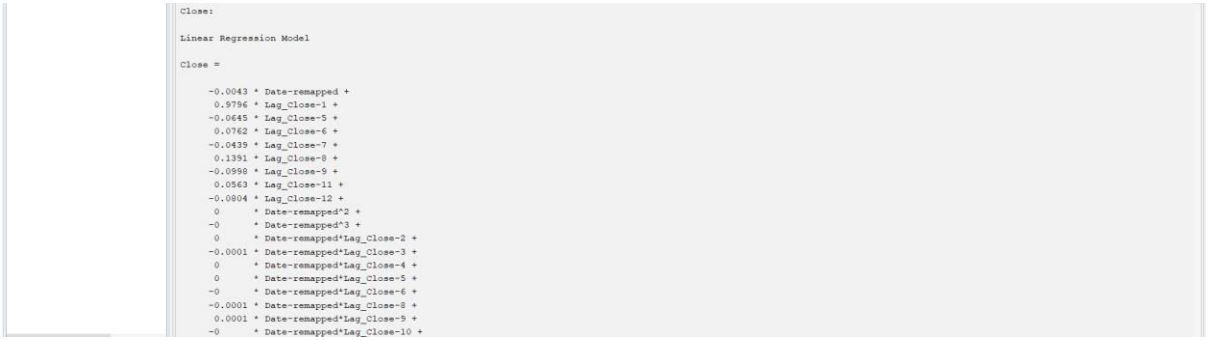
7. In the ‘Advanced configuration’ section, changed the ‘Based learner configuration’ to ‘LinearRegression’.

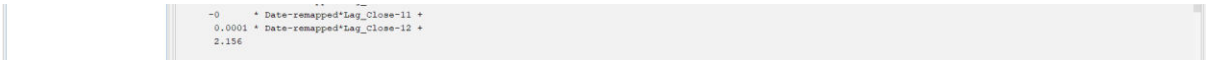


8. Output from the model:









=== Run information ===

Scheme:

LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Lagged and derived variable options:

-F Close -L 1 -M 12 -G Date

Relation: TWTR

Instances: 2238

Attributes: 7

Date

Open

High

Low

Close

Adj Close

Volume

Transformed training data:

Close

Date-remapped

Lag_Close-1

Lag_Close-2

Lag_Close-3

Lag_Close-4

Lag_Close-5

Lag_Close-6

Lag_Close-7

Lag_Close-8

Lag_Close-9

Lag_Close-10

Lag_Close-11

Lag_Close-12

Date-remapped^2

Date-remapped^3

Date-remapped*Lag_Close-1

Date-remapped*Lag_Close-2

Date-remapped*Lag_Close-3

Date-remapped*Lag_Close-4

Date-remapped*Lag_Close-5

Date-remapped*Lag_Close-6

Date-remapped*Lag_Close-7

Date-remapped*Lag_Close-8

Date-remapped*Lag_Close-9

Date-remapped*Lag_Close-10

Date-remapped*Lag_Close-11

Date-remapped*Lag_Close-12

Close:

Linear Regression Model

Close =

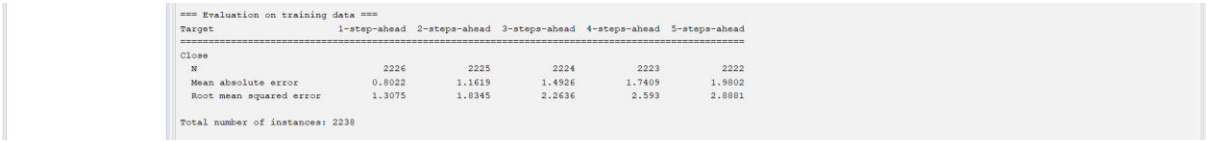
-0.0043 * Date-remapped +
0.9796 * Lag_Close-1 +
-0.0645 * Lag_Close-5 +
0.0762 * Lag_Close-6 +
-0.0439 * Lag_Close-7 +
0.1391 * Lag_Close-8 +
-0.0998 * Lag_Close-9 +
0.0563 * Lag_Close-11 +
-0.0804 * Lag_Close-12 +
0 * Date-remapped^2 +
-0 * Date-remapped^3 +
0 * Date-remapped*Lag_Close-2 +
-0.0001 * Date-remapped*Lag_Close-3 +
0 * Date-remapped*Lag_Close-4 +
0 * Date-remapped*Lag_Close-5 +
-0 * Date-remapped*Lag_Close-6 +
-0.0001 * Date-remapped*Lag_Close-8 +
0.0001 * Date-remapped*Lag_Close-9 +
-0 * Date-remapped*Lag_Close-10 +
-0 * Date-remapped*Lag_Close-11 +
0.0001 * Date-remapped*Lag_Close-12 +
2.156

=== Evaluation on training data ===

Target	1-step-ahead	2-steps-ahead	3-steps-ahead	4-steps-ahead	5-steps-ahead
Close					
N	2226	2225	2224	2223	2222

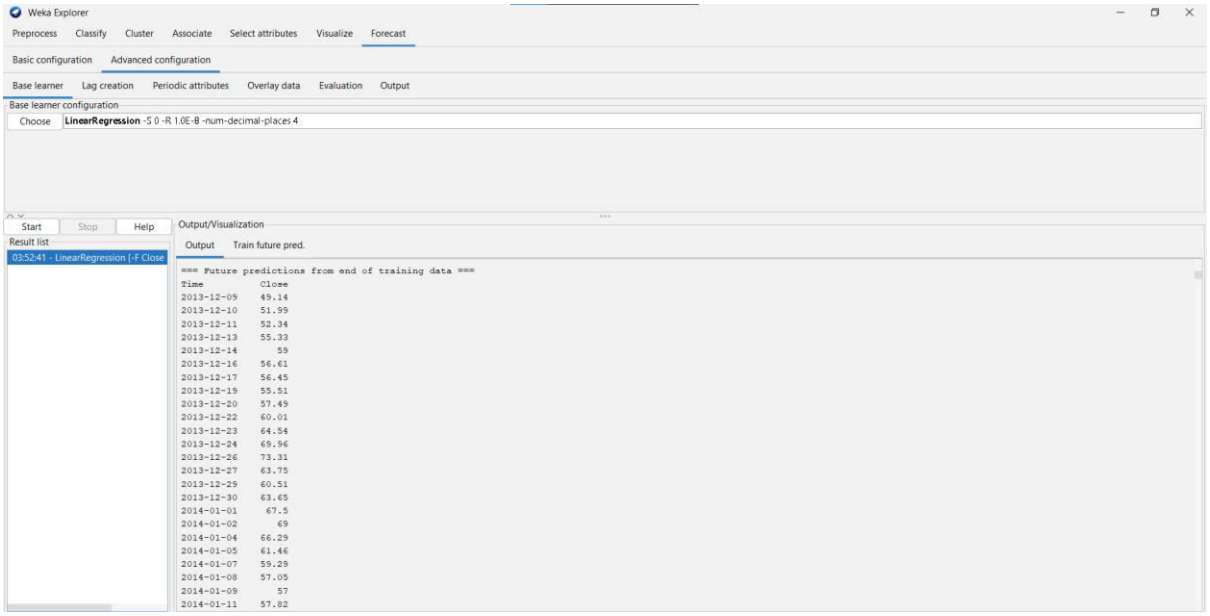
Mean absolute error	0.8022	1.1619	1.4926	1.7409	1.9802
Root mean squared error	1.3075	1.8345	2.2636	2.593	2.8881

Total number of instances: 2238

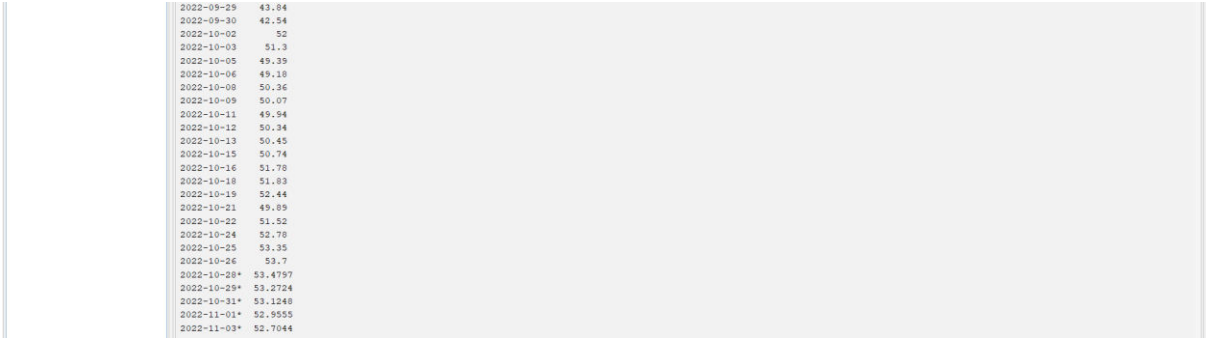


The model created some lags as independent features, and the performance of the model is quite good. RMSE and MAE values are very low.

9. Predictions from the model:



...



=== Future predictions from end of training data ===

Time	Close
2013-12-09	49.14
2013-12-10	51.99
2013-12-11	52.34
2013-12-13	55.33

2013-12-14	59
2013-12-16	56.61
2013-12-17	56.45
2013-12-19	55.51
2013-12-20	57.49
2013-12-22	60.01
2013-12-23	64.54
2013-12-24	69.96
2013-12-26	73.31
2013-12-27	63.75
2013-12-29	60.51
2013-12-30	63.65
2014-01-01	67.5
2014-01-02	69
2014-01-04	66.29
2014-01-05	61.46
2014-01-07	59.29
2014-01-08	57.05
2014-01-09	57
2014-01-11	57.82
2014-01-12	58.21
2014-01-14	61.57
2014-01-15	60.57
2014-01-17	62.2
2014-01-18	62.53
2014-01-20	62.44
2014-01-21	62.8
2014-01-22	61.74
2014-01-24	57.91
2014-01-25	60.44
2014-01-27	59.45
2014-01-28	63.47
2014-01-30	64.5
2014-01-31	65.25

2014-02-02	66.32
2014-02-03	65.97
2014-02-05	50.03
2014-02-06	54.35
2014-02-07	52.92
2014-02-09	54
2014-02-10	56.85
2014-02-12	56.47
2014-02-13	57.44
2014-02-15	58.18
2014-02-16	55.5
2014-02-18	56.63
2014-02-19	55.92
2014-02-20	55.78
2014-02-22	54.96
2014-02-23	55.87
2014-02-25	55.77
2014-02-26	54.91
2014-02-28	53.71
2014-03-01	54.28
2014-03-03	54.38
2014-03-04	54.83
2014-03-06	53.53
2014-03-07	53.88
2014-03-08	54.02
2014-03-10	54.5
2014-03-11	53.57
2014-03-13	51.92
2014-03-14	52.05
2014-03-16	51.13
2014-03-17	51.24
2014-03-19	50.12
2014-03-20	50.92
2014-03-21	48.77

2014-03-23	47.88
2014-03-24	44.43
2014-03-26	46.32
2014-03-27	47.3
2014-03-29	46.67
2014-03-30	46.98
2014-04-01	45.73
2014-04-02	44.05
2014-04-04	43.14
2014-04-05	42.45
2014-04-06	41.78
2014-04-08	42.49
2014-04-09	41.34
2014-04-11	40.05
2014-04-12	40.87
2014-04-14	45.52
2014-04-15	44.42
2014-04-17	45.01
2014-04-18	46.13
2014-04-19	46.02
2014-04-21	45.95
2014-04-22	44.82
2014-04-24	41.61
2014-04-25	40.73
2014-04-27	42.62
2014-04-28	38.97
2014-04-30	39.09
2014-05-01	39.02
2014-05-03	38.75
2014-05-04	31.85
2014-05-05	30.66
2014-05-07	31.96
2014-05-08	32.05
2014-05-10	33.94

2014-05-11	33.39
2014-05-13	32.85
2014-05-14	32.77
2014-05-16	32.26
2014-05-17	32.07
2014-05-18	31.77
2014-05-20	31.75
2014-05-21	31.52
2014-05-23	30.5
2014-05-24	30.51
2014-05-26	33.77
2014-05-27	34
2014-05-29	32.44
2014-05-30	31.75
2014-06-01	32.58
2014-06-02	32.9
2014-06-03	33.89
2014-06-05	33.33
2014-06-06	34.47
2014-06-08	35.37
2014-06-09	35.54
2014-06-11	36.79
2014-06-12	36.9
2014-06-14	38.02
2014-06-15	38.02
2014-06-16	38.74
2014-06-18	38.9
2014-06-19	39.24
2014-06-21	39.52
2014-06-22	38.48
2014-06-24	39.46
2014-06-25	41.44
2014-06-27	40.93
2014-06-28	40.97

2014-06-30	42.05
2014-07-01	41.77
2014-07-02	41.33
2014-07-04	40.23
2014-07-05	37.41
2014-07-07	38.06
2014-07-08	37.84
2014-07-10	38.33
2014-07-11	38.31
2014-07-13	37.88
2014-07-14	37.43
2014-07-15	36.87
2014-07-17	37.05
2014-07-18	38.05
2014-07-20	37.65
2014-07-21	37.75
2014-07-23	38.71
2014-07-24	38.16
2014-07-26	37.93
2014-07-27	38.59
2014-07-29	46.3
2014-07-30	45.19
2014-07-31	44.13
2014-08-02	43.47
2014-08-03	43.83
2014-08-05	43.46
2014-08-06	43
2014-08-08	43.13
2014-08-09	43.27
2014-08-11	43.81
2014-08-12	44.15
2014-08-13	45.33
2014-08-15	44.76
2014-08-16	45.12

2014-08-18	45.09
2014-08-19	45.06
2014-08-21	45.11
2014-08-22	45.98
2014-08-24	46.1
2014-08-25	48.17
2014-08-27	48.06
2014-08-28	49.43
2014-08-29	49.75
2014-08-31	51.02
2014-09-01	49.33
2014-09-03	50.24
2014-09-04	50.7
2014-09-06	52
2014-09-07	50.61
2014-09-09	52.91
2014-09-10	52.64
2014-09-11	52.11
2014-09-13	49.38
2014-09-14	50.83
2014-09-16	50.7
2014-09-17	50.88
2014-09-19	53
2014-09-20	51.94
2014-09-22	52.17
2014-09-23	52.96
2014-09-25	51.45
2014-09-26	51.89
2014-09-27	51.74
2014-09-29	51.58
2014-09-30	50.06
2014-10-02	51.85
2014-10-03	53.94
2014-10-05	53.49

2014-10-06	53.53
2014-10-08	55.42
2014-10-09	55.29
2014-10-10	50.4
2014-10-12	48.49
2014-10-13	48.58
2014-10-15	49.99
2014-10-16	48.23
2014-10-18	48.77
2014-10-19	50.7
2014-10-21	50.63
2014-10-22	49.08
2014-10-24	49.67
2014-10-25	49.95
2014-10-26	48.56
2014-10-28	43.78
2014-10-29	42.08
2014-10-31	41.8
2014-11-01	41.47
2014-11-03	40.21
2014-11-04	40.9
2014-11-06	40.37
2014-11-07	40.84
2014-11-08	40.31
2014-11-10	39.59
2014-11-11	39.59
2014-11-13	42.54
2014-11-14	40.04
2014-11-16	41.85
2014-11-17	40.47
2014-11-19	40.61
2014-11-20	39.71
2014-11-22	39.81
2014-11-23	40.03

2014-11-24	40.19
2014-11-26	39.76
2014-11-27	41.13
2014-11-29	41.74
2014-11-30	39.04
2014-12-02	38.91
2014-12-03	39.06
2014-12-05	38.79
2014-12-06	38.49
2014-12-07	36.29
2014-12-09	37.05
2014-12-10	36.35
2014-12-12	36.7
2014-12-13	37.1
2014-12-15	36.85
2014-12-16	35.13
2014-12-18	35.57
2014-12-19	36.73
2014-12-21	37.08
2014-12-22	38.43
2014-12-23	37.57
2014-12-25	37.61
2014-12-26	37.6
2014-12-28	36.42
2014-12-29	35.86
2014-12-31	35.87
2015-01-01	36.56
2015-01-03	36.38
2015-01-04	38.76
2015-01-05	37.28
2015-01-07	39.09
2015-01-08	40.17
2015-01-10	39.37
2015-01-11	39.65

2015-01-13 39.85

2015-01-14 36.93

2015-01-16 37.31

2015-01-17 37.57

2015-01-19 37.83

2015-01-20 39.07

2015-01-21 39.42

2015-01-23 40.1

2015-01-24 38.92

2015-01-26 37.15

2015-01-27 36.68

2015-01-29 37.53

2015-01-30 37.46

2015-02-01 39.79

2015-02-02 40.72

2015-02-03 41.26

2015-02-05 48.01

2015-02-06 47.32

2015-02-08 46.26

2015-02-09 47.5

2015-02-11 47.95

2015-02-12 48.5

2015-02-14 48.03

2015-02-15 47.82

2015-02-17 48.7

2015-02-18 49.11

2015-02-19 48.47

2015-02-21 48.69

2015-02-22 48.55

2015-02-24 49.41

2015-02-25 48.08

2015-02-27 48.15

2015-02-28 47.71

2015-03-02 47.57

2015-03-03	47.35
2015-03-04	46.75
2015-03-06	47.59
2015-03-07	45.84
2015-03-09	46.27
2015-03-10	47.07
2015-03-12	46.66
2015-03-13	46.43
2015-03-15	46.93
2015-03-16	47.2
2015-03-18	47.93
2015-03-19	48.44
2015-03-20	48.46
2015-03-22	51.47
2015-03-23	49.5
2015-03-25	49.92
2015-03-26	50.01
2015-03-28	49.89
2015-03-29	50.08
2015-03-31	50.47
2015-04-01	50.42
2015-04-03	50.84
2015-04-04	52.87
2015-04-05	52.3
2015-04-07	52.17
2015-04-08	51.94
2015-04-10	51.62
2015-04-11	51.2
2015-04-13	51.3
2015-04-14	52.03
2015-04-16	50.66
2015-04-17	51.4
2015-04-18	51.32
2015-04-20	51.73

2015-04-21	51.41
2015-04-23	50.82
2015-04-24	51.66
2015-04-26	42.27
2015-04-27	38.49
2015-04-29	38.96
2015-04-30	37.84
2015-05-02	37.88
2015-05-03	37.42
2015-05-04	37.26
2015-05-06	37.71
2015-05-07	37.59
2015-05-09	37.31
2015-05-10	37.48
2015-05-12	37.72
2015-05-13	37.33
2015-05-15	37.1
2015-05-16	37.28
2015-05-17	37.5
2015-05-19	36.78
2015-05-20	36.68
2015-05-22	36.6
2015-05-23	36.51
2015-05-25	36.41
2015-05-26	36.83
2015-05-28	36.67
2015-05-29	36.63
2015-05-31	36.4
2015-06-01	37
2015-06-02	36.71
2015-06-04	37
2015-06-05	36.46
2015-06-07	35.88
2015-06-08	35.85

2015-06-10	35.84
2015-06-11	35.9
2015-06-13	34.67
2015-06-14	34.82
2015-06-15	34.69
2015-06-17	34.66
2015-06-18	35.86
2015-06-20	35.55
2015-06-21	35.37
2015-06-23	35.17
2015-06-24	35.17
2015-06-26	35.26
2015-06-27	34.21
2015-06-29	36.22
2015-06-30	35.4
2015-07-01	35.72
2015-07-03	35.43
2015-07-04	35.52
2015-07-06	34.76
2015-07-07	34.36
2015-07-09	34.91
2015-07-10	35.78
2015-07-12	36.72
2015-07-13	35.66
2015-07-14	36.1
2015-07-16	35.67
2015-07-17	35.81
2015-07-19	36.63
2015-07-20	36.09
2015-07-22	36.19
2015-07-23	35.42
2015-07-25	34.7
2015-07-26	36.54
2015-07-28	31.24

2015-07-29	31.47
2015-07-30	31.01
2015-08-01	29.27
2015-08-02	29.34
2015-08-04	28.48
2015-08-05	27.54
2015-08-07	27.04
2015-08-08	29.5
2015-08-10	29.62
2015-08-11	29.39
2015-08-12	28.54
2015-08-14	29.06
2015-08-15	29.06
2015-08-17	28.3
2015-08-18	27.61
2015-08-20	26
2015-08-21	25.87
2015-08-23	25.17
2015-08-24	24.38
2015-08-26	25.03
2015-08-27	26.46
2015-08-28	26.83
2015-08-30	27.79
2015-08-31	27.03
2015-09-02	27.82
2015-09-03	28.3
2015-09-05	28.15
2015-09-06	27.18
2015-09-08	27.18
2015-09-09	27.71
2015-09-10	27.39
2015-09-12	26.9
2015-09-13	27.17
2015-09-15	27.75

2015-09-16	27.41
2015-09-18	27.96
2015-09-19	27.38
2015-09-21	26.83
2015-09-22	26.79
2015-09-24	26.6
2015-09-25	25.29
2015-09-26	25.26
2015-09-28	25.59
2015-09-29	26.94
2015-10-01	24.68
2015-10-02	26.31
2015-10-04	28.15
2015-10-05	27.62
2015-10-07	29.83
2015-10-08	30.32
2015-10-09	30.85
2015-10-11	28.75
2015-10-12	29.06
2015-10-14	29.38
2015-10-15	29.71
2015-10-17	31.15
2015-10-18	30.91
2015-10-20	30.91
2015-10-21	29.3
2015-10-23	29.15
2015-10-24	30.28
2015-10-25	30.89
2015-10-27	31.34
2015-10-28	30.87
2015-10-30	29.06
2015-10-31	28.46
2015-11-02	29.2
2015-11-03	29.13

2015-11-05 29.36

2015-11-06 28.66

2015-11-07 28.28

2015-11-09 27.09

2015-11-10 27.05

2015-11-12 26.5

2015-11-13 26.13

2015-11-15 25.18

2015-11-16 25.41

2015-11-18 25.23

2015-11-19 25.9

2015-11-21 26.32

2015-11-22 26.27

2015-11-23 25.2

2015-11-25 25.52

2015-11-26 26.06

2015-11-28 25.75

2015-11-29 25.4

2015-12-01 25.53

2015-12-02 25.4

2015-12-04 25.9

2015-12-05 25.02

2015-12-06 24.46

2015-12-08 24.99

2015-12-09 24.31

2015-12-11 25.91

2015-12-12 24.84

2015-12-14 24.92

2015-12-15 23.95

2015-12-17 24.3

2015-12-18 23.31

2015-12-20 22.99

2015-12-21 22.14

2015-12-22 22.56

2015-12-24	22.67
2015-12-25	22.97
2015-12-27	22.53
2015-12-28	22.47
2015-12-30	22.23
2015-12-31	23.14
2016-01-02	22.56
2016-01-03	21.92
2016-01-04	21.39
2016-01-06	20.26
2016-01-07	19.98
2016-01-09	19.65
2016-01-10	19.62
2016-01-12	18.68
2016-01-13	19
2016-01-15	17.94
2016-01-16	16.69
2016-01-18	17.38
2016-01-19	17.83
2016-01-20	17.84
2016-01-22	17.02
2016-01-23	17.01
2016-01-25	16.78
2016-01-26	16.49
2016-01-28	16.8
2016-01-29	17.91
2016-01-31	16.08
2016-02-01	16.56
2016-02-02	16.91
2016-02-04	15.72
2016-02-05	14.9
2016-02-07	14.4
2016-02-08	14.98
2016-02-10	14.31

2016-02-11	15.88
2016-02-13	16.36
2016-02-14	17.46
2016-02-16	18.43
2016-02-17	18.31
2016-02-18	18.3
2016-02-20	18.3
2016-02-21	18
2016-02-23	17.59
2016-02-24	17.94
2016-02-26	18.12
2016-02-27	17.85
2016-02-29	18.54
2016-03-01	19.31
2016-03-02	19.36
2016-03-04	19.17
2016-03-05	18.33
2016-03-07	17.66
2016-03-08	16.61
2016-03-10	16.81
2016-03-11	17.12
2016-03-13	16.19
2016-03-14	16.7
2016-03-16	16.85
2016-03-17	16.85
2016-03-18	16.89
2016-03-20	16.86
2016-03-21	16.01
2016-03-23	15.91
2016-03-24	15.6
2016-03-26	15.97
2016-03-27	16.36
2016-03-29	16.55
2016-03-30	15.98

2016-03-31	17.09
2016-04-02	17.05
2016-04-03	17.26
2016-04-05	16.98
2016-04-06	16.65
2016-04-08	16.51
2016-04-09	16.57
2016-04-11	17.37
2016-04-12	17.53
2016-04-14	17.58
2016-04-15	17.31
2016-04-16	16.92
2016-04-18	17.4
2016-04-19	17.51
2016-04-21	17.23
2016-04-22	17.09
2016-04-24	17.75
2016-04-25	14.86
2016-04-27	14.64
2016-04-28	14.62
2016-04-29	14.4
2016-05-01	14.01
2016-05-02	14.84
2016-05-04	14.12
2016-05-05	14.4
2016-05-07	14.2
2016-05-08	14.63
2016-05-10	14.59
2016-05-11	14.08
2016-05-13	14.1
2016-05-14	14.29
2016-05-15	14.34
2016-05-17	14.14
2016-05-18	14.15

2016-05-20	14.43
2016-05-21	14.41
2016-05-23	14.03
2016-05-24	14.41
2016-05-26	14.3
2016-05-27	15.1
2016-05-28	15.22
2016-05-30	15.02
2016-05-31	15.2
2016-06-02	15.2
2016-06-03	15.27
2016-06-05	15
2016-06-06	14.95
2016-06-08	14.6
2016-06-09	14.02
2016-06-11	14.55
2016-06-12	15.36
2016-06-13	15.96
2016-06-15	15.87
2016-06-16	16.1
2016-06-18	16.34
2016-06-19	16.32
2016-06-21	16.13
2016-06-22	17.04
2016-06-24	16.44
2016-06-25	15.84
2016-06-27	16.42
2016-06-28	16.83
2016-06-29	16.91
2016-07-01	17.28
2016-07-02	17.14
2016-07-04	17.2
2016-07-05	17.37
2016-07-07	18.08

2016-07-08	17.71
2016-07-10	18.1
2016-07-11	17.74
2016-07-12	17.96
2016-07-14	18.08
2016-07-15	18.65
2016-07-17	18.33
2016-07-18	18.56
2016-07-20	18.39
2016-07-21	18.37
2016-07-23	18.65
2016-07-24	18.45
2016-07-26	15.77
2016-07-27	16.31
2016-07-28	16.64
2016-07-30	16.64
2016-07-31	16.42
2016-08-02	17.61
2016-08-03	18.13
2016-08-05	18.26
2016-08-06	18.2
2016-08-08	18.68
2016-08-09	19.04
2016-08-10	19.78
2016-08-12	19.54
2016-08-13	20.86
2016-08-15	20.4
2016-08-16	20.17
2016-08-18	19
2016-08-19	18.98
2016-08-21	18.55
2016-08-22	18.69
2016-08-24	18.25
2016-08-25	18.32

2016-08-26	18.3
2016-08-28	18.47
2016-08-29	18.38
2016-08-31	19.21
2016-09-01	19.5
2016-09-03	19.55
2016-09-04	19.93
2016-09-06	19.87
2016-09-07	18.7
2016-09-08	18.11
2016-09-10	18.15
2016-09-11	17.76
2016-09-13	18.08
2016-09-14	18.3
2016-09-16	19.11
2016-09-17	18.36
2016-09-19	18.39
2016-09-20	18.49
2016-09-22	18.63
2016-09-23	22.62
2016-09-24	23.37
2016-09-26	23.72
2016-09-27	22.96
2016-09-29	23.01
2016-09-30	23.05
2016-10-02	24
2016-10-03	23.52
2016-10-05	24.87
2016-10-06	19.87
2016-10-07	19.85
2016-10-09	17.56
2016-10-10	18
2016-10-12	18.05
2016-10-13	17.79

2016-10-15	16.88
2016-10-16	16.73
2016-10-18	16.83
2016-10-19	17.07
2016-10-21	16.9
2016-10-22	18.09
2016-10-23	18.03
2016-10-25	17.26
2016-10-26	17.29
2016-10-28	17.4
2016-10-29	17.66
2016-10-31	17.95
2016-11-01	17.49
2016-11-03	17.61
2016-11-04	17.58
2016-11-05	18.02
2016-11-07	18.41
2016-11-08	18.38
2016-11-10	19.13
2016-11-11	18.37
2016-11-13	18.55
2016-11-14	19.14
2016-11-16	18.98
2016-11-17	18.63
2016-11-19	18.55
2016-11-20	18.73
2016-11-21	18.6
2016-11-23	18.63
2016-11-24	18.22
2016-11-26	18.06
2016-11-27	18.3
2016-11-29	18.19
2016-11-30	18.49
2016-12-02	18.03

2016-12-03	17.93
2016-12-04	18.23
2016-12-06	18.23
2016-12-07	19.48
2016-12-09	19.64
2016-12-10	19.65
2016-12-12	18.93
2016-12-13	19.37
2016-12-15	18.93
2016-12-16	18.79
2016-12-18	18.63
2016-12-19	18.24
2016-12-20	17.92
2016-12-22	17.08
2016-12-23	16.41
2016-12-25	16.5
2016-12-26	16.61
2016-12-28	16.39
2016-12-29	16.39
2016-12-31	16.3
2017-01-01	16.44
2017-01-02	16.86
2017-01-04	17.09
2017-01-05	17.17
2017-01-07	17.5
2017-01-08	17.37
2017-01-10	17.3
2017-01-11	17.38
2017-01-13	17.25
2017-01-14	16.96
2017-01-16	17.11
2017-01-17	16.79
2017-01-18	16.58
2017-01-20	16.61

2017-01-21	16.52
2017-01-23	16.73
2017-01-24	16.81
2017-01-26	16.57
2017-01-27	16.94
2017-01-29	17.62
2017-01-30	17.24
2017-01-31	17.78
2017-02-02	17.61
2017-02-03	17.93
2017-02-05	18.26
2017-02-06	18.72
2017-02-08	16.41
2017-02-09	15.58
2017-02-11	15.81
2017-02-12	16.52
2017-02-14	16.74
2017-02-15	16.35
2017-02-16	16.62
2017-02-18	16.42
2017-02-19	16.08
2017-02-21	16.03
2017-02-22	15.98
2017-02-24	16.06
2017-02-25	15.77
2017-02-27	15.79
2017-02-28	15.79
2017-03-01	15.75
2017-03-03	15.56
2017-03-04	15.18
2017-03-06	15.24
2017-03-07	15.22
2017-03-09	15.12
2017-03-10	15.21

2017-03-12	15.32
2017-03-13	15.03
2017-03-15	15.19
2017-03-16	15.08
2017-03-17	15.09
2017-03-19	14.54
2017-03-20	14.98
2017-03-22	14.93
2017-03-23	15.14
2017-03-25	14.99
2017-03-26	14.94
2017-03-28	15.04
2017-03-29	14.92
2017-03-30	14.95
2017-04-01	14.84
2017-04-02	14.69
2017-04-04	14.53
2017-04-05	14.39
2017-04-07	14.29
2017-04-08	14.36
2017-04-10	14.31
2017-04-11	14.42
2017-04-13	14.3
2017-04-14	14.4
2017-04-15	14.44
2017-04-17	14.54
2017-04-18	14.65
2017-04-20	14.63
2017-04-21	14.71
2017-04-23	14.66
2017-04-24	15.82
2017-04-26	16.61
2017-04-27	16.48
2017-04-28	17.54

2017-04-30	18.24
2017-05-01	18.57
2017-05-03	18.48
2017-05-04	18.69
2017-05-06	18.31
2017-05-07	18.37
2017-05-09	18.54
2017-05-10	18.39
2017-05-12	18.61
2017-05-13	19.23
2017-05-14	19.49
2017-05-16	18.28
2017-05-17	18.51
2017-05-19	18.35
2017-05-20	18.43
2017-05-22	18.15
2017-05-23	17.98
2017-05-25	17.95
2017-05-26	18.23
2017-05-27	18.43
2017-05-29	18.32
2017-05-30	18.53
2017-06-01	18.31
2017-06-02	18.23
2017-06-04	17.57
2017-06-05	17.44
2017-06-07	17.59
2017-06-08	16.9
2017-06-10	17.04
2017-06-11	16.97
2017-06-12	16.76
2017-06-14	16.83
2017-06-15	16.67
2017-06-17	17.06

2017-06-18	16.91
2017-06-20	17.78
2017-06-21	18.15
2017-06-23	18.5
2017-06-24	18.29
2017-06-25	18.12
2017-06-27	17.95
2017-06-28	17.65
2017-06-30	17.87
2017-07-01	17.65
2017-07-03	17.82
2017-07-04	17.92
2017-07-06	18.02
2017-07-07	18.08
2017-07-09	18.64
2017-07-10	19.25
2017-07-11	19.32
2017-07-13	19.64
2017-07-14	19.94
2017-07-16	19.98
2017-07-17	20.12
2017-07-19	20.53
2017-07-20	20.11
2017-07-22	20
2017-07-23	19.97
2017-07-24	19.61
2017-07-26	16.84
2017-07-27	16.75
2017-07-29	16.09
2017-07-30	16.21
2017-08-01	16.07
2017-08-02	16.18
2017-08-04	16.29
2017-08-05	16.4

2017-08-07	16.15
2017-08-08	16.14
2017-08-09	15.75
2017-08-11	15.92
2017-08-12	16.09
2017-08-14	15.95
2017-08-15	16.16
2017-08-17	15.87
2017-08-18	15.99
2017-08-20	16.11
2017-08-21	16.63
2017-08-22	16.96
2017-08-24	16.89
2017-08-25	16.65
2017-08-27	16.77
2017-08-28	16.93
2017-08-30	16.93
2017-08-31	16.91
2017-09-02	16.86
2017-09-03	16.65
2017-09-05	16.83
2017-09-06	17.22
2017-09-07	17.45
2017-09-09	17.66
2017-09-10	18.17
2017-09-12	18.2
2017-09-13	18.21
2017-09-15	18.01
2017-09-16	17.6
2017-09-18	17.76
2017-09-19	17.62
2017-09-20	17.58
2017-09-22	17.61
2017-09-23	16.98

2017-09-25	16.59
2017-09-26	16.95
2017-09-28	16.85
2017-09-29	16.87
2017-10-01	17.09
2017-10-02	17.59
2017-10-04	17.75
2017-10-05	18.25
2017-10-06	17.85
2017-10-08	17.67
2017-10-09	17.41
2017-10-11	17.73
2017-10-12	18.45
2017-10-14	18.63
2017-10-15	18.33
2017-10-17	18.28
2017-10-18	18.02
2017-10-20	17.89
2017-10-21	17.87
2017-10-22	17.37
2017-10-24	17.25
2017-10-25	17.14
2017-10-27	20.31
2017-10-28	21.68
2017-10-30	21.25
2017-10-31	20.62
2017-11-02	20.61
2017-11-03	19.71
2017-11-04	19.9
2017-11-06	19.39
2017-11-07	19.66
2017-11-09	19.59
2017-11-10	19.9
2017-11-12	20.32

2017-11-13	20.17
2017-11-15	20.05
2017-11-16	19.91
2017-11-18	20.36
2017-11-19	20.76
2017-11-20	21.13
2017-11-22	21.88
2017-11-23	22.27
2017-11-25	22.42
2017-11-26	21.82
2017-11-28	21.83
2017-11-29	20.79
2017-12-01	20.58
2017-12-02	20.71
2017-12-03	20.4
2017-12-05	20.77
2017-12-06	21.09
2017-12-08	21.01
2017-12-09	21.1
2017-12-11	22.05
2017-12-12	21.65
2017-12-14	21.66
2017-12-15	22.58
2017-12-17	22.23
2017-12-18	24.68
2017-12-19	25.08
2017-12-21	25.2
2017-12-22	25.05
2017-12-24	24.46
2017-12-25	24.26
2017-12-27	24.23
2017-12-28	24.31
2017-12-30	24.01
2017-12-31	24.51

2018-01-01 24.45

2018-01-03 23.99

2018-01-04 24.32

2018-01-06 24.59

2018-01-07 24.17

2018-01-09 24.25

2018-01-10 24.35

2018-01-12 25.41

2018-01-13 24.66

2018-01-15 24.56

2018-01-16 24.04

2018-01-17 23.66

2018-01-19 23.32

2018-01-20 22.75

2018-01-22 22.37

2018-01-23 22.16

2018-01-25 24.27

2018-01-26 25.18

2018-01-28 25.62

2018-01-29 25.81

2018-01-30 27.14

2018-02-01 25.92

2018-02-02 25.13

2018-02-04 25.24

2018-02-05 26.91

2018-02-07 30.18

2018-02-08 31.51

2018-02-10 30.95

2018-02-11 33.44

2018-02-13 33.75

2018-02-14 33.61

2018-02-15 33.06

2018-02-17 32.84

2018-02-18 33.38

2018-02-20	32.11
2018-02-21	32.66
2018-02-23	32.16
2018-02-24	31.32
2018-02-26	31.86
2018-02-27	32.24
2018-02-28	33
2018-03-02	34.58
2018-03-03	34.43
2018-03-05	35.76
2018-03-06	34.85
2018-03-08	35.35
2018-03-09	35.5
2018-03-11	34.11
2018-03-12	36.6
2018-03-14	35.8
2018-03-15	35.58
2018-03-16	34.98
2018-03-18	31.35
2018-03-19	32.73
2018-03-21	31.2
2018-03-22	31.03
2018-03-24	31.91
2018-03-25	28.07
2018-03-27	28.45
2018-03-28	29.01
2018-03-29	28.04
2018-03-31	27.54
2018-04-01	28.25
2018-04-03	28.64
2018-04-04	28.1
2018-04-06	28.01
2018-04-07	29.53
2018-04-09	29.39

2018-04-10	29
2018-04-12	28.76
2018-04-13	28.58
2018-04-14	31.84
2018-04-16	31.54
2018-04-17	31.54
2018-04-19	31.91
2018-04-20	31.22
2018-04-22	30.47
2018-04-23	29.75
2018-04-25	30.27
2018-04-26	29
2018-04-27	30.31
2018-04-29	30.3
2018-04-30	30.55
2018-05-02	30.67
2018-05-03	31.04
2018-05-05	31.33
2018-05-06	31.85
2018-05-08	32.46
2018-05-09	32.87
2018-05-11	32.75
2018-05-12	33.39
2018-05-13	32.75
2018-05-15	32.77
2018-05-16	32.58
2018-05-18	32.63
2018-05-19	33.63
2018-05-21	32.86
2018-05-22	33.42
2018-05-24	33.52
2018-05-25	33.63
2018-05-26	34.04
2018-05-28	34.36

2018-05-29	34.7
2018-05-31	36.65
2018-06-01	37.88
2018-06-03	39.8
2018-06-04	40.1
2018-06-06	39.7
2018-06-07	41.21
2018-06-09	41.42
2018-06-10	43.49
2018-06-11	44.07
2018-06-13	46.76
2018-06-14	45.8
2018-06-16	46
2018-06-17	44.95
2018-06-19	46.13
2018-06-20	45.24
2018-06-22	45.88
2018-06-23	44.17
2018-06-24	44.84
2018-06-26	43.7
2018-06-27	44.79
2018-06-29	43.67
2018-06-30	44.98
2018-07-02	43.89
2018-07-03	45.06
2018-07-05	46.65
2018-07-06	44.14
2018-07-08	43.75
2018-07-09	43.87
2018-07-10	45.26
2018-07-12	44.49
2018-07-13	44.26
2018-07-15	44.71
2018-07-16	43.34

2018-07-18	43.44
2018-07-19	43.42
2018-07-21	43.31
2018-07-22	42.17
2018-07-23	44.22
2018-07-25	42.94
2018-07-26	34.12
2018-07-28	31.38
2018-07-29	31.87
2018-07-31	31.91
2018-08-01	32.82
2018-08-03	31.96
2018-08-04	32.98
2018-08-06	32.67
2018-08-07	31.84
2018-08-08	31.96
2018-08-10	32.01
2018-08-11	32.8
2018-08-13	33.19
2018-08-14	32.38
2018-08-16	32.83
2018-08-17	32.73
2018-08-19	32.6
2018-08-20	33.69
2018-08-21	33.81
2018-08-23	33.88
2018-08-24	34.28
2018-08-26	35.89
2018-08-27	35.49
2018-08-29	35.35
2018-08-30	35.64
2018-09-01	35.18
2018-09-02	34.84
2018-09-04	32.73

2018-09-05	30.81
2018-09-06	30.49
2018-09-08	30.54
2018-09-09	30.89
2018-09-11	29.75
2018-09-12	30.39
2018-09-14	30.12
2018-09-15	28.86
2018-09-17	29.22
2018-09-18	29.52
2018-09-19	29.85
2018-09-21	28.5
2018-09-22	28.6
2018-09-24	29.11
2018-09-25	29.01
2018-09-27	29.42
2018-09-28	28.46
2018-09-30	28.31
2018-10-01	28.19
2018-10-03	29.01
2018-10-04	28.23
2018-10-05	28.39
2018-10-07	28.45
2018-10-08	29.27
2018-10-10	26.79
2018-10-11	27
2018-10-13	27.99
2018-10-14	28.61
2018-10-16	29.87
2018-10-17	29.55
2018-10-18	29.29
2018-10-20	28.83
2018-10-21	29.18
2018-10-23	28.77

2018-10-24 27.54

2018-10-26 31.8

2018-10-27 32.36

2018-10-29 32.39

2018-10-30 33.86

2018-11-01 34.75

2018-11-02 34.62

2018-11-03 34.3

2018-11-05 34.02

2018-11-06 34.42

2018-11-08 34.99

2018-11-09 34.18

2018-11-11 34.08

2018-11-12 32.01

2018-11-14 32.49

2018-11-15 32.91

2018-11-16 33.15

2018-11-18 33.67

2018-11-19 31.98

2018-11-21 31.06

2018-11-22 31.61

2018-11-24 31.12

2018-11-25 32.82

2018-11-27 32.61

2018-11-28 32.73

2018-11-30 31.3

2018-12-01 31.45

2018-12-02 33.66

2018-12-04 32.56

2018-12-05 32.96

2018-12-07 32.83

2018-12-08 33.43

2018-12-10 34.45

2018-12-11 36.25

2018-12-13	35.89
2018-12-14	35.87
2018-12-15	33.43
2018-12-17	33.74
2018-12-18	32.93
2018-12-20	29.29
2018-12-21	27.31
2018-12-23	26.45
2018-12-24	28.66
2018-12-26	28.68
2018-12-27	28.43
2018-12-29	28.74
2018-12-30	28.81
2018-12-31	27.99
2019-01-02	29.95
2019-01-03	31.34
2019-01-05	31.8
2019-01-06	32.25
2019-01-08	33.09
2019-01-09	32.87
2019-01-11	32.37
2019-01-12	33.02
2019-01-14	32.47
2019-01-15	32.85
2019-01-16	33.27
2019-01-18	32.25
2019-01-19	30.97
2019-01-21	31.61
2019-01-22	32.9
2019-01-24	33.13
2019-01-25	31.64
2019-01-27	32.26
2019-01-28	33.56
2019-01-29	33.19

2019-01-31	33.94
2019-02-01	34.37
2019-02-03	34.16
2019-02-04	30.8
2019-02-06	30.01
2019-02-07	30.23
2019-02-09	30.39
2019-02-10	31.12
2019-02-12	30.96
2019-02-13	31.23
2019-02-14	31.65
2019-02-16	31.37
2019-02-17	30.76
2019-02-19	31.71
2019-02-20	31.99
2019-02-22	31.01
2019-02-23	30.41
2019-02-25	30.78
2019-02-26	30.62
2019-02-27	30.5
2019-03-01	31.03
2019-03-02	30.8
2019-03-04	30.12
2019-03-05	30.04
2019-03-07	30.87
2019-03-08	31.16
2019-03-10	31.3
2019-03-11	31.03
2019-03-13	31.22
2019-03-14	31.08
2019-03-15	31.27
2019-03-17	32.57
2019-03-18	32.61
2019-03-20	33.02

2019-03-21	32.59
2019-03-23	33.06
2019-03-24	32.28
2019-03-26	32.87
2019-03-27	32.88
2019-03-28	33.44
2019-03-30	33.75
2019-03-31	34.38
2019-04-02	34.42
2019-04-03	34.72
2019-04-05	34.86
2019-04-06	35.14
2019-04-08	34.75
2019-04-09	34.58
2019-04-11	34.37
2019-04-12	34.71
2019-04-13	34.46
2019-04-15	34.48
2019-04-16	34.4
2019-04-18	34.39
2019-04-19	39.77
2019-04-21	39.29
2019-04-22	38.48
2019-04-24	38.67
2019-04-25	39.78
2019-04-26	39.91
2019-04-28	39.29
2019-04-29	39.95
2019-05-01	40.8
2019-05-02	40.23
2019-05-04	38.62
2019-05-05	38.58
2019-05-07	38.79
2019-05-08	38.45

2019-05-10	36.59
2019-05-11	36.93
2019-05-12	37.9
2019-05-14	38.3
2019-05-15	37.5
2019-05-17	37.15
2019-05-18	37.47
2019-05-20	38.58
2019-05-21	37.19
2019-05-23	37.41
2019-05-24	37.29
2019-05-25	36.85
2019-05-27	37.14
2019-05-28	36.44
2019-05-30	34.43
2019-05-31	36.1
2019-06-02	36.33
2019-06-03	36.59
2019-06-05	37.93
2019-06-06	37.64
2019-06-08	37.21
2019-06-09	37.49
2019-06-10	36.34
2019-06-12	36.15
2019-06-13	36.44
2019-06-15	36.65
2019-06-16	36.29
2019-06-18	35.44
2019-06-19	35.02
2019-06-21	35.58
2019-06-22	34.72
2019-06-23	35.22
2019-06-25	34.75
2019-06-26	34.9

2019-06-28	36.08
2019-06-29	36.22
2019-07-01	36.02
2019-07-02	36.25
2019-07-04	36.45
2019-07-05	37.65
2019-07-07	37.47
2019-07-08	37.21
2019-07-09	37.84
2019-07-11	38.68
2019-07-12	37.99
2019-07-14	37.7
2019-07-15	37.66
2019-07-17	36.77
2019-07-18	37.58
2019-07-20	37.9
2019-07-21	38.73
2019-07-22	38.12
2019-07-24	41.52
2019-07-25	41.5
2019-07-27	41
2019-07-28	42.31
2019-07-30	42.08
2019-07-31	42.85
2019-08-02	40.37
2019-08-03	41.32
2019-08-05	41.73
2019-08-06	42.07
2019-08-07	41.53
2019-08-09	40.48
2019-08-10	41.81
2019-08-12	40.65
2019-08-13	40.09
2019-08-15	40.58

2019-08-16	41.7
2019-08-18	42.29
2019-08-19	42.8
2019-08-20	42.18
2019-08-22	41
2019-08-23	41.44
2019-08-25	42.14
2019-08-26	41.68
2019-08-28	42.49
2019-08-29	42.65
2019-08-31	41.96
2019-09-01	43.36
2019-09-03	45.3
2019-09-04	45.42
2019-09-05	44.26
2019-09-07	43.25
2019-09-08	43.25
2019-09-10	43.2
2019-09-11	42.63
2019-09-13	42.76
2019-09-14	43.24
2019-09-16	43.24
2019-09-17	42.93
2019-09-18	43.23
2019-09-20	43.38
2019-09-21	41.42
2019-09-23	42.5
2019-09-24	42.44
2019-09-26	41.35
2019-09-27	41.2
2019-09-29	40.25
2019-09-30	39.7
2019-10-02	40
2019-10-03	40.36

2019-10-04	40.8
2019-10-06	39.7
2019-10-07	39.5
2019-10-09	39.54
2019-10-10	40.36
2019-10-12	39.78
2019-10-13	40.28
2019-10-15	39.91
2019-10-16	39.61
2019-10-17	38.99
2019-10-19	40.09
2019-10-20	38.81
2019-10-22	38.83
2019-10-23	30.75
2019-10-25	30.3
2019-10-26	30.06
2019-10-28	29.85
2019-10-29	29.86
2019-10-31	29.97
2019-11-01	29.62
2019-11-02	30.05
2019-11-04	29.81
2019-11-05	29.54
2019-11-07	29.05
2019-11-08	29.21
2019-11-10	29.34
2019-11-11	29.05
2019-11-13	29.08
2019-11-14	28.89
2019-11-15	29.25
2019-11-17	29.49
2019-11-18	29.44
2019-11-20	29.17
2019-11-21	29.79

2019-11-23	30.03
2019-11-24	30.54
2019-11-26	30.96
2019-11-27	31.17
2019-11-29	30.91
2019-11-30	30.42
2019-12-01	29.97
2019-12-03	30
2019-12-04	30.04
2019-12-06	30.19
2019-12-07	30.21
2019-12-09	29.84
2019-12-10	30.55
2019-12-12	30.3
2019-12-13	30.39
2019-12-14	30.97
2019-12-16	30.7
2019-12-17	31.68
2019-12-19	32.03
2019-12-20	32.13
2019-12-22	32.43
2019-12-23	32.5
2019-12-25	32.63
2019-12-26	32.55
2019-12-28	32.03
2019-12-29	32.05
2019-12-30	32.3
2020-01-01	31.52
2020-01-02	31.64
2020-01-04	32.54
2020-01-05	33.05
2020-01-07	33.22
2020-01-08	32.78
2020-01-10	32.69

2020-01-11	32.82
2020-01-12	33.23
2020-01-14	34.19
2020-01-15	34.22
2020-01-17	34.09
2020-01-18	34.02
2020-01-20	33.89
2020-01-21	33.19
2020-01-23	32.78
2020-01-24	33.42
2020-01-26	33.63
2020-01-27	33.22
2020-01-28	32.48
2020-01-30	33.07
2020-01-31	33.96
2020-02-02	33.39
2020-02-03	38.41
2020-02-05	37.03
2020-02-06	35.96
2020-02-08	35.65
2020-02-09	36.79
2020-02-10	37.16
2020-02-12	36.91
2020-02-13	38.06
2020-02-15	38.77
2020-02-16	39.05
2020-02-18	38.31
2020-02-19	35.89
2020-02-21	35.21
2020-02-22	34.63
2020-02-24	33.01
2020-02-25	33.2
2020-02-26	35.82
2020-02-28	34.93

2020-02-29	36
2020-03-02	34.97
2020-03-03	33.46
2020-03-05	32.46
2020-03-06	34.31
2020-03-08	31.3
2020-03-09	26.78
2020-03-10	29.29
2020-03-12	24.04
2020-03-13	24.48
2020-03-15	22
2020-03-16	24.13
2020-03-18	23.95
2020-03-19	24.69
2020-03-21	25.85
2020-03-22	25.97
2020-03-24	26.41
2020-03-25	25.29
2020-03-26	25.59
2020-03-28	24.56
2020-03-29	23.32
2020-03-31	23.02
2020-04-01	23.09
2020-04-03	24.93
2020-04-04	25.61
2020-04-06	27.86
2020-04-07	27.77
2020-04-08	27.21
2020-04-10	27.94
2020-04-11	27.51
2020-04-13	26.55
2020-04-14	26.7
2020-04-16	27.01
2020-04-17	25.75

2020-04-19	28.44
2020-04-20	27.86
2020-04-22	28.74
2020-04-23	30
2020-04-24	28.79
2020-04-26	31.09
2020-04-27	28.68
2020-04-29	27.84
2020-04-30	28.23
2020-05-02	28.07
2020-05-03	27.68
2020-05-05	28.77
2020-05-06	29.93
2020-05-08	29.69
2020-05-09	29.15
2020-05-10	28.13
2020-05-12	28.56
2020-05-13	29
2020-05-15	29.64
2020-05-16	29.88
2020-05-18	32.23
2020-05-19	32.34
2020-05-21	32.62
2020-05-22	34.01
2020-05-23	33.07
2020-05-25	31.6
2020-05-26	30.97
2020-05-28	31.89
2020-05-29	32.26
2020-05-31	34.88
2020-06-01	33.72
2020-06-03	34.87
2020-06-04	36.64
2020-06-06	35.92

2020-06-07	35.14
2020-06-08	33.03
2020-06-10	33.4
2020-06-11	34.11
2020-06-13	34.63
2020-06-14	34.34
2020-06-16	34.03
2020-06-17	33.41
2020-06-19	33.47
2020-06-20	32.91
2020-06-21	31.73
2020-06-23	31.37
2020-06-24	29.05
2020-06-26	29.49
2020-06-27	29.79
2020-06-29	30.81
2020-06-30	30.87
2020-07-02	32.34
2020-07-03	32.99
2020-07-05	35.41
2020-07-06	35.72
2020-07-07	35.4
2020-07-09	33.82
2020-07-10	34.38
2020-07-12	35.67
2020-07-13	35.28
2020-07-15	35.81
2020-07-16	37.06
2020-07-18	37.01
2020-07-19	36.94
2020-07-20	38.44
2020-07-22	37.54
2020-07-23	36.85
2020-07-25	36.61

2020-07-26	37.16
2020-07-28	36.72
2020-07-29	36.4
2020-07-31	36.39
2020-08-01	36.35
2020-08-03	36.79
2020-08-04	37.69
2020-08-05	37.14
2020-08-07	37.44
2020-08-08	37.28
2020-08-10	37.44
2020-08-11	37.82
2020-08-13	37.9
2020-08-14	37.97
2020-08-16	38.01
2020-08-17	38.89
2020-08-18	38.96
2020-08-20	39.26
2020-08-21	40.49
2020-08-23	40.55
2020-08-24	41.08
2020-08-26	40.39
2020-08-27	41.07
2020-08-29	40.58
2020-08-30	41.15
2020-09-01	43.67
2020-09-02	41.63
2020-09-03	39.87
2020-09-05	38.19
2020-09-06	39.56
2020-09-08	38.95
2020-09-09	38.93
2020-09-11	38.77
2020-09-12	39.09

2020-09-14	39.6
2020-09-15	39.35
2020-09-16	40.15
2020-09-18	39.9
2020-09-19	42.73
2020-09-21	45.33
2020-09-22	43.14
2020-09-24	43.84
2020-09-25	44.15
2020-09-27	44.74
2020-09-28	44.5
2020-09-30	46.7
2020-10-01	46.12
2020-10-02	47.31
2020-10-04	45.6
2020-10-05	45.87
2020-10-07	46.01
2020-10-08	45.9
2020-10-10	48.25
2020-10-11	47
2020-10-13	45.98
2020-10-14	46.03
2020-10-15	45.81
2020-10-17	45.82
2020-10-18	46.35
2020-10-20	50.24
2020-10-21	50.28
2020-10-23	50.44
2020-10-24	49
2020-10-26	51.27
2020-10-27	48.53
2020-10-29	52.43
2020-10-30	41.36
2020-10-31	39.47

2020-11-02	41.73
2020-11-03	42.76
2020-11-05	43.71
2020-11-06	43.12
2020-11-08	43.19
2020-11-09	42.41
2020-11-11	43.63
2020-11-12	42.81
2020-11-13	43.48
2020-11-15	42.73
2020-11-16	42.85
2020-11-18	43.33
2020-11-19	43.62
2020-11-21	44.68
2020-11-22	44.94
2020-11-24	45.23
2020-11-25	46.43
2020-11-27	46.59
2020-11-28	46.51
2020-11-29	46.67
2020-12-01	47.23
2020-12-02	47.79
2020-12-04	47.73
2020-12-05	47.9
2020-12-07	47.43
2020-12-08	47.23
2020-12-10	51.21
2020-12-11	51.44
2020-12-12	52.02
2020-12-14	52.82
2020-12-15	54.03
2020-12-17	54.63
2020-12-18	55.87
2020-12-20	54.64

2020-12-21	54.91
2020-12-23	54.3
2020-12-24	53.97
2020-12-26	54.43
2020-12-27	54.36
2020-12-28	54.33
2020-12-30	54.15
2020-12-31	54.53
2021-01-02	53.88
2021-01-03	53.26
2021-01-05	52.33
2021-01-06	51.48
2021-01-08	48.18
2021-01-09	47.04
2021-01-10	47.22
2021-01-12	45.79
2021-01-13	45.18
2021-01-15	45.93
2021-01-16	47.6
2021-01-18	47.12
2021-01-19	48.06
2021-01-21	47.84
2021-01-22	49.67
2021-01-24	48.19
2021-01-25	51.57
2021-01-26	50.53
2021-01-28	52.66
2021-01-29	54
2021-01-31	54.58
2021-02-01	56.51
2021-02-03	56.78
2021-02-04	58.2
2021-02-06	59.87
2021-02-07	67.77

2021-02-08	68.56
2021-02-10	71.9
2021-02-11	73.96
2021-02-13	71.79
2021-02-14	72.26
2021-02-16	72.28
2021-02-17	70.49
2021-02-19	73.17
2021-02-20	71.92
2021-02-22	74.59
2021-02-23	77.06
2021-02-24	77.63
2021-02-26	73.67
2021-02-27	70.86
2021-03-01	66.75
2021-03-02	66.95
2021-03-04	63.48
2021-03-05	67.52
2021-03-07	64.83
2021-03-08	68.46
2021-03-09	68.1
2021-03-11	70.28
2021-03-12	69.25
2021-03-14	70.2
2021-03-15	66.72
2021-03-17	66.23
2021-03-18	65.21
2021-03-20	64.27
2021-03-21	62.06
2021-03-23	61.2
2021-03-24	61.26
2021-03-25	62.94
2021-03-27	62.99
2021-03-28	63.63

2021-03-30	63.83
2021-03-31	64.24
2021-04-02	67
2021-04-03	68.99
2021-04-05	71.22
2021-04-06	71.19
2021-04-07	70.86
2021-04-09	72.45
2021-04-10	69.74
2021-04-12	71.71
2021-04-13	70.12
2021-04-15	67.94
2021-04-16	65.7
2021-04-18	67.33
2021-04-19	64.31
2021-04-21	67.02
2021-04-22	66.72
2021-04-23	66.01
2021-04-25	65.7
2021-04-26	65.09
2021-04-28	55.22
2021-04-29	54.58
2021-05-01	54.4
2021-05-02	53.56
2021-05-04	53.81
2021-05-05	53.79
2021-05-06	51.81
2021-05-08	52.88
2021-05-09	50.7
2021-05-11	50.11
2021-05-12	51.73
2021-05-14	52.6
2021-05-15	53.19
2021-05-17	52.92

2021-05-18	54.71
2021-05-20	54.45
2021-05-21	57.06
2021-05-22	57
2021-05-24	57.85
2021-05-25	58.08
2021-05-27	58
2021-05-28	57.44
2021-05-30	57.16
2021-05-31	57.01
2021-06-02	59
2021-06-03	59.63
2021-06-04	58.81
2021-06-06	59.71
2021-06-07	60.3
2021-06-09	60.5
2021-06-10	60.83
2021-06-12	60.23
2021-06-13	59.93
2021-06-15	60.71
2021-06-16	60.85
2021-06-18	61.96
2021-06-19	63.78
2021-06-20	66.49
2021-06-22	68.25
2021-06-23	67.93
2021-06-25	68.29
2021-06-26	69
2021-06-28	68.81
2021-06-29	68.11
2021-07-01	69.2
2021-07-02	70.21
2021-07-03	68.76
2021-07-05	66.83

2021-07-06	68.97
2021-07-08	69.86
2021-07-09	69.6
2021-07-11	70.27
2021-07-12	68.07
2021-07-14	66.41
2021-07-15	66.02
2021-07-17	67.94
2021-07-18	69.54
2021-07-19	69.57
2021-07-21	71.69
2021-07-22	68.69
2021-07-24	68.33
2021-07-25	69.96
2021-07-27	70.37
2021-07-28	69.75
2021-07-30	69.13
2021-07-31	68.63
2021-08-02	68.7
2021-08-03	69.28
2021-08-04	67.37
2021-08-06	67.01
2021-08-07	65.68
2021-08-09	65.46
2021-08-10	64.95
2021-08-12	64.82
2021-08-13	63.78
2021-08-15	62.88
2021-08-16	62.12
2021-08-17	62.05
2021-08-19	62.52
2021-08-20	64.13
2021-08-22	63.98
2021-08-23	64.17

2021-08-25	62.61
2021-08-26	63.43
2021-08-28	64.7
2021-08-29	64.5
2021-08-31	65.55
2021-09-01	64.6
2021-09-02	64.66
2021-09-04	64.98
2021-09-05	62.27
2021-09-07	62.46
2021-09-08	61.56
2021-09-10	59.66
2021-09-11	60.19
2021-09-13	61.84
2021-09-14	62.29
2021-09-15	62.47
2021-09-17	60.95
2021-09-18	62.78
2021-09-20	64.25
2021-09-21	66.69
2021-09-23	67.22
2021-09-24	65.37
2021-09-26	62.45
2021-09-27	60.07
2021-09-29	60.39
2021-09-30	61.98
2021-10-01	58.39
2021-10-03	59.86
2021-10-04	61.29
2021-10-06	63.97
2021-10-07	63.68
2021-10-09	62.1
2021-10-10	61.45
2021-10-12	62.2

2021-10-13	63.17
2021-10-14	62.77
2021-10-16	64.84
2021-10-17	66.11
2021-10-19	65.8
2021-10-20	65.4
2021-10-22	62.24
2021-10-23	62.11
2021-10-25	61.43
2021-10-26	54.81
2021-10-28	54.29
2021-10-29	53.54
2021-10-30	55.11
2021-11-01	53.99
2021-11-02	54.53
2021-11-04	53.68
2021-11-05	53.15
2021-11-07	54.08
2021-11-08	53.7
2021-11-10	52.33
2021-11-11	51.98
2021-11-12	52.25
2021-11-14	52.95
2021-11-15	52.11
2021-11-17	50.61
2021-11-18	48.6
2021-11-20	48.4
2021-11-21	47.31
2021-11-23	47.14
2021-11-24	47.52
2021-11-26	47.07
2021-11-27	45.78
2021-11-28	43.94
2021-11-30	42.82

2021-12-01	42.65
2021-12-03	42.07
2021-12-04	44.47
2021-12-06	44.46
2021-12-07	45.72
2021-12-09	46.46
2021-12-10	45.56
2021-12-11	44.6
2021-12-13	44.35
2021-12-14	44.07
2021-12-16	43.13
2021-12-17	43.07
2021-12-19	42.98
2021-12-20	44.36
2021-12-22	43.9
2021-12-23	44.16
2021-12-25	44.33
2021-12-26	43.43
2021-12-27	42.76
2021-12-29	44.46
2021-12-30	43.22
2022-01-01	42.66
2022-01-02	40.85
2022-01-04	39.5
2022-01-05	39.59
2022-01-07	39.67
2022-01-08	39.97
2022-01-09	40.66
2022-01-11	40.25
2022-01-12	38.7
2022-01-14	38.44
2022-01-15	37.3
2022-01-17	37.35
2022-01-18	37.28

2022-01-20	34.82
2022-01-21	35.06
2022-01-23	34.17
2022-01-24	33.62
2022-01-25	33.84
2022-01-27	35.27
2022-01-28	37.51
2022-01-30	38.12
2022-01-31	36.51
2022-02-02	34.48
2022-02-03	36.94
2022-02-05	36.04
2022-02-06	35.98
2022-02-07	37.83
2022-02-09	37.08
2022-02-10	35.84
2022-02-12	35.69
2022-02-13	36.98
2022-02-15	36.24
2022-02-16	35.43
2022-02-18	34.32
2022-02-19	32.93
2022-02-21	32.76
2022-02-22	34.98
2022-02-23	35.29
2022-02-25	35.55
2022-02-26	35.56
2022-02-28	34.62
2022-03-01	33.49
2022-03-03	33.39
2022-03-04	32.42
2022-03-06	32.68
2022-03-07	34.37
2022-03-08	34.12

2022-03-10	33
2022-03-11	33.03
2022-03-13	33.97
2022-03-14	35.37
2022-03-16	37.3
2022-03-17	37.77
2022-03-19	37.44
2022-03-20	38.43
2022-03-22	38
2022-03-23	38.82
2022-03-24	38.6
2022-03-26	39.12
2022-03-27	40.69
2022-03-29	39
2022-03-30	38.69
2022-04-01	39.31
2022-04-02	49.97
2022-04-04	50.98
2022-04-05	50.77
2022-04-06	48.03
2022-04-08	46.23
2022-04-09	47.01
2022-04-11	44.48
2022-04-12	45.85
2022-04-14	45.08
2022-04-15	48.45
2022-04-17	46.16
2022-04-18	46.72
2022-04-20	47.08
2022-04-21	48.93
2022-04-22	51.7
2022-04-24	49.68
2022-04-25	48.64
2022-04-27	49.11

2022-04-28	49.02
2022-04-30	49.14
2022-05-01	48.87
2022-05-03	49.06
2022-05-04	50.36
2022-05-05	49.8
2022-05-07	47.96
2022-05-08	47.26
2022-05-10	46.09
2022-05-11	45.08
2022-05-13	40.72
2022-05-14	37.39
2022-05-16	38.32
2022-05-17	36.85
2022-05-19	37.29
2022-05-20	38.29
2022-05-21	37.86
2022-05-23	35.76
2022-05-24	37.16
2022-05-26	39.52
2022-05-27	40.17
2022-05-29	39.6
2022-05-30	39.3
2022-06-01	39.91
2022-06-02	40.16
2022-06-03	39.56
2022-06-05	40.13
2022-06-06	40.44
2022-06-08	39.53
2022-06-09	38.98
2022-06-11	37.03
2022-06-12	37.22
2022-06-14	37.99
2022-06-15	37.36

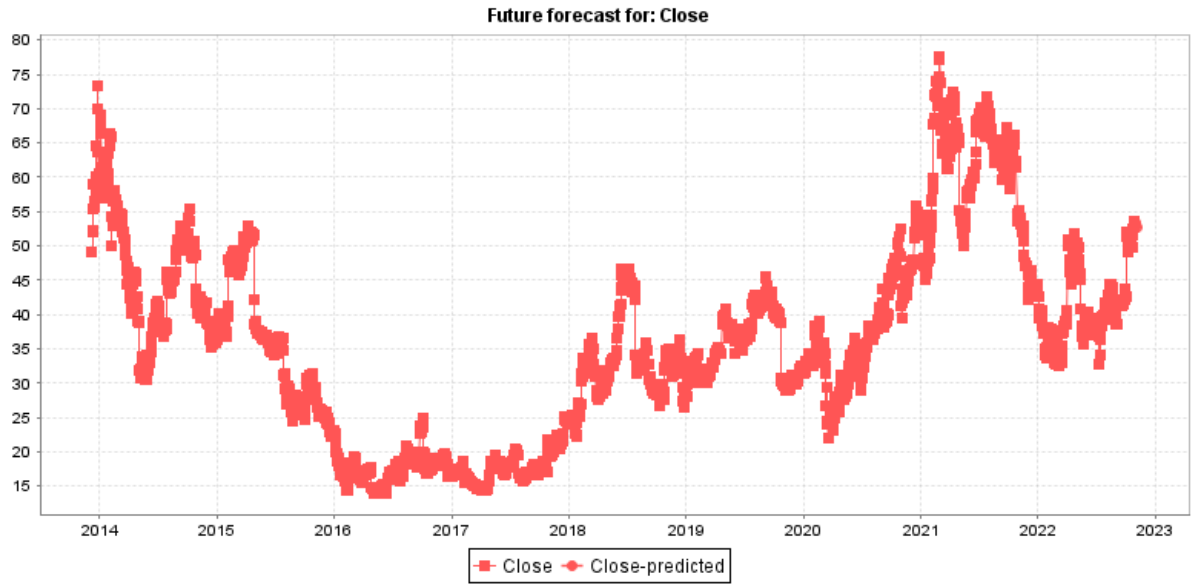
2022-06-17	37.78
2022-06-18	38.91
2022-06-19	38.53
2022-06-21	38.68
2022-06-22	39.41
2022-06-24	39.19
2022-06-25	38.79
2022-06-27	37.8
2022-06-28	37.39
2022-06-30	38.23
2022-07-01	38.38
2022-07-02	38.21
2022-07-04	38.79
2022-07-05	36.81
2022-07-07	32.65
2022-07-08	34.06
2022-07-10	36.75
2022-07-11	36.29
2022-07-13	37.74
2022-07-14	38.41
2022-07-16	39.49
2022-07-17	39.6
2022-07-18	39.52
2022-07-20	39.84
2022-07-21	39.24
2022-07-23	39.34
2022-07-24	39.85
2022-07-26	40.89
2022-07-27	41.61
2022-07-29	40.89
2022-07-30	40.98
2022-07-31	41
2022-08-02	41.06
2022-08-03	42.52

2022-08-05	42.94
2022-08-06	42.83
2022-08-08	44.43
2022-08-09	43.94
2022-08-11	44.26
2022-08-12	44.5
2022-08-14	44.4
2022-08-15	43.99
2022-08-16	43.86
2022-08-18	43.99
2022-08-19	43.01
2022-08-21	39.86
2022-08-22	40.79
2022-08-24	41.05
2022-08-25	40.46
2022-08-27	40.04
2022-08-28	39.32
2022-08-29	38.75
2022-08-31	38.62
2022-09-01	38.63
2022-09-03	38.65
2022-09-04	41.2
2022-09-06	41.85
2022-09-07	42.19
2022-09-09	41.41
2022-09-10	41.74
2022-09-12	41.9
2022-09-13	42.14
2022-09-14	41.45
2022-09-16	41.66
2022-09-17	41.68
2022-09-19	41.27
2022-09-20	41.4
2022-09-22	41.58

2022-09-23	41.52
2022-09-25	42.09
2022-09-26	43.25
2022-09-27	42.74
2022-09-29	43.84
2022-09-30	42.54
2022-10-02	52
2022-10-03	51.3
2022-10-05	49.39
2022-10-06	49.18
2022-10-08	50.36
2022-10-09	50.07
2022-10-11	49.94
2022-10-12	50.34
2022-10-13	50.45
2022-10-15	50.74
2022-10-16	51.78
2022-10-18	51.83
2022-10-19	52.44
2022-10-21	49.89
2022-10-22	51.52
2022-10-24	52.78
2022-10-25	53.35
2022-10-26	53.7
2022-10-28*	53.4797
2022-10-29*	53.2724
2022-10-31*	53.1248
2022-11-01*	52.9555
2022-11-03*	52.7044

Above are the predictions from the model. The last 5 values are future predictions from the model.

10. Train Future Pred plot:


Future forecast for: Close
80
75
70
65
60
55
50
45
40
35
30
25
20
15
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
Close
Close-predicted

CHAPTER 9
TEXT MINING AND ANALYSIS

Dr. Feras Al-Obeidat

Module 4: Advanced Analytics – Theory and Methods

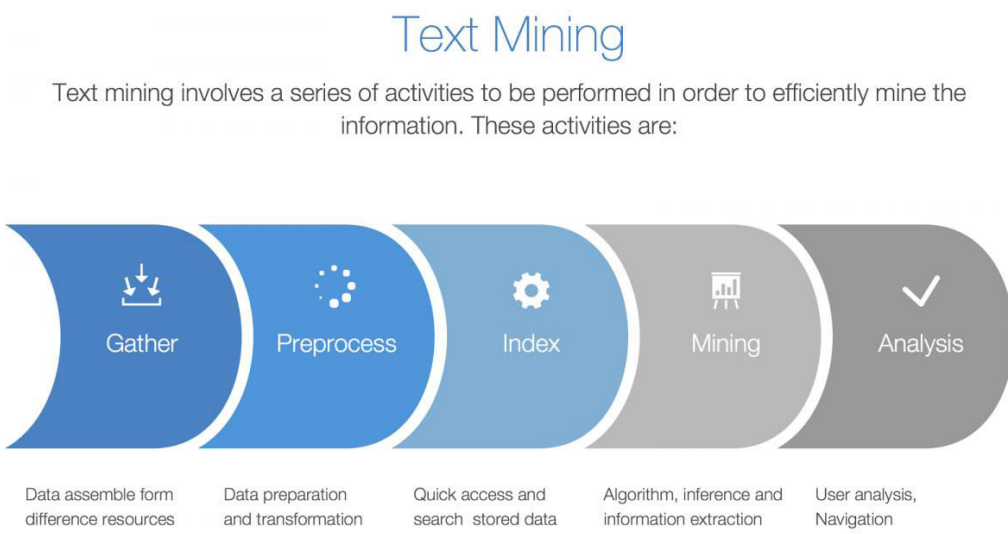
Lesson 8: Text Analysis

During this lesson the following topics are covered:

- Challenges with text analysis
- Key tasks in text analysis
- Definition of terms used in text analysis
  - Term frequency, inverse document frequency
- Representation and features of documents and corpus
- Use of regular expressions in parsing text
- Metrics used to measure the quality of search results
  - Relevance with tf-idf, precision and recall

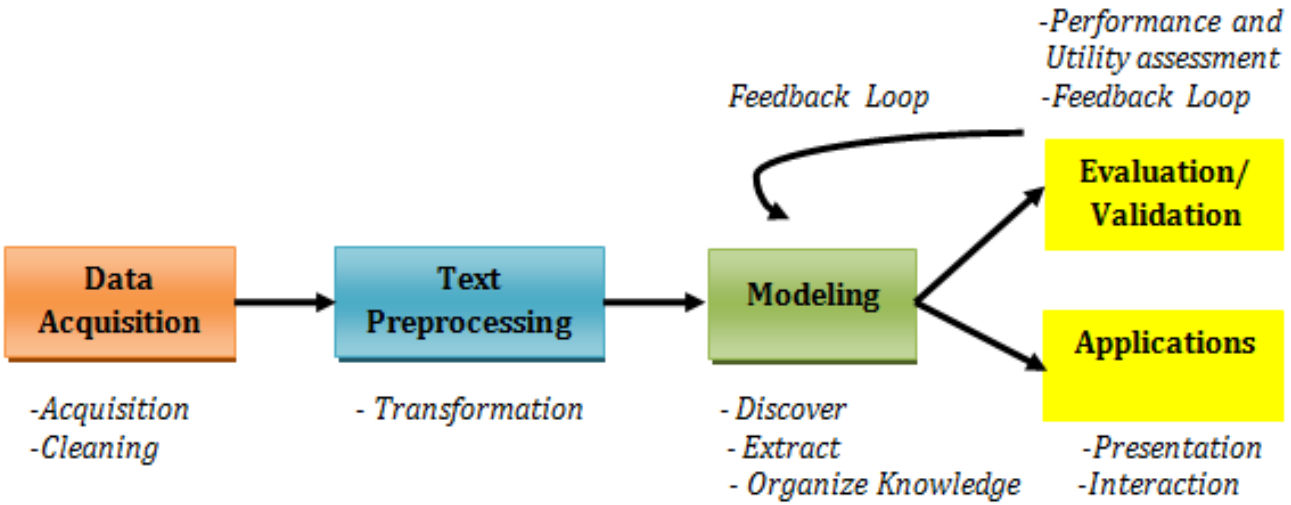
Intro to Text Mining

- Text mining, also known as text analysis, is the process of transforming unstructured text data into meaningful and actionable information.
- **Data helps companies get smart insights on people's opinions about a product or service.** Think about all the potential ideas that you could get from analyzing emails, product reviews, social media posts, customer feedback, support tickets, etc. On the other side, there's the dilemma of how to process all this data. And that's where text mining plays a major role.



Text Mining process

- **Text mining** combines notions of statistics, linguistics, and machine learning to create models that learn from training data and can predict results on new information based on their previous experience.



Text Analytics process

- **Text analytics,** on the other hand, uses results from analyses performed by text mining models, to create graphs and all kinds of data visualizations.

Basic Methods: Word Frequency

Word frequency can be used to identify the most recurrent terms or concepts in a set of data.

Finding out the most mentioned words in unstructured text can be particularly useful when analyzing customer reviews, social media conversations or customer feedback.

- For example, if the words “Expensive”, “Overpriced”, and “Overrated” frequently appear on your customer reviews, it may indicate you need to adjust your prices (or your target market!)

Basic Methods: Collocation

Collocation refers to a sequence of words that commonly appear near each other. The most common types of collocations are unigram, bigrams and trigrams

- **Bigrams** are pair of words that are likely to go together, like "**Get started**", "**Save time**", or "**Decision making**".
- **Trigrams** are a combination of three words, like "**Within walking distance**" or "**Keep in touch**".

Identifying collocations — and counting them as one single word — improves the granularity of the text, allows a *better understanding* of its semantic structure and, in the end, *leads to more accurate text mining results*.

Basic Methods: Concordance

- **Concordance** is used to recognize the particular context or instance in which a word or set of words appears. We all know that the human language can be ambiguous: the same word can be used in many different contexts. Analyzing the concordance of a word can help understand its exact meaning based on context.
- For example, here are a few sentences extracted from a set of r

Preceding context	Target	Following context
It saves time and helps teams	**work**	more efficiently.
Some advanced features only	**work**	in one language (English)
It enables us to	**work**	towards better conversion and retention.
We recommend this to several of the small businesses we	**work**	with, and they are all happy with the results.

Advanced Methods: Text Extraction

- **Text extraction** is a text analysis technique that extracts specific pieces of data from a text, like **keywords**, **entity names**, **addresses**, **emails**, etc. By using text extraction, companies can avoid all the hassle of sorting through their data manually to pull out key information. some of the main tasks of text extraction:
  - Keyword Extraction
  - Name Entity Recognition
  - Feature Extraction
- Most times, it can be useful to combine text extraction with text classification in the same analysis.

Text Extraction: Keyword Extraction

- **Keyword Extraction:** keywords are the most relevant terms within a text and can be used to summarize its content. Utilizing a keyword extractor allows you to index data to be searched, summarize the content of a text or create tag clouds, among other things.

Text Extraction: Name Entity Recognition

- **Named Entity Recognition** allows you to identify and extract the names of companies, organizations or persons from a text.

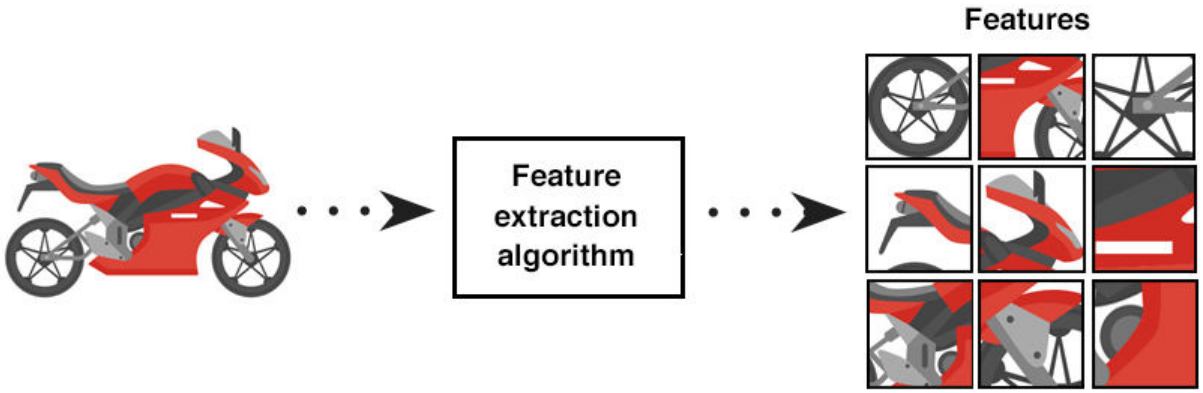
In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

"**NORP**" which represents "Nationalities or religious or political groups"
Geopolitical Entity (**GPE**).

Text Extraction: Feature Extraction

- **Feature Extraction** helps identify specific characteristics of a product or service in a set of data. For example, if you are analyzing product descriptions, you could easily extract features like "**colour**", "**brand**", "**model**", etc.



Advanced methods: Text Classification

- **Text classification** is the process of **assigning categories** (tags) to unstructured text data. This essential task of Natural Language Processing (NLP) makes it easy to organize and structure complex text, turning it into meaningful data. some of the most popular tasks of text classification are:

  - **Topic Analysis**
  - Language Detection
  - Intent Detection
  - Sentiment Analysis

Text Classification: Topic Analysis

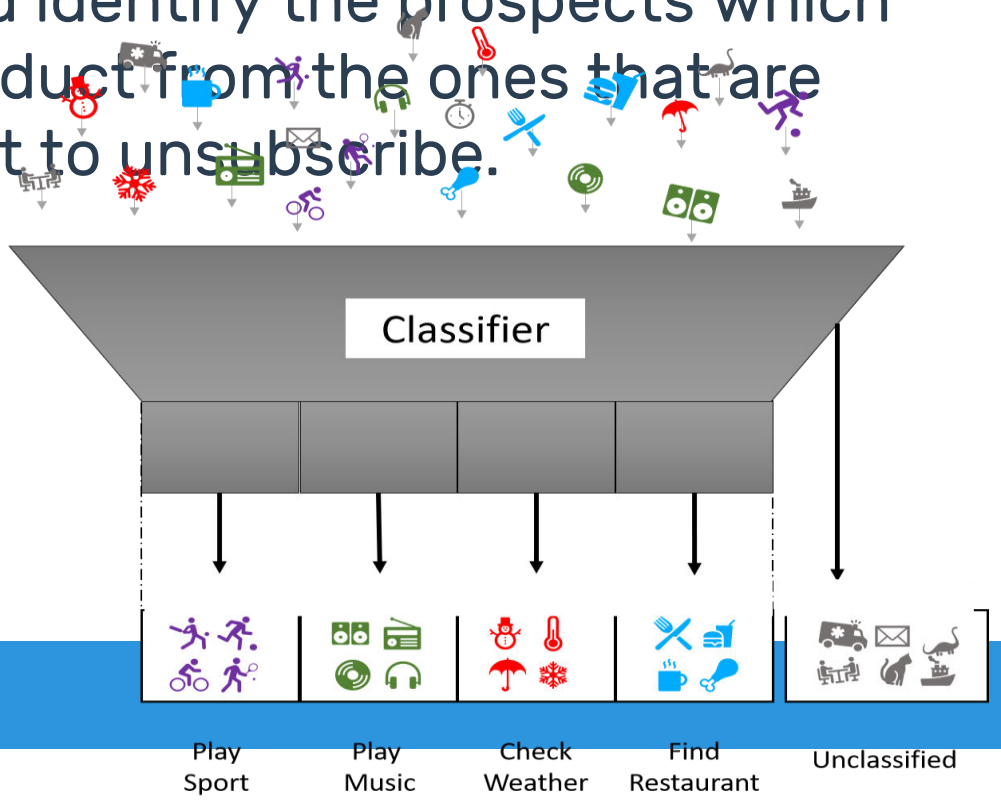
- **Topic Analysis** (also called **topic detection**, **topic modelling**, or **topic extraction**) is a machine learning technique that organizes and understands large collections of text data, by assigning "tags" or categories according to each individual text's topic or theme.
- For example, a support ticket saying *"My Online Order Hasn't Arrived"* can be classified as ***"Shipping Issues"***.

Text Classification: Language Detection

- **Language Detection** allows you to classify a text based on its language. One of its most useful applications is automatically routing support tickets to the right geographically located team. Automating this task is quite simple and helps teams save valuable time.

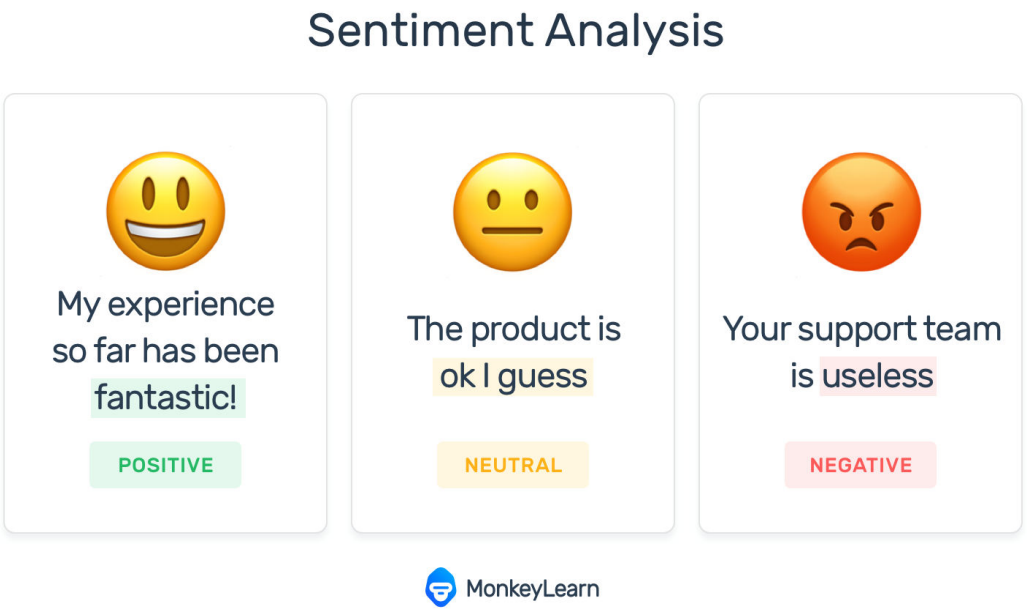
Text Classification: Intent Detection

- You could use a text classifier to recognize the intentions or the purpose behind a text automatically. This can be particularly useful when *analyzing customer conversations.*
- For example, you could sift through different outbound sales email responses and identify the prospects which are interested in your product from the ones that are not, or the ones who want to unsubscribe.



Text Classification: **Sentiment Analysis**

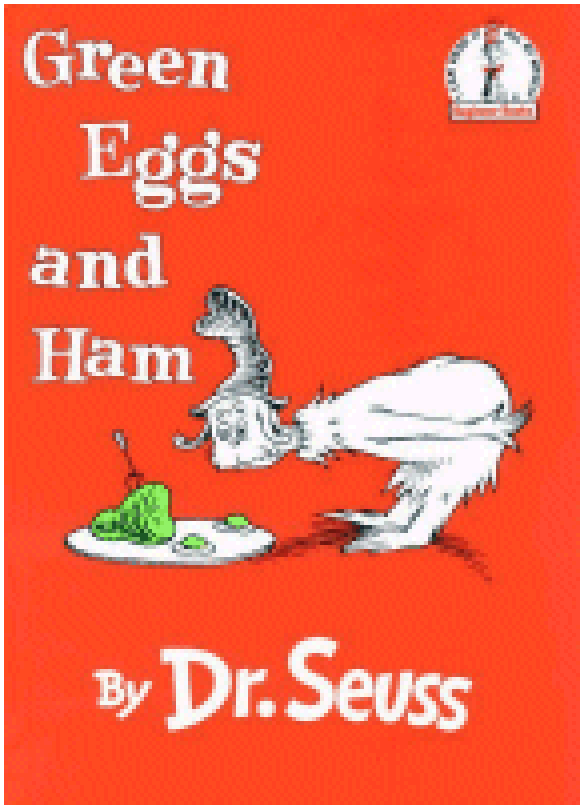
- **Sentiment analysis** (also known **as opinion mining or emotion AI**) refers to the use of natural language processing to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is used for many applications, especially in business intelligence. Some examples of applications for sentiment analysis include:
  - Analyzing the social media discussion around a certain topic
  - Evaluating survey responses
  - Determining whether product reviews are positive or negative



Text Analysis

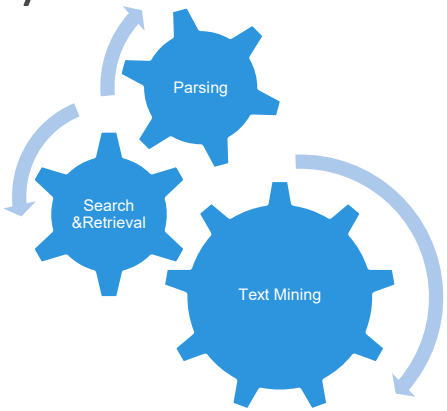
Encompasses the processing and representation of text for analysis and learning tasks

- **High-dimensionality**
  - Every distinct term is a dimension
  - *Green Eggs and Ham*: A 50-D problem!
- **Data is Un-structured**

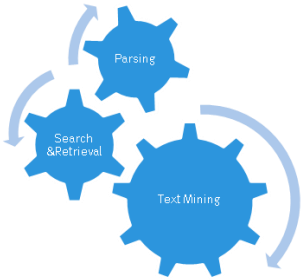


Text Analysis – Problem-solving Tasks

- **Parsing**
  - Impose a structure on the unstructured/semi-structured text for downstream analysis
- **Search/Retrieval**
  - Which documents have this word or phrase?
  - Which documents are about this topic or this entity?
- **Text-mining**
  - "**Understand**" the content
  - **Clustering**, **classification**
- **Tasks are not an ordered list**
  - Does not represent process
  - Set of tasks used appropriately depending on the problem addressed

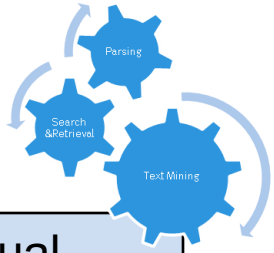


Example: Brand Management



- **Acme** currently makes two products
  - bPhone
  - bEbook
- They have lots of competition. They want to maintain their reputation for excellent products and keep their sales high.
- What is the buzz on **Acme**?
  - Search for mentions of **Acme** products
    - Twitter, Facebook, Review Sites, etc.
  - What do people say?
    - Positive or negative?
    - What do people think is good or bad about the products?

Buzz Tracking: The Process



1. Monitor social networks, review sites for mentions of our products.	**Parse** the data feeds to get actual content. Find and filter the raw text for product names (Use **Regular Expression**).
2. Collect the reviews.	Extract the relevant raw text. Convert the raw text into a suitable **document representation**. **Index** into our review **corpus**.
3. Sort the reviews by product.	**Classification** (or **"Topic Tagging"**)
4. Are they good reviews or bad reviews? We can keep a simple count here, for trend analysis.	**Classification** (sentiment analysis)
5. Marketing calls up and reads selected reviews in full, for greater insight.	**Search/Information Retrieval**.

Parsing the Feeds

1. Monitor social networks, review sites for mentions of our products

- Impose structure on semi-structured data.
- We need to know where to look for what we are looking for.

```
<channel>
<title>All about Phones</title>
<description>My Phone Review Site</description>
<link>http://www.phones.com/link.htm</link>

<item>
<title>bPhone: The best!</title>
<description>I love LOVE my bPhone!</description>
<link>http://www.phones.com/link.htm</link>
<guid isPermaLink="false"> 1102345</guid>
<pubDate>Tue, 29 Aug 2011 09:00:00 -0400</pubDate>
</item>

</channel>
```

Regular Expressions

1. Monitor social networks, review sites for mentions of our products

- Regular Expressions (regexp) are a means for finding words, strings or particular patterns in text.
- A match is a Boolean response. The basic use is to ask "does this regexp match this string?"

regexp	matches	Note
b[P|p]hone	bPhone, bphone	Pipe "|" means "or"
bEbo*k	bEbk, bEbok, bEbook, bEboook …	"*" matches 0 or more repetitions of the preceding letter
^I love	A line starting with "I love"	"^" means start of a string
Acme$	A line ending with "Acme"	"$" means the end of a string

Extract and Represent Text

2. Collect the reviews

Document Representation:

A structure for analysis

- **"Bag of words"**
  - common representation
  - A vector with one dimension for every unique term in space
    - **term-frequency (tf)**: number times a term occurs
  - Good for basic search, classification
- **Reduce Dimensionality**
  - Term Space – not ALL terms
    - no stop words: "the", "a"
    - often no pronouns
  - Stemming
    - "phone" = "phones"

"I love LOVE my bPhone!"

Convert this to a vector in the term space:

acme	0
bebook	0
bPhone	1
fantastic	0
love	2
slow	0
terrible	0
terrific	0

Document Representation - Other Features

2. Collect the reviews

- Feature:
  - Anything about the document that is used for search or analysis.
- Title
- Keywords or tags
- Date information
- Source information
- Named entities

Representing a Corpus (Collection of Documents)

2. Collect the reviews

- Reverse index
  - For every possible feature, a list of all the documents that contain that feature
- Corpus metrics
  - Volume
  - Corpus-wide term frequencies
  - Inverse Document Frequency (IDF)
    - more on this later
- Challenge: a Corpus is **dynamic**
  - Index, metrics must be updated continuously

Text Classification (I) - "Topic Tagging"

3. Sort the Reviews by Product

Not as straightforward as it seems

"The bPhone-5X has coverage everywhere. It's much less flaky than my old bPhone-4G."

"While I love Acme's bPhone series, I've been quite disappointed by the bEbook. The text is illegible, and it makes even my old Newton look blazingly fast."

"Topic Tagging"

3. Sort the Reviews by Product

Judicious choice of features

- Product mentioned in title?
- Tweet, or review?
- Term frequency
- Canonicalize abbreviations
  - "5X" = "bPhone-5X"

Text Classification (II) Sentiment Analysis

4. Are they good reviews or bad reviews?

- Naïve Bayes is a good first attempt
- But you need tagged training data!
  - The major bottleneck in text classification
- What to do?
  - Hand-tagging
  - Clues from review sites
    - thumbs-up or down, **# of stars**
  - **Cluster documents**, then label the clusters

Search and Information Retrieval

5. Marketing calls up and reads selected reviews in full, for greater insight.

- Marketing calls up documents with *queries*:
  - Collection of search terms
    - "**bPhone battery life**"
  - Can also be represented as "bag of words"
  - Possibly restricted by other attributes
    - within the last month
    - from this review site

Quality of Search Results

5. Marketing calls up and reads selected reviews in full, for greater insight.

- Is this document what I wanted?
- Used to rank search results

- Precision
  - What % of documents in the result are relevant?
- Recall
  - Of all the relevant documents in the corpus, what % were returned to me?

Computing Relevance (Term Frequency)

5. Marketing calls up and reads selected reviews in full, for greater insight.

- Assign each term in a document a weight for that term.
- The weight of a term *t* in a document *d* is a function of the number of times *t* appears in *d*.
  - The weight can be simply set to the number of occurrences of *t* in *d* :

$$tf\,(t, d) = count\,(t, d)$$

  - The term frequency may optionally be normalized.

Inverse Document Frequency (idf)

5. Marketing calls up and reads selected reviews in full, for greater insight.

$$idf(t) = log\,[N/df(t)]$$

- *N*: Number of documents in the corpus
- *df*(t): Number of documents in the corpus that contain a term *t*

- Measures term uniqueness in corpus
  - "phone" vs. "brick"
- Indicates the importance of the term
  - Search (relevance)
  - Classification (discriminatory power)

TF-IDF and Modified Retrieval Algorithm

5. Marketing calls up and reads selected reviews in full, for greater insight.

term t in document d:

$$tfidf(\mathrm{t}, \mathrm{d}) = \mathbf{tf\ (t, d) * idf(t)}$$

query: *brick, phone*

- Document with "brick" a few times more relevant than document with "phone" many times
- Measure of **Relevance with tf-idf**
- Call up all the documents that have any of the terms from the query, and sum up the tf-idf of each term:

$$\text{Relevance}(\mathrm{d}) = \sum_{i \in [1,n]} tfidf(\mathrm{t}_i, \mathrm{d})$$

TF-IDF and Modified Retrieval Algorithm, example

- The process to find meaning of documents using TF-IDF is very similar to Bag of words,
- Clean data / Preprocessing — Clean data (standardise data) , Normalize data(all lower case) , lemmatize data (all words to root words).
- Tokenize words with frequency
- Find TF for words
- Find IDF for words
- Vectorize vocab

example

- Let's cover an example of 3 documents -
- **Document 1** It is **going to** rain today. 1/6
- **Document 2** Today I am not **going** outside.
- **Document 3** I am **going to** watch the season premiere. 1/8

To find TF-IDF we need to perform the steps we laid out above, let's get to it.

- Step 1 Clean data and Tokenize

Word	Count
going	3
to	2
today	2
i	2
am	2
it	1
is	1
rain	1

Example, continue

- Step 2 Find TF for all docs

TF = (Number of repetitions of word in a document) / (# of words in a document)

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

Example, continue

- Step 3: Find IDF:

IDF =Log[(Number of documents) / (Number of documents containing the word)]

In Excel use LN(3/3)

Words	IDF Value
going	log(3/3)
to	log(3/2)
today	log(3/2)
i	log(3/2)
am	log(3/2)
It	log(3/1)
is	log(3/1)
rain	log(3/1)

Example, continue

- Step 4: Build model i.e. stack all words next to each other

Words	IDF Value
going	0
to	0.41
today	0.41
i	0.41
am	0.41
It	1.09
is	1.09
rain	1.09

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

IDF Value and TF value of 3 documents.

Example, continue

- Step 5: Compare results and use table to ask questions

e.g., **0.41*0.12**

Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

Remember, the final equation = TF-IDF = TF * IDF

Example, continue- Analysis and outcomes

- You can easily see using this table that words like **‘it’,’is’,’rain’** are important for document 1 but not for document 2 and document 3 which means Document 1 and 2&3 are different w.r.t talking about rain.
- You can also say that Document 1 and 2 talk about something **‘today’**, and document 2 and 3 discuss something about the writer because of the word **‘I’.**
- This table helps you find similarities and non similarities between documents, words and more much better than Bag Of Words.

Other Relevance Metrics

5. Marketing calls up and reads selected reviews in full, for greater insight.

- "Authoritativeness" of source
  - PageRank is an example of this
- Recency of document
- How often the document has been retrieved by other users

Effectiveness of Search and Retrieval

- Relevance metric
  - important for precision, user experience
- **Effective crawl, extraction, indexing**
  - important for recall (and precision)
  - more important, often, than retrieval algorithm
- **MapReduce**
  - Reverse index, corpus term frequencies, idf

Natural Language Processing

- Unstructured text mining means extracting "features"
  - Features are structured meta-data representing the document
  - Goal: "vectorize" the documents
- After vectorization, apply advanced machine learning techniques
  - Clustering
  - Classification
    - Decision Trees
    - Naïve Bayesian Classifier
  - **Scoring**
    - Once models have been built, use them to automatically categorize incoming documents

Example: UFOs Attack

July 15th, 2010. Raytown, Missouri

When I fist noticed it, I wanted to freak out. There it was an object floating in on a direct path, It didn't move side to side or volley up and down. It moved as if though it had a mission or purpose. I was nervous, and scared, So afraid in fact that I could feel my knees buckling. I guess because I didn't know what to expect and I wanted to act non aggressive. I though that I was either going to be taken, blasted into nothing, or…

Q: What is the witness describing?

A: An encounter with a UFO.

Q: What is the emotional state of the witness?

A: Frightened, ready to flee.

Source: http://www.infochimps.com/datasets/60000-documented-ufo-sightings-with-text-descriptions-and-metada

Example: UFOs Attack

If we really are on the cusp of a major alien invasion, eyewitness testimony is the key to our survival as a species.

*Strangely, the computer finds this account **unreliable!***

*When I **fist** noticed it, I wanted to freak out ject floating in on a direct path, It **didn't** move side to side or volley up n. It moved as if though it had a mission or purpose. I was and scared, **So afraid in fact** that I could feel my knees buckling. I guess because I didn't kno d I wanted to h that I was either going to be **taken**, blasted into nothing, or…*

Machine error

Typo

Turn of phrase

Ambiguous meaning

"UFO" keyword missing

Source: http://www.infochimps.com/datasets/60000-documented-ufo-sightings-with-text-descriptions-and-metada

Example: UFOs Attack

Investigators need to…

Search — *for keywords and phrases, but your topic may be very complicated or keywords may be misspelled within the document*

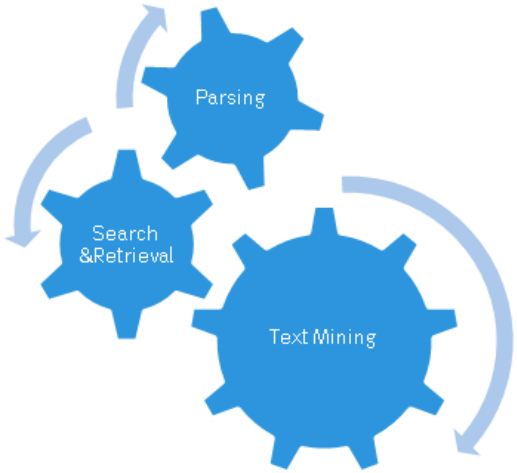
Manage — *document meta-data like **time, location and author**. Later retrieval may be key to identifying this meta-data early, and the document may be amenable to structure.*

Understand — *content via **sentiment analysis**, custom dictionaries, natural language processing, clustering, classification and good ol' domain expertise.*

…with computer-aided text mining

Challenges - Text Analysis

1. Finding the right structure for your unstructured data
2. Very high dimensionality
3. Thinking about your problem the right way

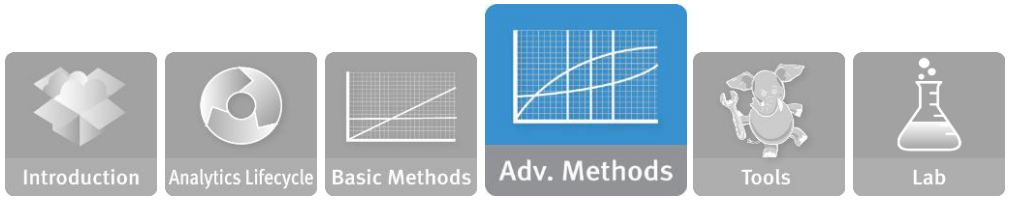


Module 4: Advanced Analytics – Theory and Methods

Lesson 8: Text Analysis - Summary

During this lesson the following topics were covered:

- Challenges with text analysis
- Key tasks in text analysis
- Definition of terms used in text analysis
  - Term frequency, inverse document frequency
- Representation and features of documents and corpus
- Use of regular expressions in parsing text
- Metrics used to measure the quality of search results
  - Relevance with tf-idf, precision and recall



Module 4: Summary

Key Topics Covered in this module	Methods Covered in this module
Algorithms and technical foundations	Categorization (unsupervised) : K-means clustering Association Rules
Key Use cases	Regression Linear Logistic
Diagnostics and validation of the model	Classification (supervised) Naïve Bayesian classifier Decision Trees
Reasons to Choose (+) and Cautions (-) of the model	Time Series Analysis
Fitting, scoring and validating model in R and in-db functions	Text Analysis

Practice 1: Text Analysis : Upload file and Load Libraries

- # Load data

```
setwd("C:\\Users\\z10095\\Desktop\\")
myFile  <- 'shz.txt' # the local file path to my research prospectus

# fill = TRUE b/c rows are of unequal length
#fill: Sometimes, we may get a file that contains the
# unequal length of rows, and we have to add blank spaces to that missing values.
dat <- read.table(myFile, header = FALSE, fill = TRUE)
dat
```

Load required libraries

```
library(dplyr) # for data wrangling
library(tidytext) # for Natural Language Processing
library(stringr) # to deal with strings
library(wordcloud) # to for generating word clouds
library(knitr) # for tables,  It combines many features into one package
library(DT) # for dynamic tables
library(tidyr)
#Tools to help to create tidy data, where each column is a variable,
#each row is an observation, and each cell contains a single value.
```

Format the text file
reshape the .txt data frame into one column

- tidy_dat <- tidyr::gather(dat, key, word) %>% select(word)
- # 1:8 %>% sum %>% sqrt. **# Answer is 6**
- tidy_dat

```
2 Understanding
3 Born
4 Life,
5 Through
6 The
7 In
8 He
9 In
10 In
11 Eventually
12 The
13 While
14 One
```

- 1:8 %>% sum %>% sqrt
- ## or:
- x=1:8
- x
- y=sum(x)
- Y #36
- sqrt(y)

- Both gives you same answer: 6

- The gather() Function
- The second tidyr function we will look into is the gather() function. With gather() it may not be clear what exactly is going on, but in this case we actually have a lot of column names the represent what we would like to have as data values.
- data is the dataframe you are working with.
- key is the name of the key column to create.
- value is the name of the value column to create.

Count # of words

```
tidy_dat$word %>% length() #there are 2832 tokens in my document
```

```
[1] 2832
```

Count # of unique words

```
unique(tidy_dat$word) %>% length() # 695 words are unique
```

```
[1] 695
```

- **unnest_tokens: Split a column into tokens**

- semi_join(x, y): **Return all rows from x where there are matching values in y, keeping just columns from x**. A semi join differs from an inner join because an **inner join** will **return one row of x for each matching row of y**, where a **semi join will never duplicate rows of x**. This is a filtering join.

Semi join and inner join

anti_join came in handy for us in a setting where we were trying to re-create an old table from the source data. We then wanted to be able to **identify the records from the original table that did not exist in our updated table**. Good example http://zevross.com/blog/2014/08/05/using-the-r-function-anti_join-to-find-unmatched-records/

Tokenize the text and Frequency

```
tokens <- tidy_dat %>%
 unnest_tokens(word, word) %>%
 dplyr::count(word, sort = TRUE) %>%
 ungroup()
tokens %>% head(10)
```

```
tokens %>% head(10)
# A tibble: 10 x 2
word n <chr> <int>
1 the   137
2 of     73
3 and    62
4 in     45
5 zayed  45
6 to     40
7 sheikh 39
8 a      32
9 his    25
10 was   22
```

Draw the Cloud

```
tokens %>%
  with(wordcloud(word, n, random.order = FALSE, max.words = 50,
colors=pal))
#
# other choices
#colors=brewer.pal(8, “Dark2″)
tokens %>% with(wordcloud(word, n,
random.order = FALSE, max.words = 50,
colors=brewer.pal(8, "Dark2")))

tokens %>% with(wordcloud(word, n,
random.order = FALSE, max.words = 50,
colors="#AD1DA5"))
```

random.order
plot words in random order. If false, they will be plotted in decreasing frequency

Remove Stop Words

```
# remove stop words
data("stop_words")
tokens_clean <- tokens %>%
  anti_join(stop_words,by = "word")
```

identify the records from the original table that did not exist in our updated table

Remove numbers

```
# remove numbers
nums <- tokens_clean %>% filter(str_detect(word, "^[0-9]")) %>%
select(word) %>% unique()

tokens_clean <- tokens_clean %>%
  anti_join(nums, by = "word")
```

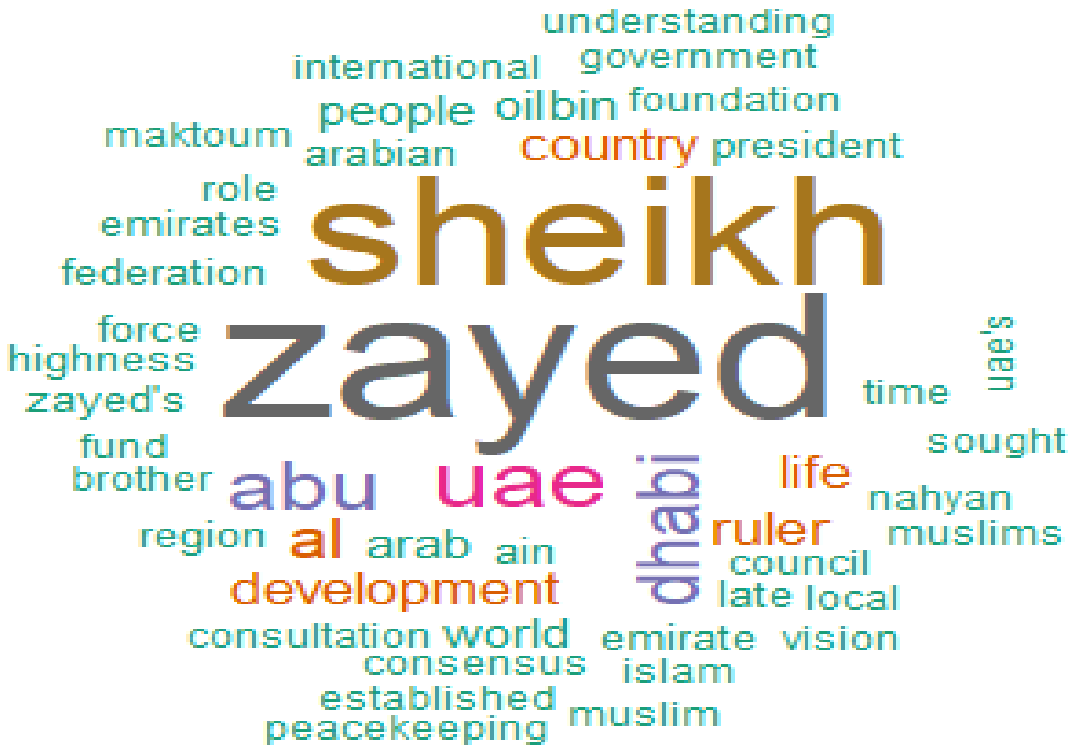
Remove other stop words

```
# remove other stop words
uni_sw <- data.frame(word = c("i.e", "e.g."))

tokens_clean <- tokens_clean %>%
  anti_join(uni_sw, by = "word")
```

Draw the Cloud again

```
# define a nice color palette
pal <- brewer.pal(8,"Dark2")

# plot the 50 most common words
tokens_clean %>%
  with(wordcloud(word, n, random.order = FALSE, max.words = 50,
colors=pal))
```

Word Cloud after cleaning



Clean data and search

```
tokens_clean %>%
  DT::datatable()
tokens_clean
```

Show 10 entries

Search:

	word	n
1	zayed	45
2	sheikh	39
3	uae	17
4	abu	14
5	dhabi	12
6	al	10
7	development	8
8	ruler	7
9	country	6
10	life	6

Showing 1 to 10 of 491 entries

Previous 1 2 3 4 5 … 50 Next

Practice : Sentiment Analysis

```
# Load the library
library(RSentiment)
calculate_total_presence_sentiment(c("This is a good text", "This is a bad text",
"This is a really bad text", "This is horrible"))
```

```
#output

[1] "Processing sentence: this is a good text"
[1] "Processing sentence: this is a bad text"
[1] "Processing sentence: this is a really bad text"
[1] "Processing sentence: this is horrible"

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] "Sarcasm" "Negative" "Very Negative" "Neutral" "Positive" "Very Positive"
[2,] "0“         "3”         "0“             "0“      "1“          "0"
```

calculate_sentiment

```
calculate_sentiment(c("This is a good text",
                      "This is a bad text",
                       "This is a really bad text", "This is horrible"))

#output
[1] "Processing sentence: this is a good text"
[1] "Processing sentence: this is a bad text"
[1] "Processing sentence: this is a really bad text"
[1] "Processing sentence: this is horrible"

        text sentiment

1 This is a good text Positive
2 This is a bad text Negative
3 This is a really bad text Negative
4 This is horrible Negative
```

Calculating Sentiment Score

```
calculate_score(c("This is a good text",
                  "This is a bad text",
                  "This is a really bad text",
                  "This is horrible"))
```

```
[1] "Processing sentence: this is a good text"
[1] "Processing sentence: this is a bad text"
[1] "Processing sentence: this is a really bad text"
[1] "Processing sentence: this is horrible"
```

```
 [1] 1 -1 -1 -1
```

Text Mining and NLP More Examples

String Matching in R Programming

- String matching is an important aspect of any language. It is useful in finding, replacing as well as removing string(s)
- A regular expression is a string that contains special symbols and characters to find and extract the information needed from the given data.
- **Operations on String Matching**
  - **Finding a String**
  - **grep() function:** It returns the index at which the pattern is found in the vector.
  - *grep(pattern, string, ignore.case=FALSE)*
  - str <- c("Man", "woman","baby", "amman", "happy")
  - grep('man', str)

```
> grep('man', str) [1] 2 4
```

String Matching in R Programming

- str <- c("Man", "woman","baby", "amman", "happy")
- grep('man', str, ignore.case ="True")

```
grep('man', str, ignore.case ="True")
[1] 1 2 4
```

grepl() function: It is a logical function that returns the value **True** if the specified pattern is found in the vector and **false** if it is not found.

Syntax

grepl(pattern, string, ignore.case=FALSE)

To find whether any instance(s) of 'the' are present in the string.

```
str <- c("Man", "woman","baby", "amman", "happy")
grepl('the', str)
[1] FALSE FALSE FALSE FALSE FALSE
```

```
grepl('wo', str)
[1] FALSE TRUE FALSE FALSE FALSE
```

String Matching in R Programming

- **regexpr() function:** It searches for occurrences of a pattern in every element of the string.
- ***Syntax:***
  regexpr*(pattern, string, ignore.case = FALSE)*

example: To find whether any instance(s) of 'he' is present in each string of the vector.

```
str <- c("Hello", "hello", "hi", "ahey", "aahead")
regexpr('he', str)
```

```
➢ regexpr('he', str)
➢ [1] -1 1 -1 2 3
```

example: To find whether any instance(s) of words starting with a vowel is present in each string of the vector.

```
str <- c("abra", "Ubra", "hunt", "quirky")
regexpr('^[aeiouAEIOU]', str)
```

```
➢ regexpr('^[aeiouAEIOU]', str)
➢  [1] 1 1 -1 -1
```

Finding and Replacing Strings in R: sub() and gsub().

In order to **search and replace** a particular string, we can use two functions namely, **sub()** and **gsub().**

sub replaces the only first occurrence of the string to be replaced and returns the modified string.

gsub() replaces all occurrences of the string to be replaced and returns the modified string.

Syntax:

sub(pattern, replaced_string, string)

gsub(pattern, replaced_string, string)

Example : To replace the first occurrence of 'he' with 'aa'

```
str = "heutabhe"
sub('he', 'aa', str)
```

```
➢ sub('he', 'aa', str)
➢ [1] "aautabhe"
```

Example : To replace all occurrences of 'he' with 'aa'

```
str = "heutabhe"
gsub('he', 'aa', str)
```

```
➢ gsub('he', 'aa', str)
➢ [1] "aautabaa"
```

Finding and Removing Strings in R: str_remove() and str_remove_all().

str_remove() removes the only first occurrence of the string/pattern to be removed and returns the modified string.

str_remove_all() removes all occurrences of the string to be removes and returns the modified string.

Syntax:

str_remove(string, pattern, ignore.case=False)

Example : Removing the first occurrence of **vowels** in the vector

```
library(stringr)
x <- c("apple", "pear", "banana", "orange")
str_remove(x, "[aeiou]")
```

```
➢ str_remove(x, "[aeiou]")
➢ [1] "pple" "par" "bnana" "range"
```

Example : Removing all occurrences of vowels in the vector

```
library(stringr)
x <- c("apple", "pear", "banana", "orange")
str_remove_all(x, "[aeiou]")
```

```
➢ str_remove_all(x, "[aeiou]")
➢  [1] "ppl" "pr" "bnn" "rng"
```

More examples, text mining applications on novels

- ***Sense and Sensibility*** is a novel by Jane Austen, published in 1811. It was published anonymously; *By A Lady* appears on the title page where the author's name might have been. It tells the story of the Dashwood sisters, Elinor (age 19) and Marianne (age 16½) as they come of age. They have an older half-brother, John, and a younger sister, Margaret (age 13).
- ***Pride and Prejudice*** is an 1813 romantic novel of manners written by Jane Austen. The novel follows the character development of Elizabeth Bennet, the dynamic protagonist of the book who learns about the repercussions of hasty judgments and comes to appreciate the difference between superficial goodness and actual goodness. Its humour lies in its honest depiction of manners, education, marriage, and money during the Regency era in Great Britain.
- ***Mansfield Park*** is the third published novel by Jane Austen, first published in 1814 by Thomas Egerton. A second edition was published in 1816 by John Murray, still within Austen's lifetime. The novel did not receive any public reviews until 1821.

https://en.wikipedia.org/

More examples, text mining applications on novels

- ***Emma***, by Jane Austen, is a novel about youthful hubris and romantic misunderstandings. It is set in the fictional country village of Highbury and the surrounding estates of Hartfield, Randalls and Donwell Abbey, and involves the relationships among people from a small number of families.[2] The novel was first published in December 1815

- ***Northanger Abbey****: This article is about the 1817 novel. For adaptations of the novel, see Jane Austen in popular culture § Northanger Abbey (1817).*

- ***Persuasion*** is the last novel fully completed by Jane Austen. It was published at the end of 1817, six months after her death.

  The story concerns Anne Elliot, a young Englishwoman of twenty-seven years, whose family moves to lower their expenses and reduce their debt by renting their home to an Admiral and his wife. The wife's brother, Navy Captain Frederick Wentworth, was engaged to Anne in 1806, but the engagement was broken when Anne was "persuaded" by her friends and family to end their relationship. Anne and Captain Wentworth, both single and unattached, meet again after a seven-year separation, setting the scene for many humorous encounters as well as a second, well-considered chance at love and marriage for Anne in her second "bloom".

https://en.wikipedia.org/

Libraries used in R for text analytics

library(tidytext)

library(tidyverse)

library(janeaustenr)

library(stringr)

library(wordcloud)

library(reshape2)

library(textdata)

Dictionaries used for sentiments

```
get_sentiments("afinn")
get_sentiments("bing")
get_sentiments("nrc")

data(sentiments)
#dataset structure
str(sentiments)
```

```
> get_sentiments("afinn")
# A tibble: 2,477 x 2 word value <chr> <dbl>
1 abandon -2
2 abandoned -2
3 abandons -2
4 abducted -2
5 abduction -2
6 abductions -2
7 abhor -3
8 abhorred -3
9 abhorrent -3
10 abhors -3 # ... with 2,467 more rows
```

```
> get_sentiments("bing")
# A tibble: 6,786 x 2 word sentiment <chr>
  <chr>
1 2-faces negative
2 abnormal negative
3 abolish negative
4 abominable negative
5 abominably negative
6 abominate negative
7 abomination negative
8 abort negative
9 aborted negative
10 aborts negative
```

```
> get_sentiments("nrc")
 # A tibble: 13,901 x 2 word sentiment <chr> <chr>
1 abacus trust
2 abandon fear
3 abandon negative
4 abandon sadness
5 abandoned anger
6 abandoned fear
7 abandoned negative
8 abandoned sadness
9 abandonment anger
10 abandonment fear # ... with 13,891 more rows
```

Sample output for **afinn** lexicon

```
afinn_lexicon <- get_sentiments("afinn")
head(afinn_lexicon)
## # A tibble: 6 × 2
```

Output

```
##       word       score
## 1    abandon       -2
## 2   abandoned     -2
```

Sample output for **nrc** lexicon

```
#NRC
nrc_lexicon <- get_sentiments("nrc")
head(nrc_lexicon)
```

Output:

```
## # A tibble: 6 × 2
##         word     sentiment
##        <chr>       <chr>
## 1    abacus         trust
## 2    abandon        fear
```

Sample output for **bing** lexicon

```
#BING
bing_lexicon <- get_sentiments("bing")
head(bing_lexicon)
```

Output

```
## # A tibble: 6 × 2
##         word     sentiment
##        <chr>         <chr>
## 1    2-faced      negative
## 2    2-faces      negative
## 3         a+      positive
## 4   abnormal      negative
## 5    abolish      negative
```

#**Get the emma book and transform it into a tidy dataset**

```
tidy_books <- austen_books() %>%
   filter(book == "Emma") %>%
   group_by(book) %>%
```

Using **row_number**() with **mutate**() will create a column of consecutive **numbers**. The **row_number**() function is useful for creating an identification **number** (an ID variable). It is also useful for labeling each observation by a grouping variable.

##cumsum() function in **R** Language is used to calculate the **cumulative sum** of the vector passed as ## argument

str_detect function returns a logical **value** (i.e. FALSE or TRUE),

```
 mutate(linenumber = row_number(),
                chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",
                                            ignore_case = TRUE)))) %>%   ungroup() %>%
  unnest_tokens(word, text)
```

https://www.tidytextmining.com/tidytext.html

nrc lexicon associated with joy

```
#Using the nrc lexicon, only the words that are associated to a sentiment of `joy`
nrc_joy <- get_sentiments("nrc") %>%
   filter(sentiment == "joy")
#Summarize the usage of `joy` words
tidy_books %>%
  semi_join(nrc_joy) %>%
  count(word, sort = T)
```

Output

```
## # A tibble: 303 × 2
##      word      n
##     <chr>    <int>
## 1    good    359
## 2   young   192
## 3  friend    166
## 4    hope    143
```

Semi join and inner join

- semi_join(x, y): **Return all rows from x where there are matching values in y, keeping just columns from x**. A semi join differs from an inner join because an **inner join** will return one row of x for each matching row of y, where a semi join will never duplicate rows of x. This is a filtering join.

  anti_join came in handy for us in a setting where we were trying to re-create an old table from the source data. We then wanted to be able to identify the records from the original table that did not exist in our updated table. Good example http://zevross.com/blog/2014/08/05/using-the-r-function-anti_join-to-find-unmatched-records/

Application in r

Basically, the **mutate** function in R programming is used to create new variables. It is used to generate new variables from data sets.

```
tidy_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",
                                                 ignore_case = TRUE)))) %>%
  ungroup() %>%
   unnest_tokens(word, text)
```

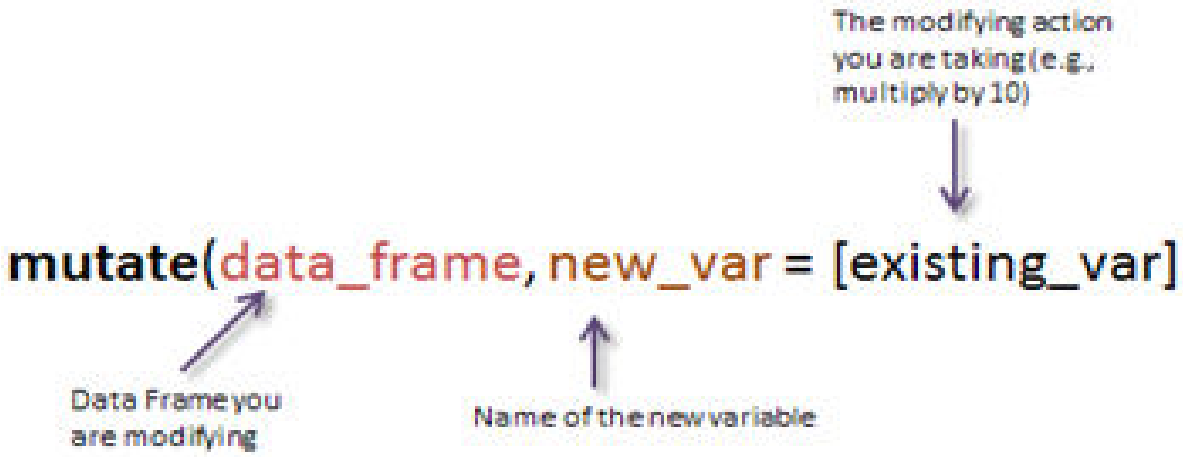
convert the text to the tidy format using unnest_tokens()

https://www.tidytextmining.com/sentiment.html

A side example, understand mutate

- #install.packages("dplyr")
- library(dplyr)
- mtcars

- mtcares2 = mutate(mtcars, mtcars_new = mpg/cyl)



Undestand mutate and group by

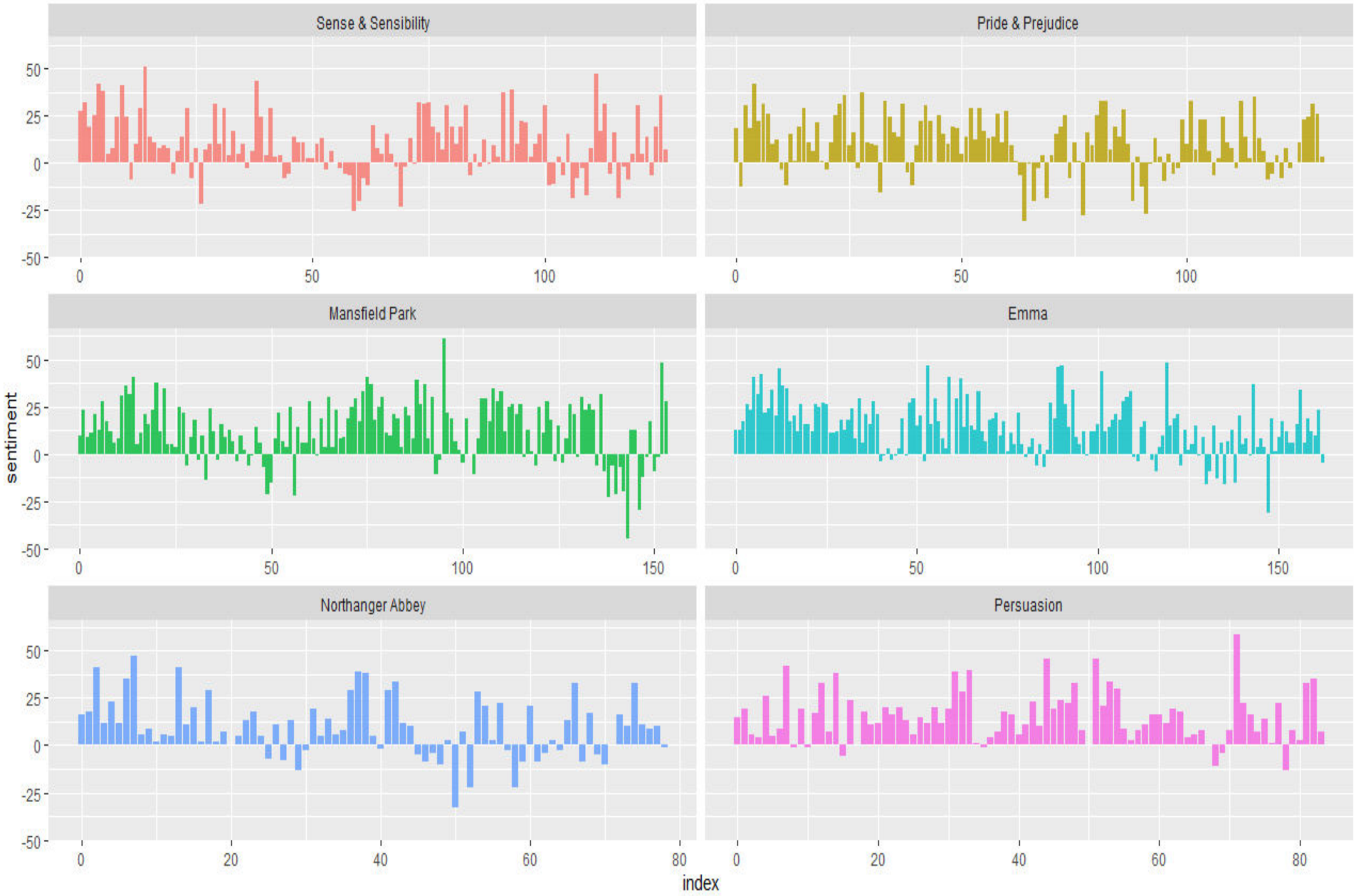
```
## Creating identification number to represent 50 individual people
ID <- c(1:20)

## Creating sex variable (10 males/10 females)
Sex <- rep(c("male", "female"), 10) # rep stands for replicate

## Creating age variable (20-39 year olds)
Age <- c(26, 25, 39, 37, 31, 34, 34, 30, 26, 33,
         39, 28, 26, 29, 33, 22, 35, 23, 26, 36)

## Creating a dependent variable called Score
Score <- c(0.010, 0.418, 0.014, 0.090, 0.061, 0.328, 0.656, 0.002, 0.639, 0.173,
           0.076, 0.152, 0.467, 0.186, 0.520, 0.493, 0.388, 0.501, 0.800, 0.482)

## Creating a unified dataset that puts together all variables
## tibble is a simple dataframe
data <- tibble(ID, Sex, Age, Score)
## group by sex
data %>%
  group_by(Sex) %>%
  summarize(m = mean(Score), # calculates the mean
            s = sd(Score),   # calculates the standard deviation
            n = n()) %>%     # calculates the total number of observations
  ungroup()
##`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 2 x 4
##x        m     s     n
##hr>  <dbl> <dbl> <int>
##1 female 0.282 0.184    10
##2 male   0.363 0.300    10


## mutate() and group_by()
data %>%
  group_by(Sex) %>%
  mutate(m = mean(Score)) %>% # calculates mean score by Sex
  ungroup()
```

```
janeaustensentiment <- tidy_books %>%
    inner_join(get_sentiments("bing")) %>%
    count(book, index = linenumber %/% 100, sentiment) %>%

  spread(sentiment, n, fill = 0) %>%

  mutate(sentiment = positive - negative)
```

```
head(janeaustensentiment)
## Source: local data frame [6 x 5]
## Groups: book, index [6]
##
##                book index negative positive sentiment
##              <fctr> <dbl>    <dbl>    <dbl>     <dbl>
## 1 Sense & Sensibility     0       20       47        27
## 2 Sense & Sensibility     1       22       54        32
## 3 Sense & Sensibility     2       16       35        19
## 4 Sense & Sensibility     3       20       45        25
## 5 Sense & Sensibility     4       21       63        42
```

Plot Sentiment

```
ggplot(data = janeaustensentiment, mapping = aes(x = index, y = sentiment, fill = book)) +
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
   facet_wrap(facets = ~ book, ncol = 2, scales = "free_x")
```



Extra References

https://www.geeksforgeeks.org/string-matching-in-r-programming/

https://en.wikipedia.org/

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3