**Name:** Sayli Bhor **Academic Year:**2021-2022

**Class:** Msc CS Part I **Roll no:**48

**Subject:** Business Intelligence and Big Data Analytics

**Index**

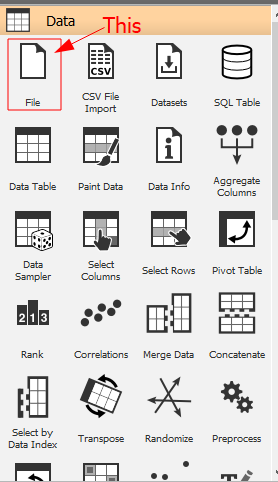
|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Title** | **Page No.** |
| 1 | Classification using orange  tool | 2 |
| 2 | Text Classification using  orange tool | 13 |
| 3 | Image Classification using  orange tool | 29 |
| 4 | Hierarchical clustering  using orange tool | 41 |
| 5 | Text Preprocessing using  orange tool | 49 |
| 6 | To predict whether the  given dataset is of fruit or vegetable using orange | 59 |
| 7 | To Predict whether using  orange tool | 63 |
| 8 | Write a java program to Calculate the Alon Matias  Szegedy Algorithm for given stream. | 69 |
| 9 | Write a Program to Construct different types of K-shingles for a given  document | 72 |
| 10 | Write a program for measuring similarity among documents and detecting passages which have been  reused. | 76 |
| 11 | Write a java Program to demonstrate the k-moments for zeroth, first and second  moments | 81 |

# Practical 1

**Aim:** Classification using orange.

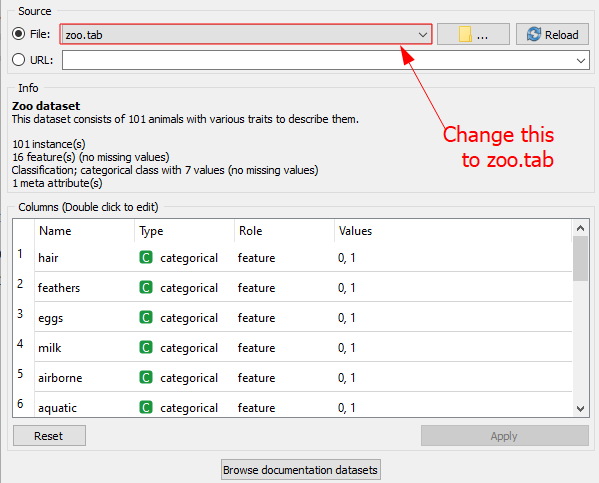
## Steps:

* Drag and drop a Filewidget from the Datasection found in the left panel to the workspace.

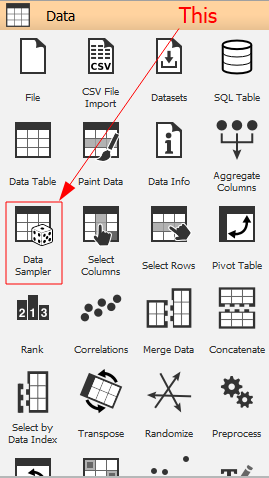




* Double click the Filewidget and choose the zoo.tabdataset.

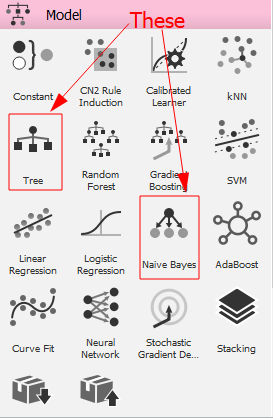


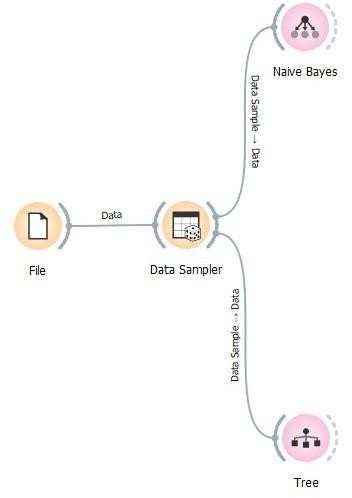
* Drag and drop a Data Samplerwidget onto the workspace and connect it to the File widget. This widget is found in the Data section.



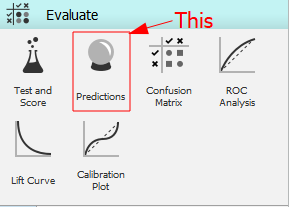


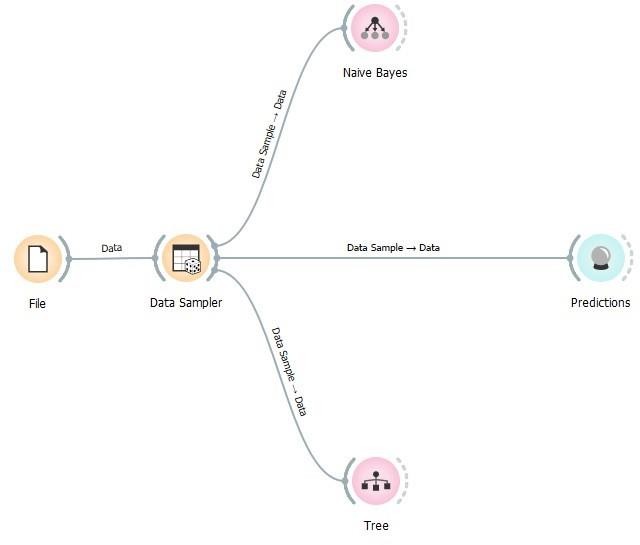
* We choose the Naive Bayes’ and Classification Treemodels to classify our data. Drag and drop these widgets to the workspace. They are found under the Modelssection.



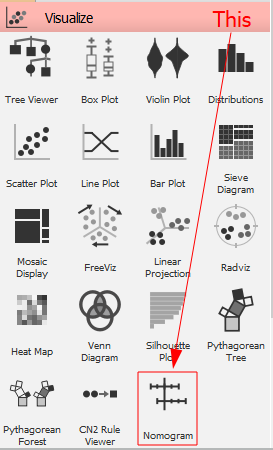


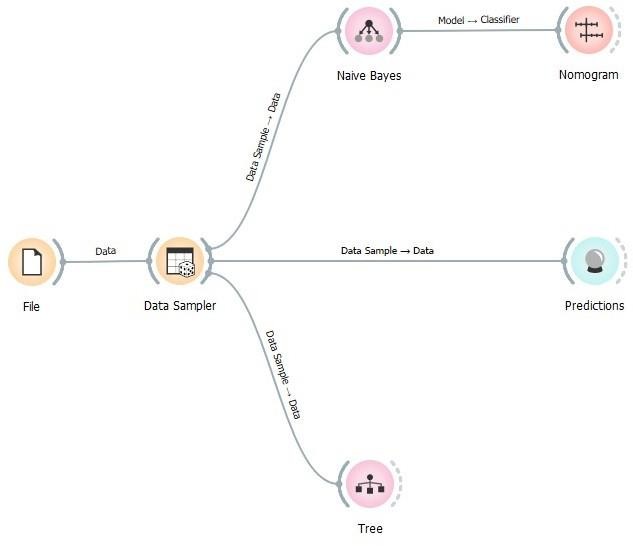
* To perform predictions on the data, we drag and drop the Predictions widget onto the workspace.This widget is found in the Evaluate section.



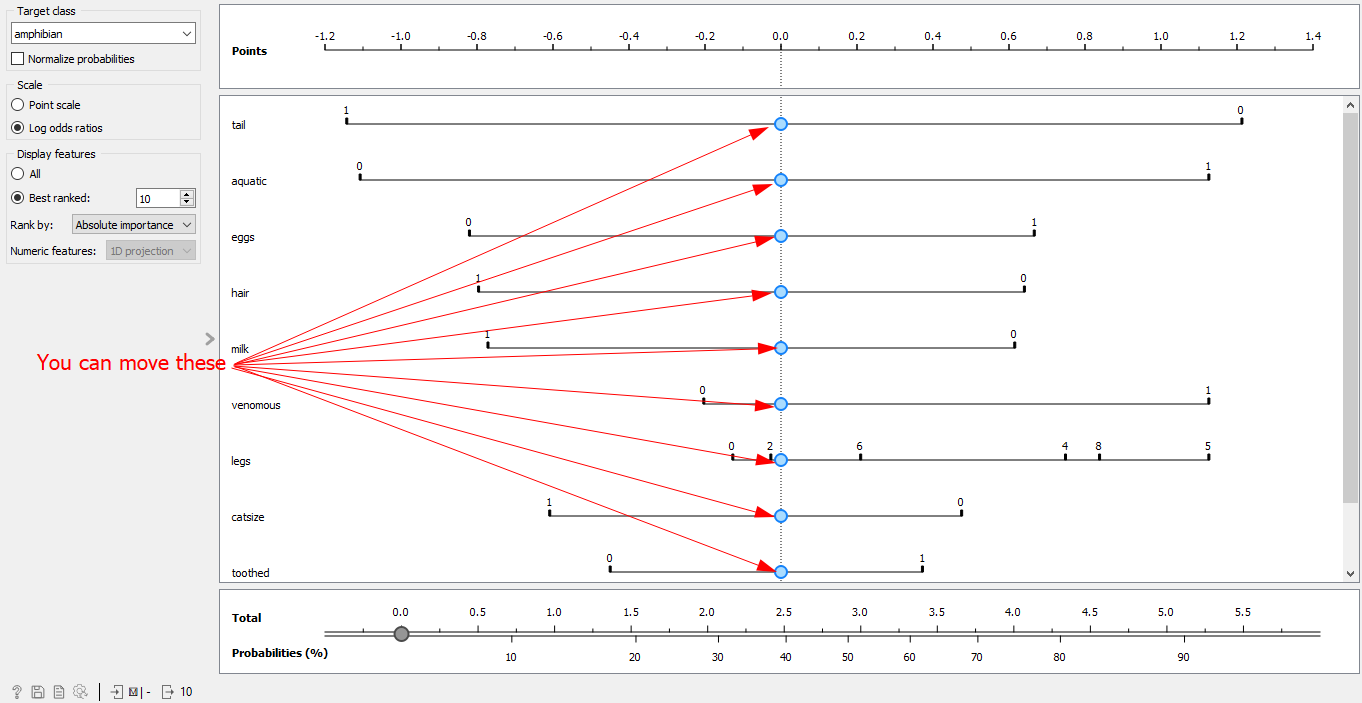


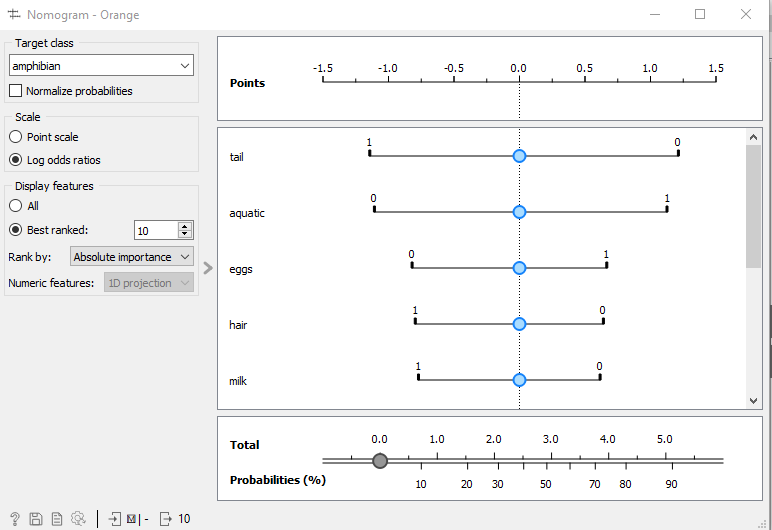
* A Nomogram is useful to view data from a Naive Bayes’ model. Drag and drop this widget onto the workspace. It can be found in the Visualise section.

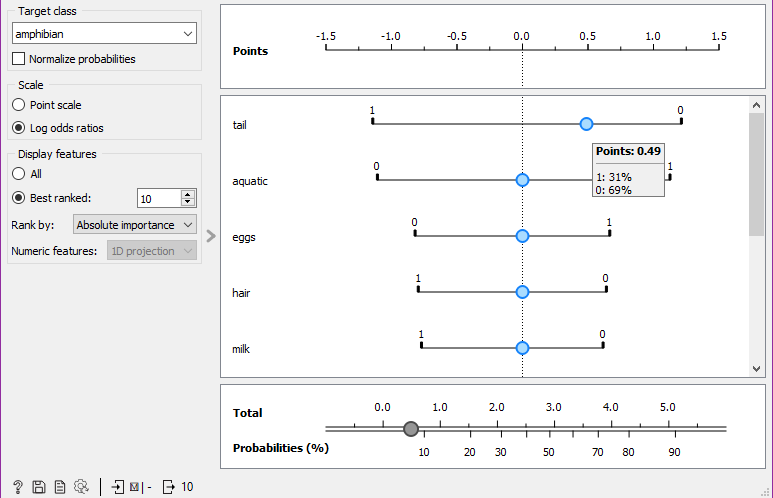




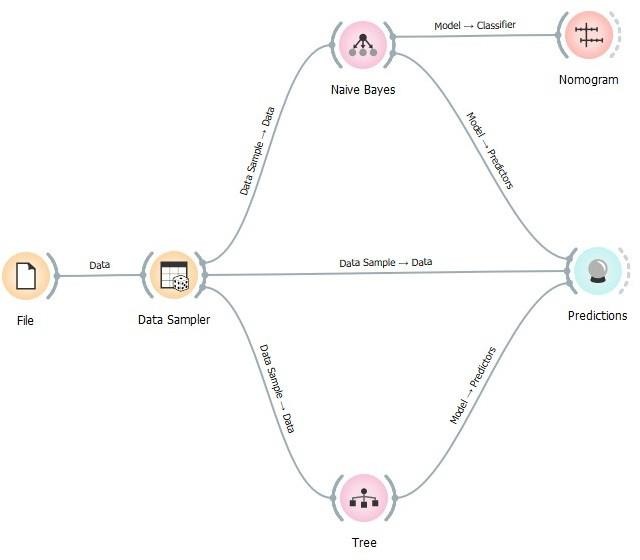
* You can move the points in the Nomogram to see the probabilities of a particular class. Here 1 indicates favourable probability while 0 indicates unfavourable probability.



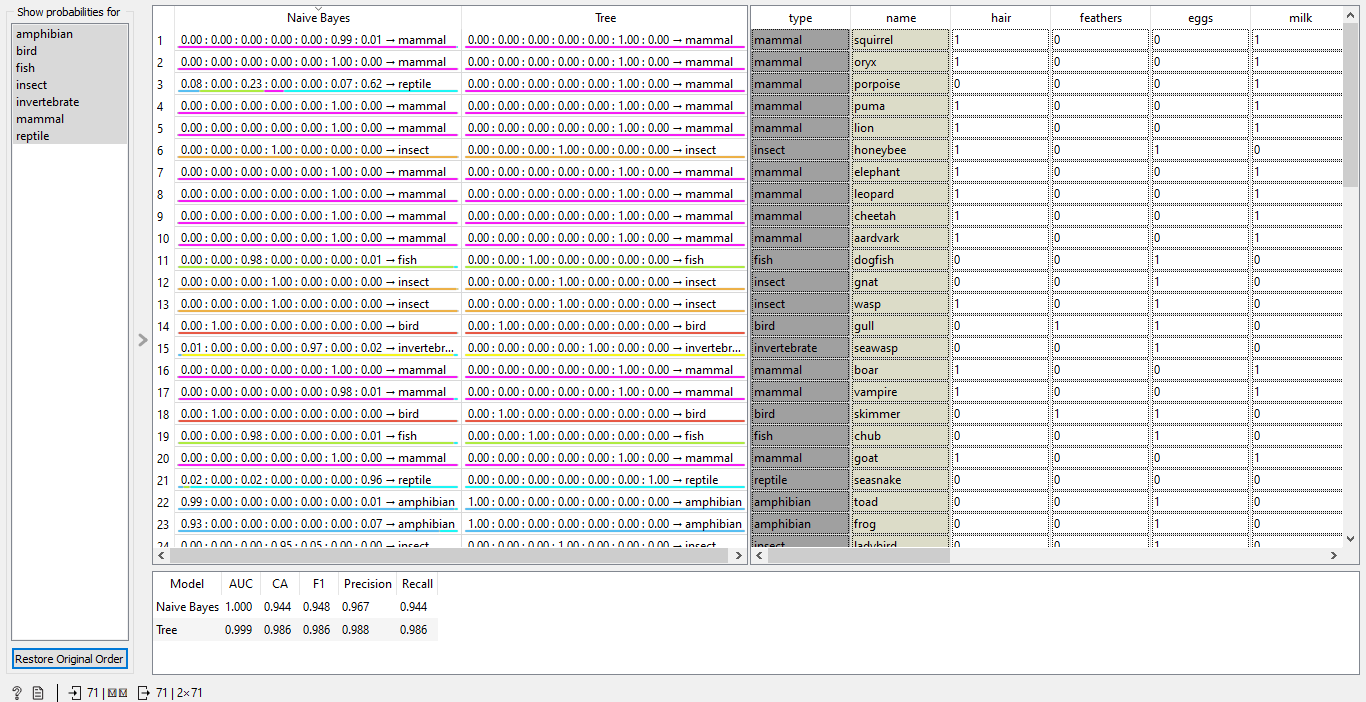




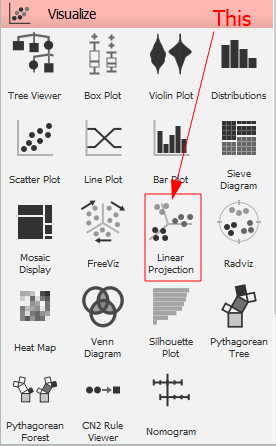
* Connect the Naive Bayes widget and Tree widget to the Predictions widget to perform predictions.

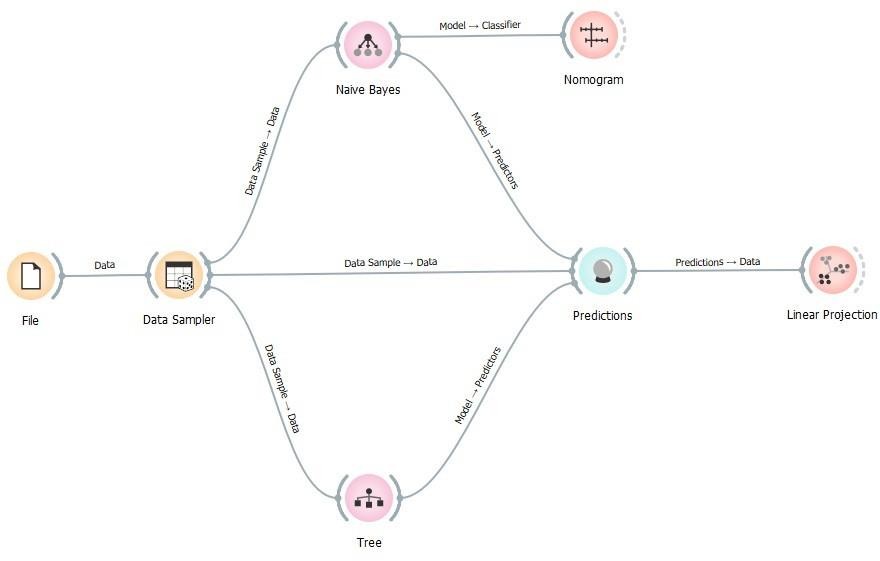


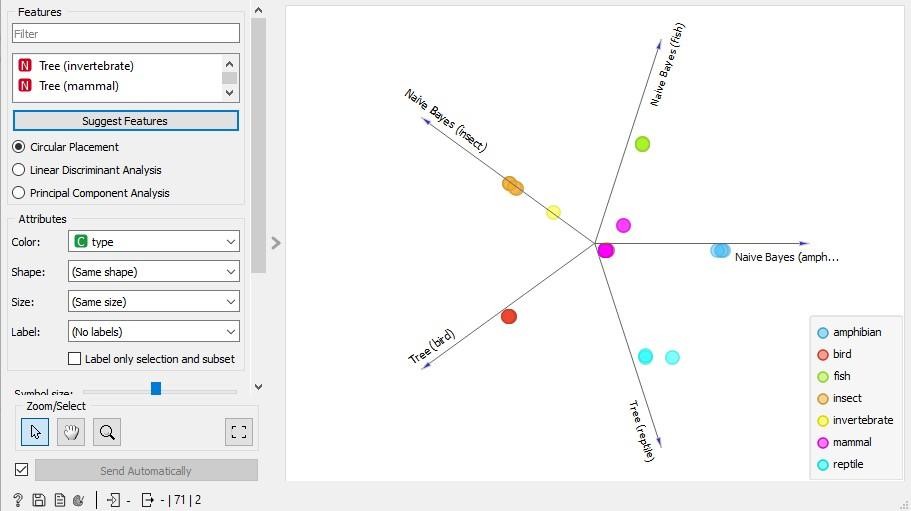
* Double click the Predictions widget to see the predictions.



* We will use the Linear Projection widget to visualize the data. Drag and drop this widget from the Visualize section.





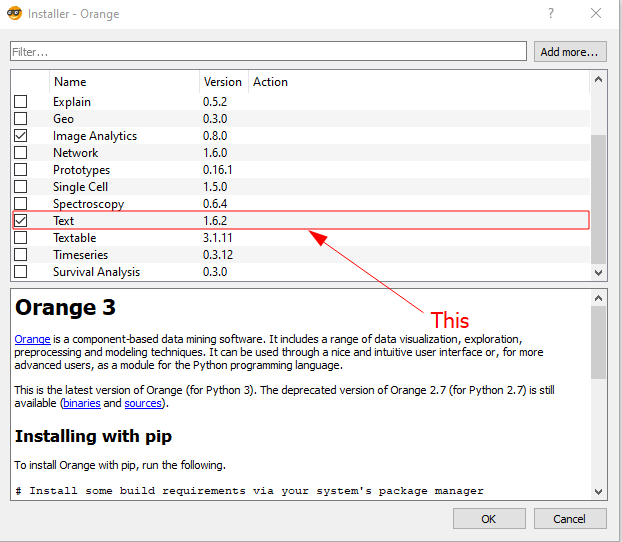


# Practical 2

**Aim:** Text classification using Orange.

## Steps:

* Before starting, we need to install the Text add-on. Install it by navigating to Options > Add ons…. You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.

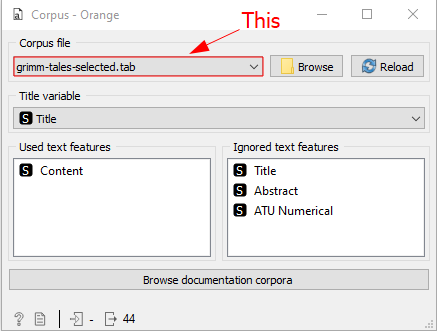


* Drag and drop a Corpus widget to the workspace. It can be found in the newly added Text Mining section.



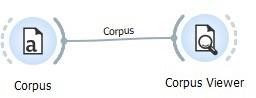


* Double click the Corpus widget and select the grimm-tales- selected.tab corpus file.

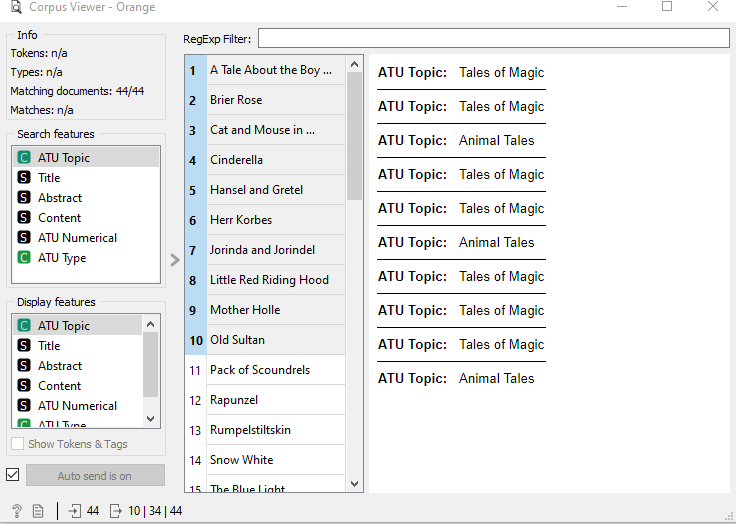


* Drag and drop a Corpus Viewer widget and connect it to the Corpus widget. The Corpus Viewer widget can be found in the Text Mining section.



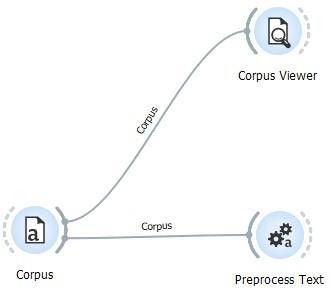


* Double click the Corpus Viewer widget to visualize the corpus in a tabular format. Select the first 10 entries to use as training set.

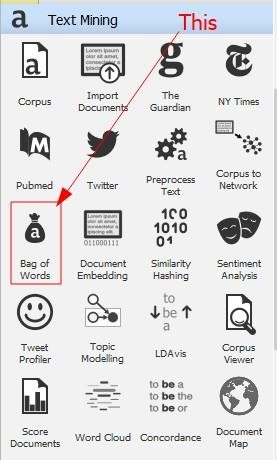


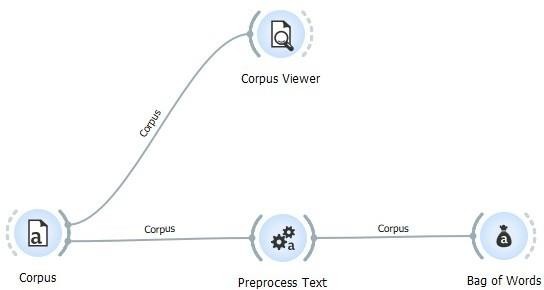
* Add a Preprocess Text widget to the workspace. This widget can also be found in the Text Miningsection.



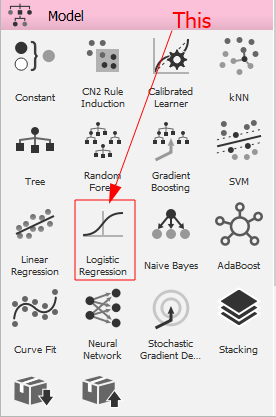


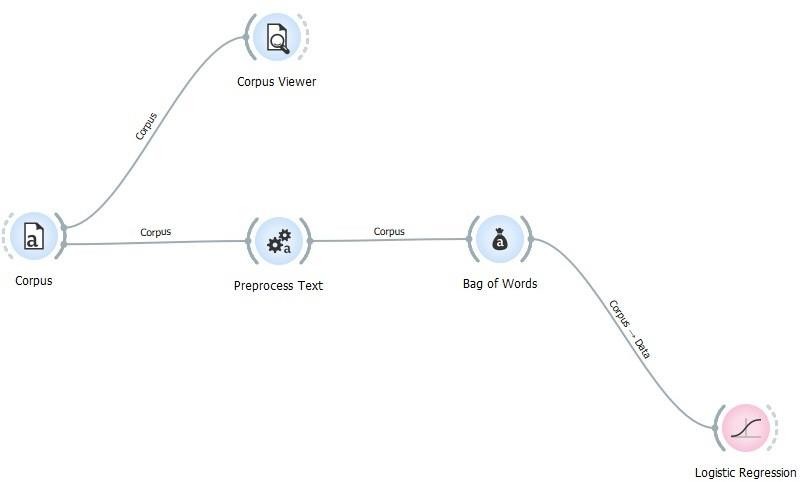
* Add a Bag Of Words widget to the workspace and connect it to the Preprocess Text widget. This widget is also found in the Text Miningsection.



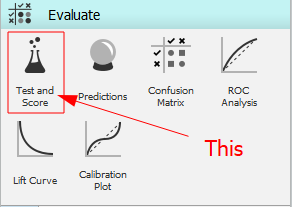


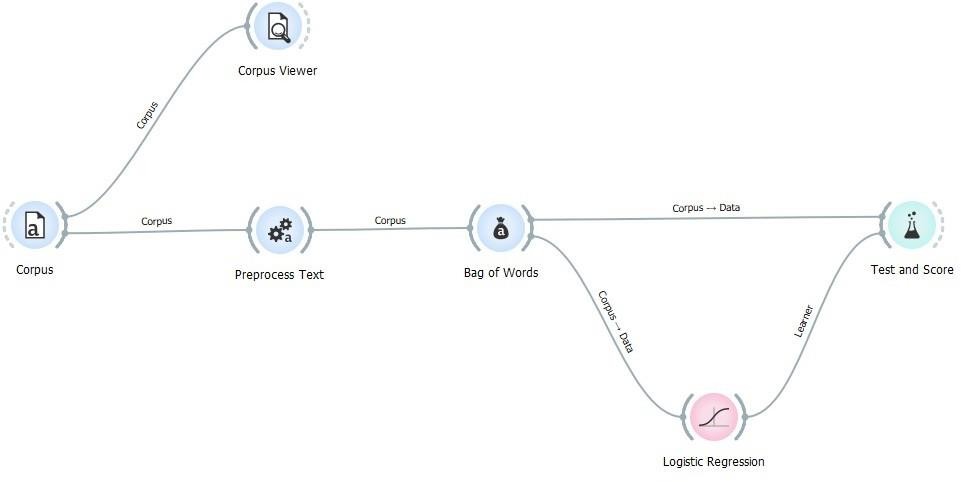
* We will use Logistic Regression to model our data. Drag and drop this widget from the Model section onto the workspace.



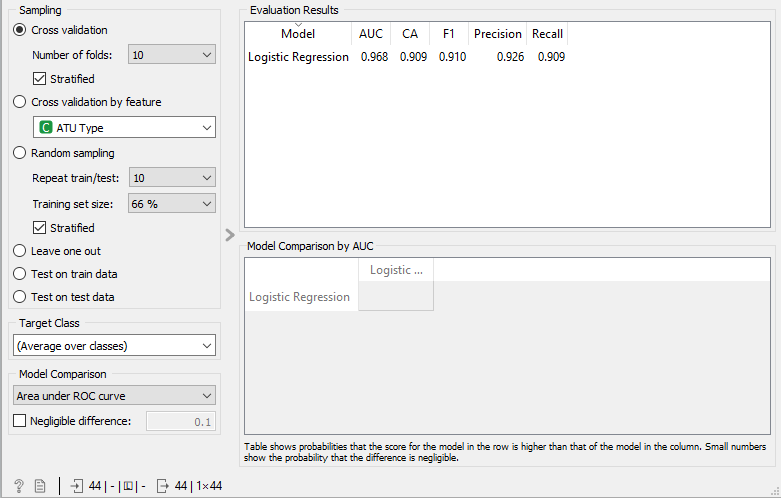


* We will use the Test and Score widget to check our model. Drag and drop this widget from the Evaluatesection onto the workspace and connect it to the Logistic Regression model and Bag of Words widget.

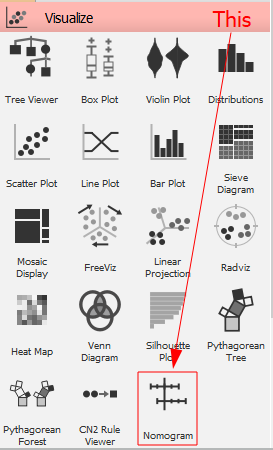


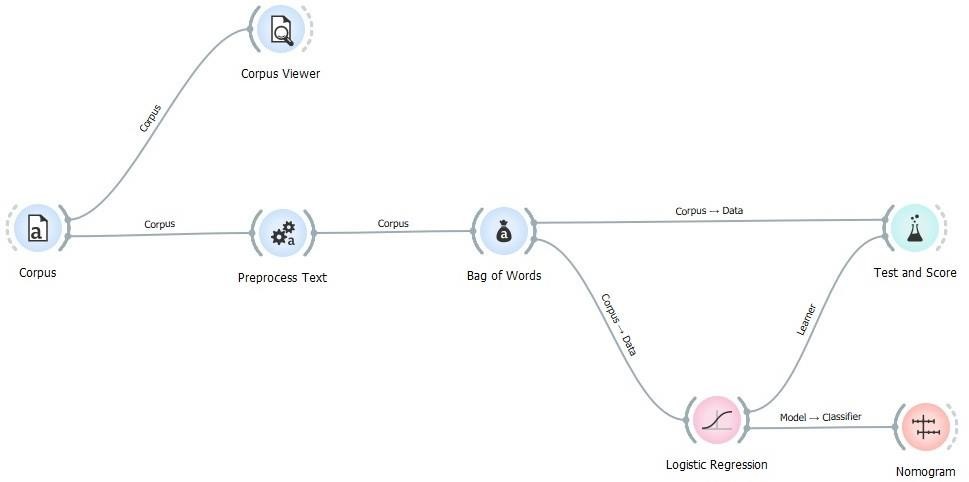


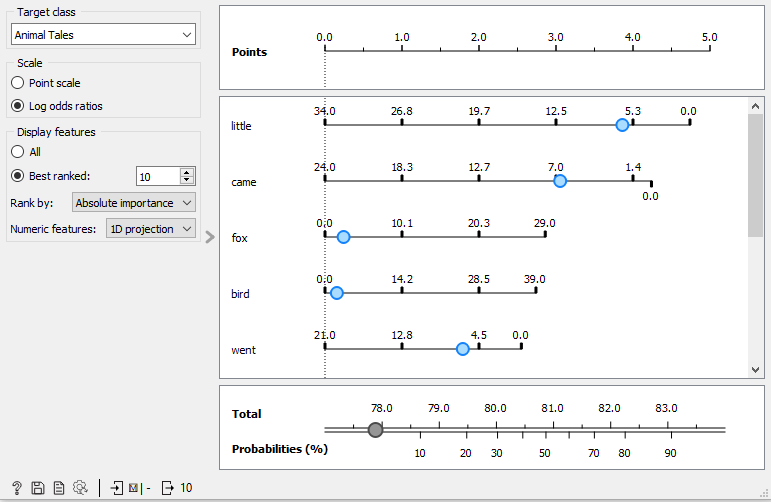
* Double-click the Test and Score widget after it finishes processing. It will provide you with data such as the accuracy of the model etc.



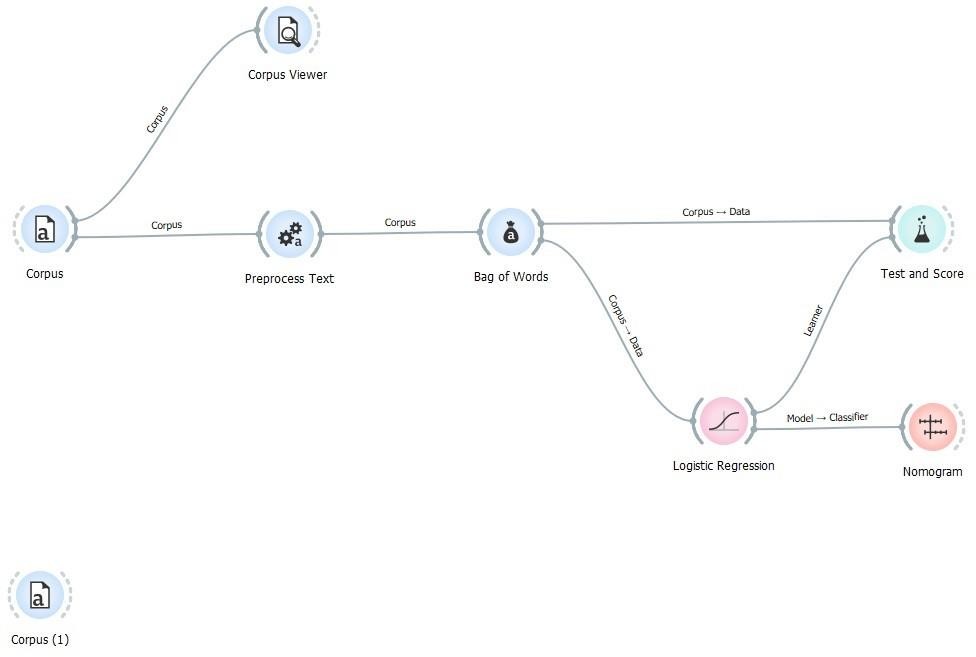
* We will use a Nomogram to visualize our model. Drag and drop it from the Visualize section onto the workspace and connect it to the Linear Regressionmodel.

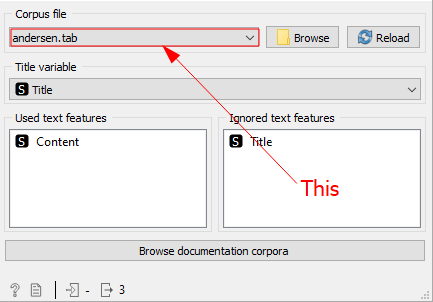




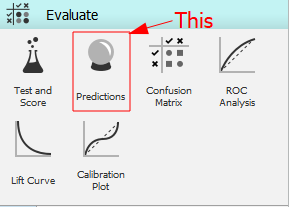


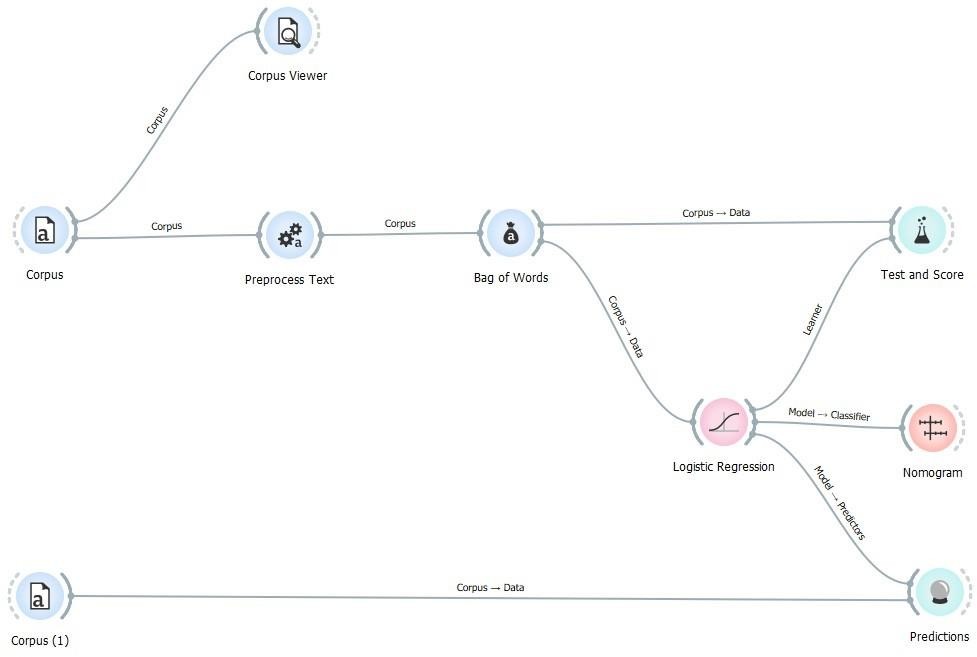
* To see whether the model works as intended, we create a new Corpuswidget and set the file to andersen.tab.

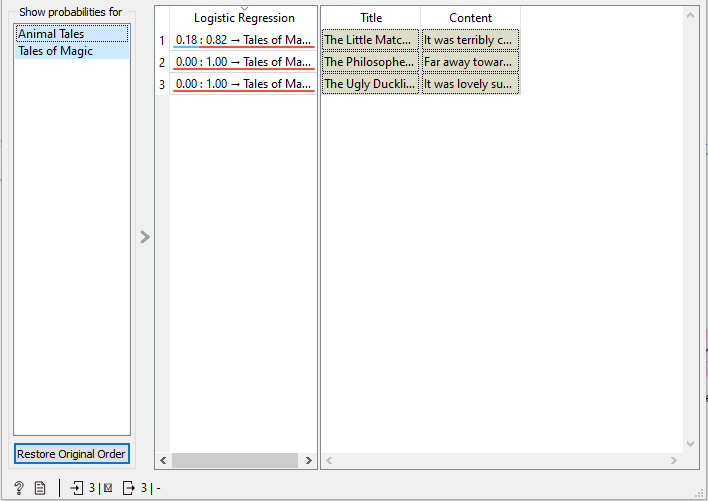




* Now drag and drop a Predictions widget from the Evaluate section onto the workspace. Connect it to the model as well as the new Corpuswidget to visualise the results.





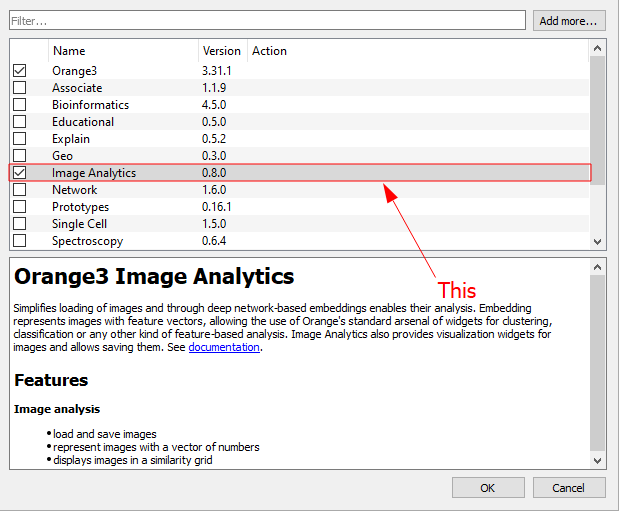


# Practical 3

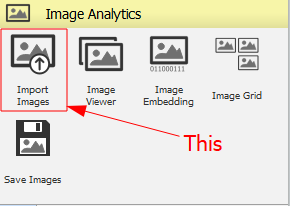
**Aim:** Image classification using orange.

## Steps:

* Before starting, we need to install the ImageProcessing add-on. Install it by navigating to Options > Add ons…. You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.

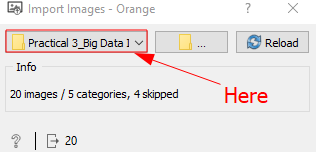


* Add a Import Images widget from the newly added Image Analytics section to the workspace.

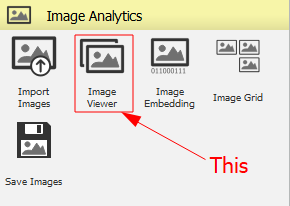


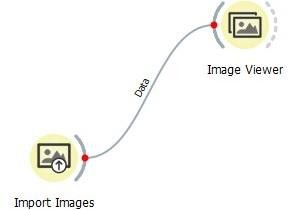


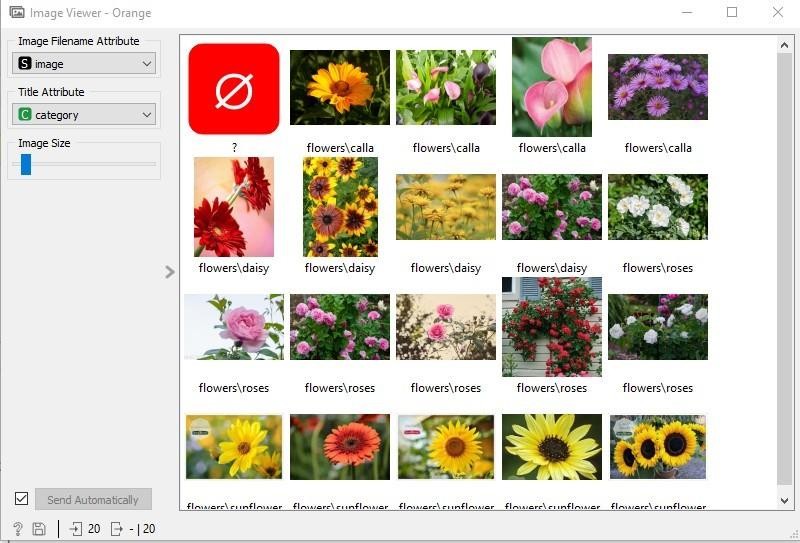
* Load the directory containing the images in the newly added Import Images widget by double clicking the widget.



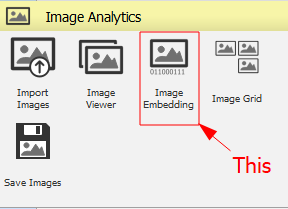
* We can view these images using the Image Viewerwidget found in the Image Analytics section. Drag and drop this widget onto the workspace and connect it to the Import Imageswidget.

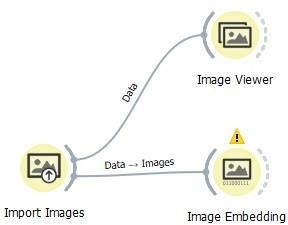




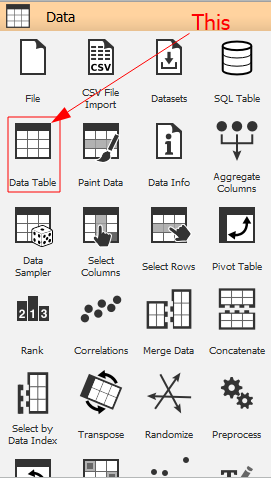


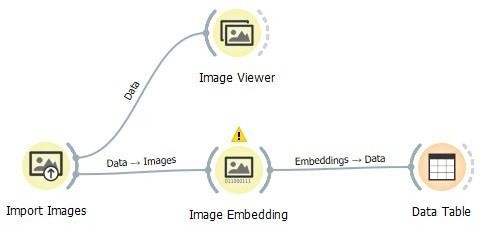
* As our models can only process numbers, we need to convert the images into numerical data. This is where Image Embedding comes into play. Drag and drop a Image Embeddingwidget from the Image Analytics section onto the workspace and connect it to the Import Imageswidget.

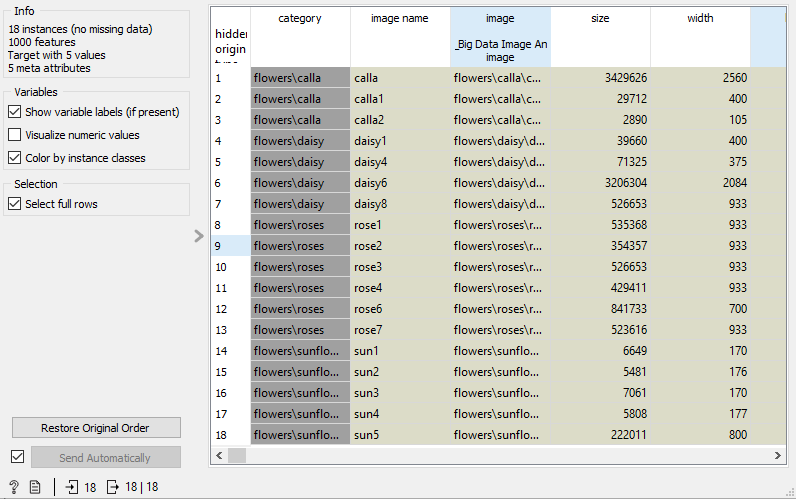




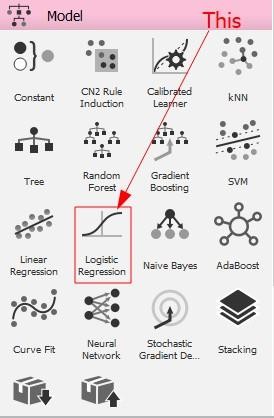
* We use a Data Table to visualise the tabular data generated by the Image Embeddingwidget. Drag and drop a Data Tablewidget from the Data section onto the workspace and connect it to the Image Embedding widget.

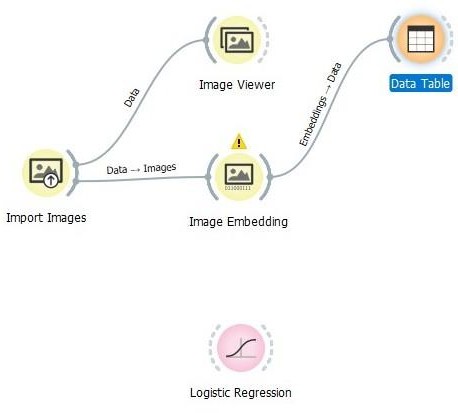




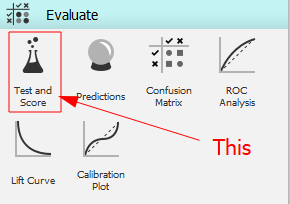


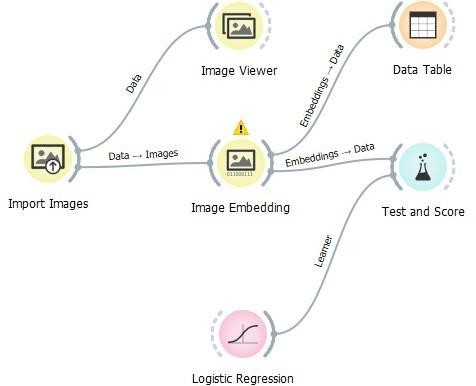
* We will use Logistic Regressionas our model. Drag and drop a Logistic Regression widget from the Model section onto the workspace.

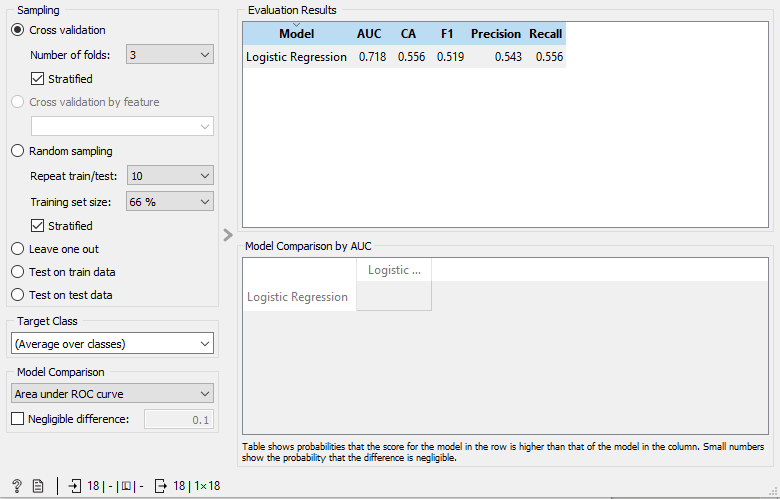




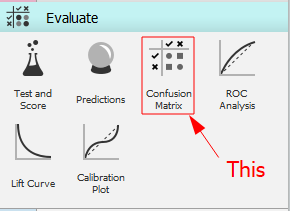
* Drag and drop a Test and Scorewidget from the Evaluatesection onto the workspace. This widget will allow us to verify our model. Connect this widget to the Image Embedding and Logistic Regressionwidgets.

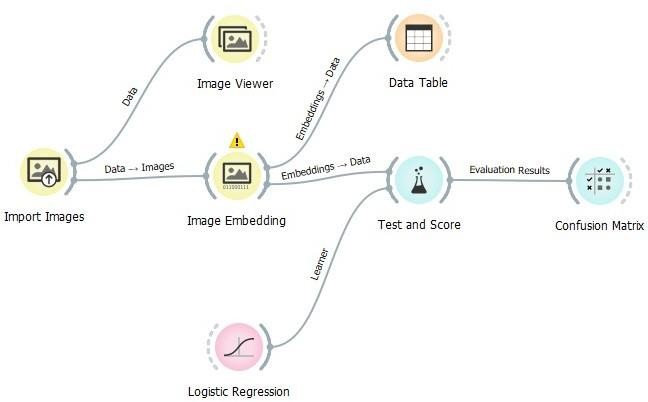


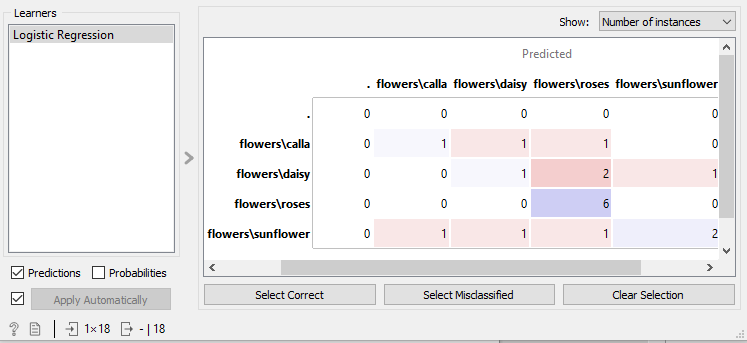




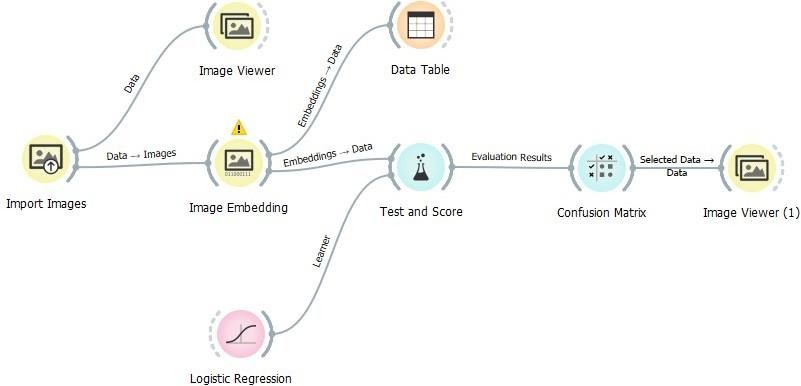
* We now drag and drop a Confusion Matrix widget from the Evaluate section onto the workspace and connect it to the Testand Score widget.

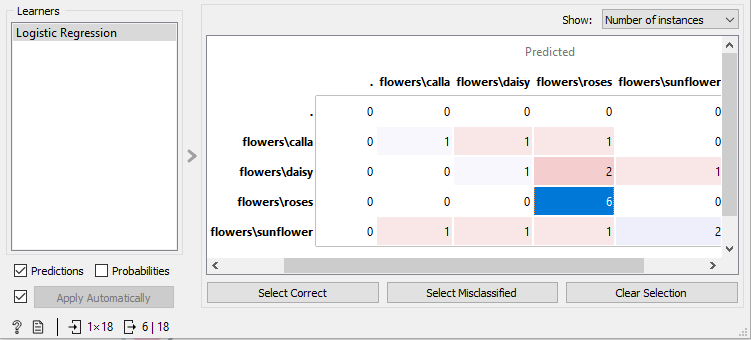


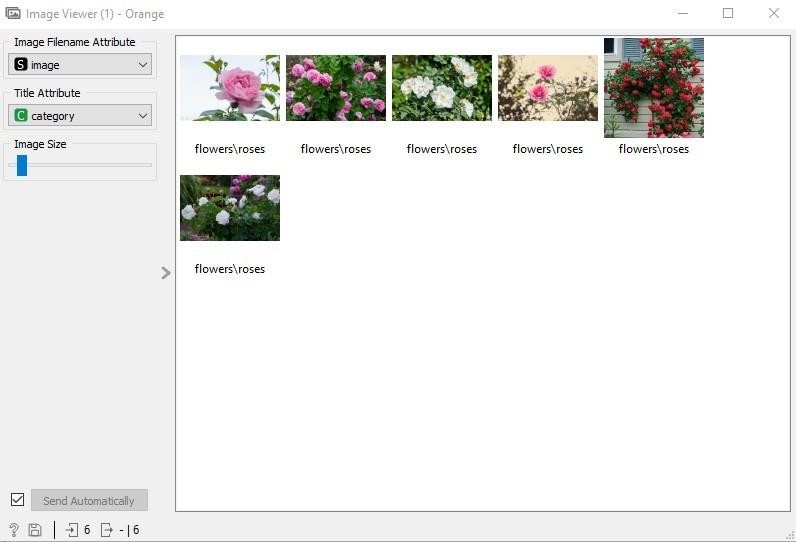




* Drag and drop another Image Viewerwidget to view the selected cell(s) from the Confusion Matrix.





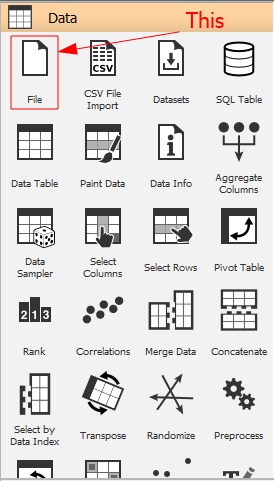


# Practical 4

**Aim:** Hierarchical clustering using orange.

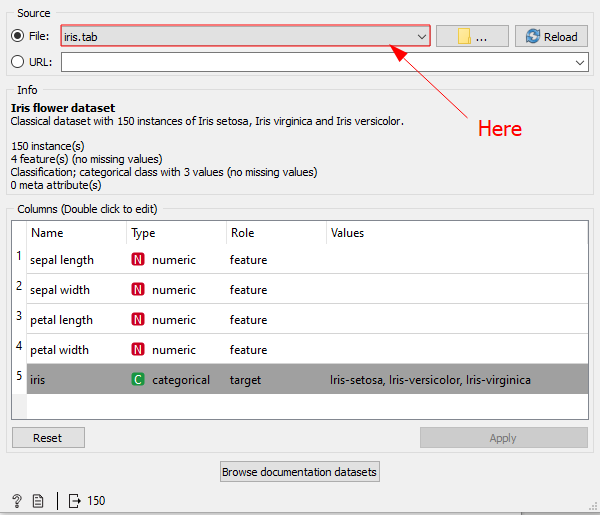
## Steps:

* Drag and drop a Filewidget from the Datasection onto the workspace.

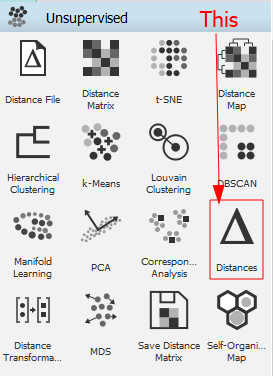


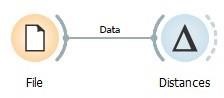


* Double click the Filewidget and set the name to iris.tab.

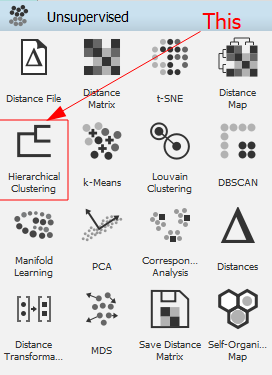


* Drag and drop the Distanceswidget from the Unsupervised section onto the workspace and connect it to the File widget.



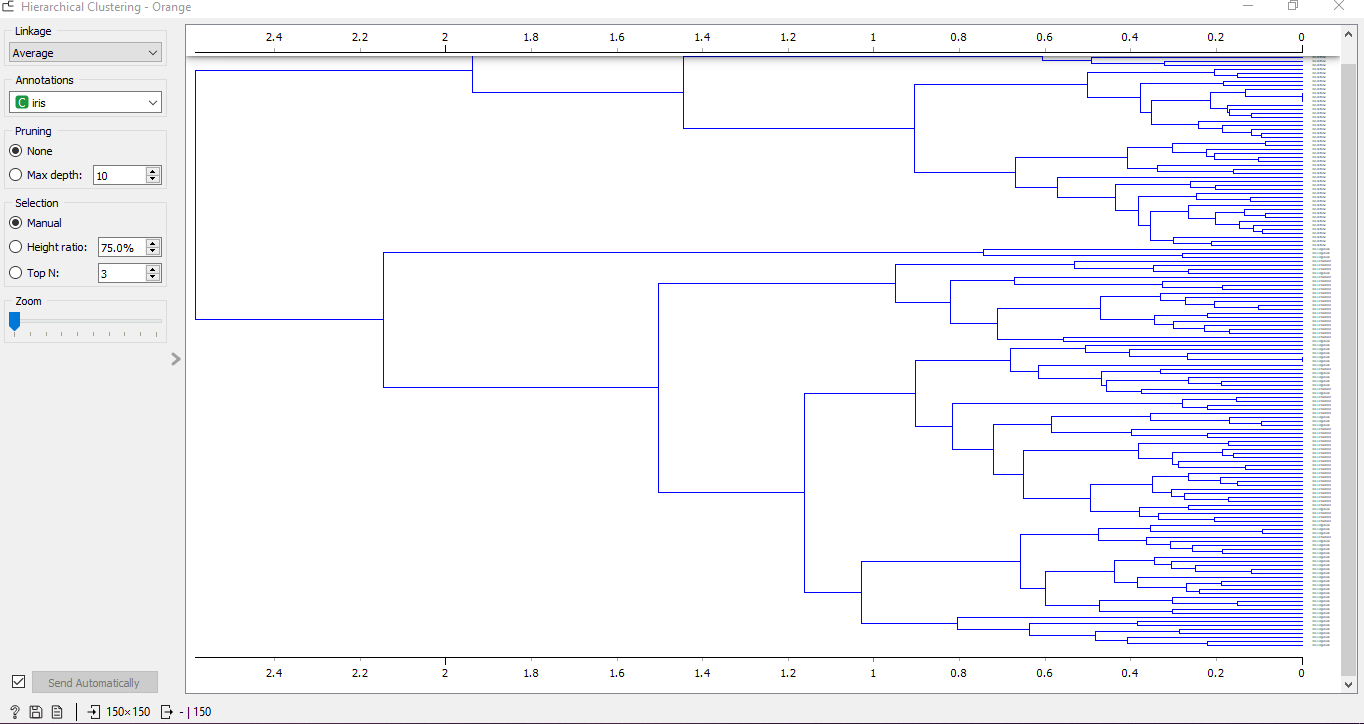


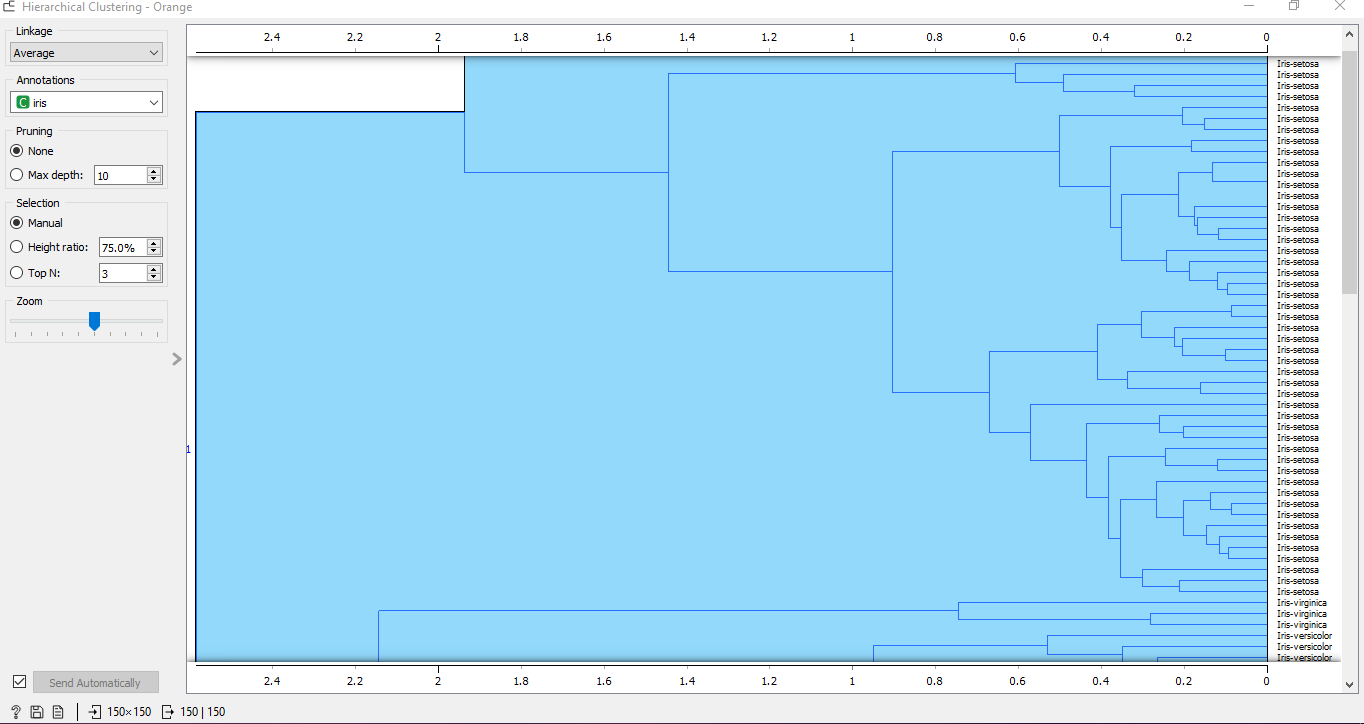
* Drag and drop a Hierarchical Clusteringwidget from the Unsupervised section to the workspace and connect it to the Distanceswidget.



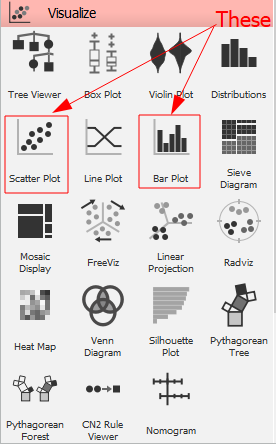


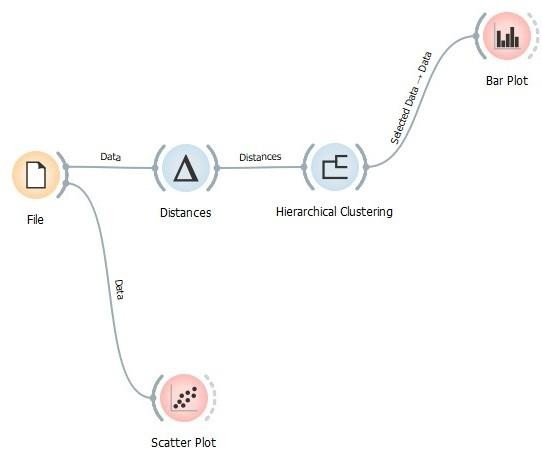
* Double click the Hierarchical Clustering widget to view the dendogram. Select any sub-cluster as per convenience.

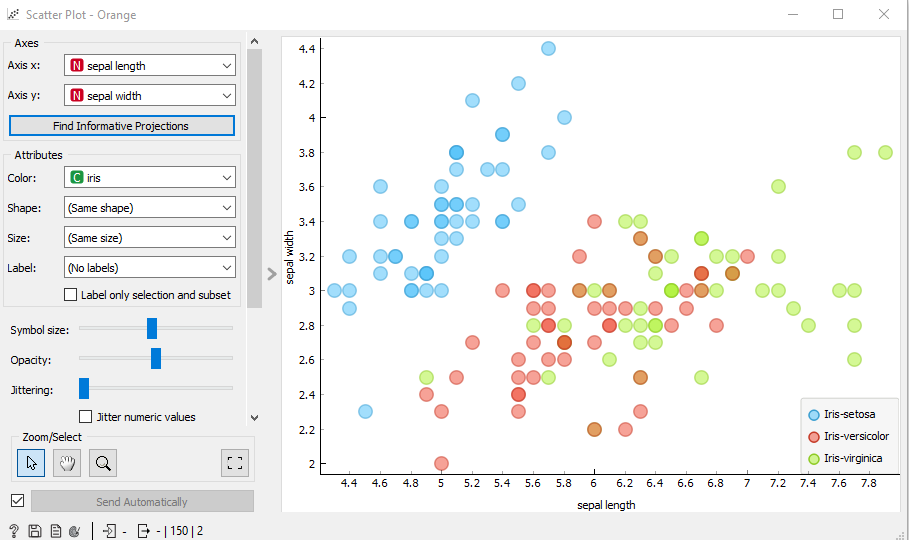


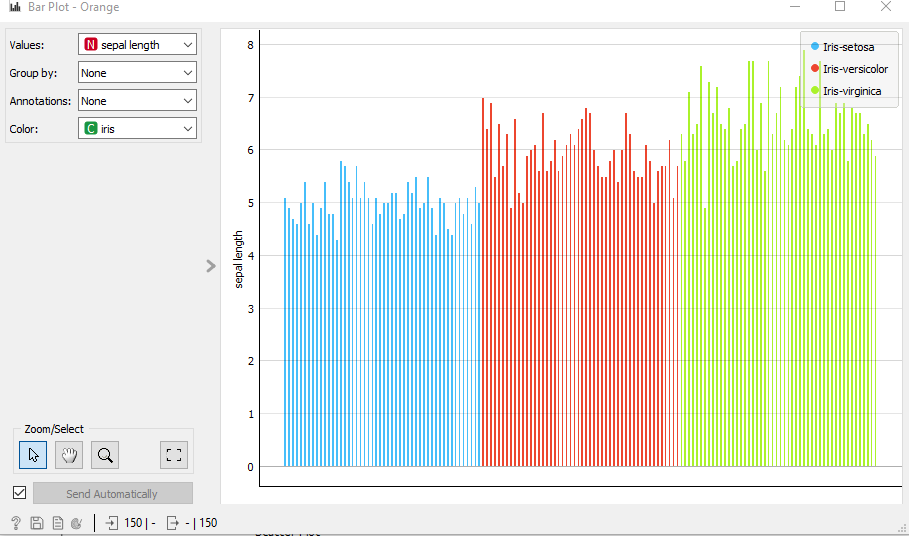


* Choose any visualization method of your choice. This example assumes a Bar Plotfor the sub-cluster and a Scatter Plotfor the source dataset. You can find both of these widgets in the Visualize section. Connect the Bar Plot to the Hierarchical Clustering widget and Scatter Plotto the Filewidget.







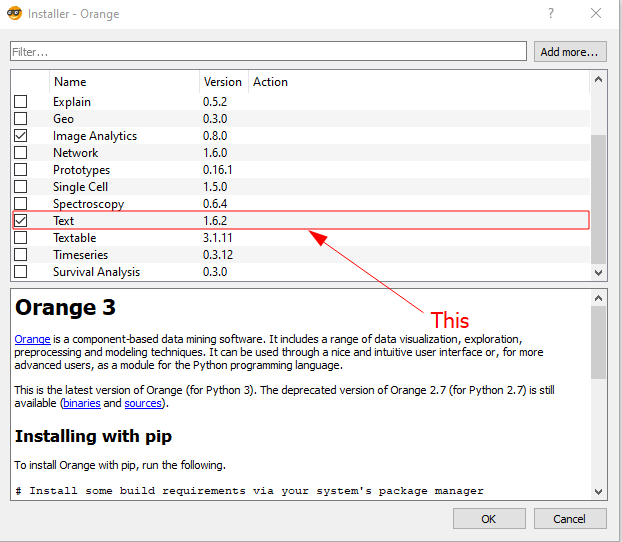


# Practical 5

**Aim:** Text Preprocessing using orange.

## Steps:

* Before starting, we need to install the Text add-on. Install it by navigating to Options > Add ons…. You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.

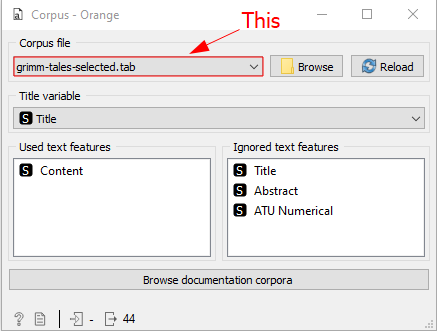


* Drag and drop a Corpuswidget to the workspace. It can be found in the newly added Text Mining section.



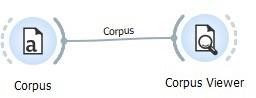


* Double click the Corpus widget and select the grimm-tales- selected.tab corpus file.

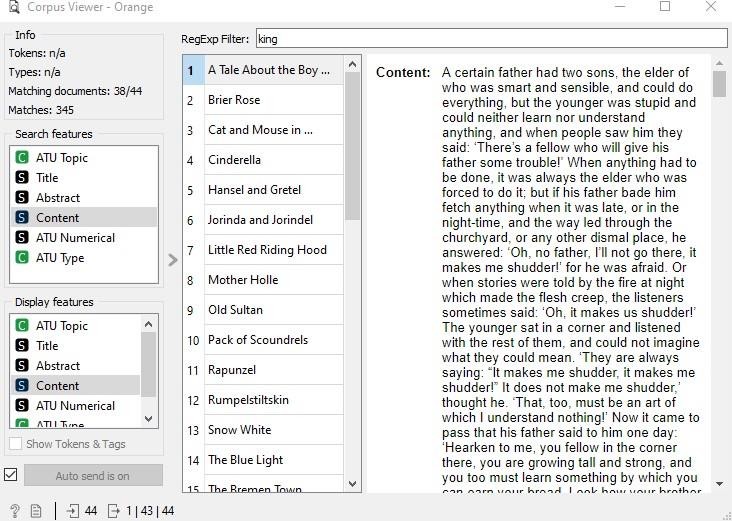


* Drag and drop a Corpus Viewer widget and connect it to the Corpus widget. The Corpus Viewer widget can be found in the Text Mining section.

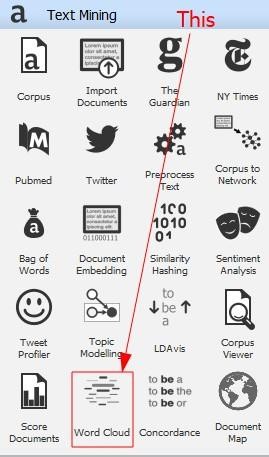


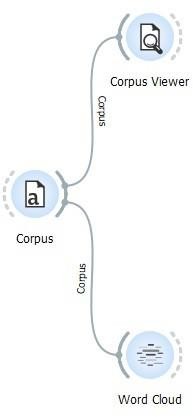


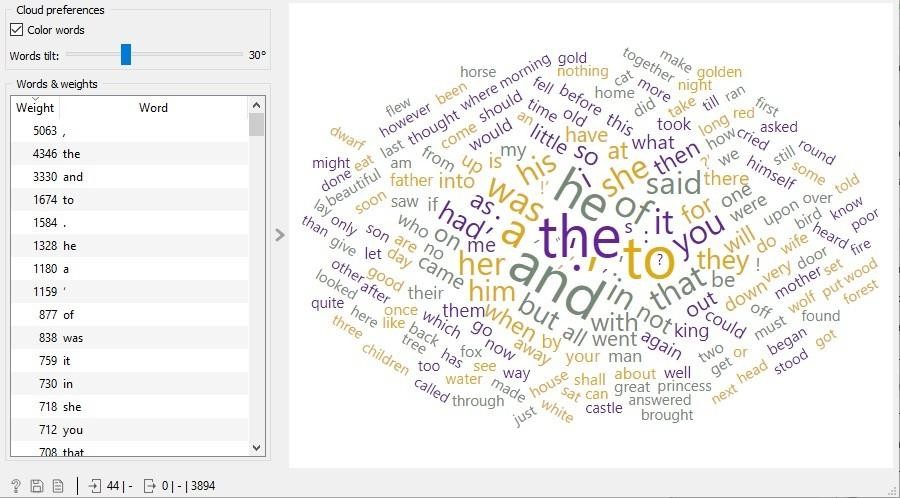
* Double-clicking the Corpus Viewer allows us to peek through the corpus and also allows us to filter text.



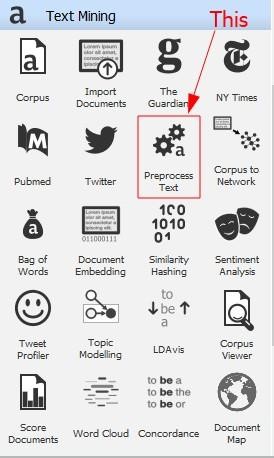
* Another method of visualizing data is though Word Clouds. Drag and drop a Word Cloud widget from the Text Mining section and connect it to the Corpuswidget.

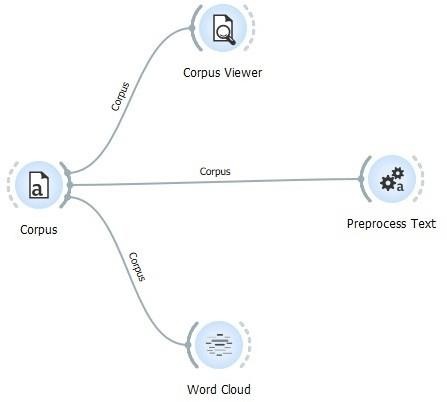




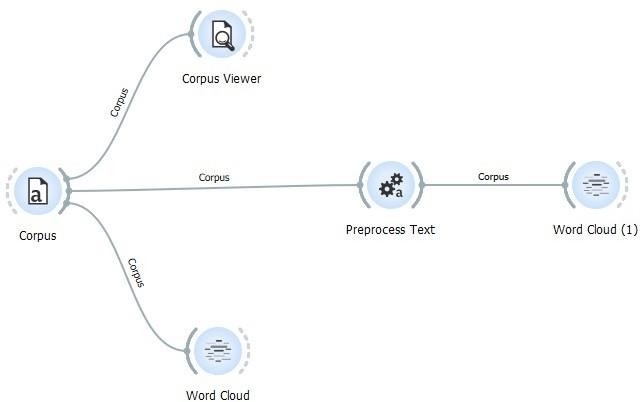


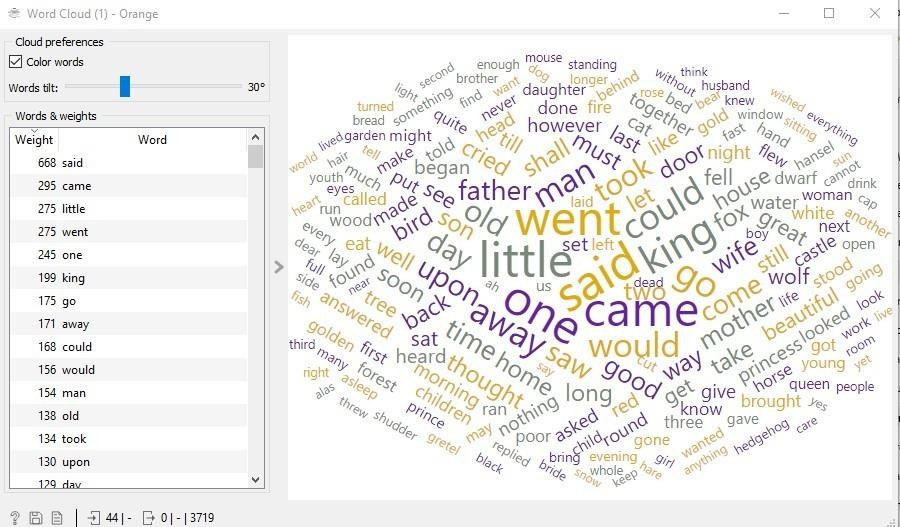
* We see that a lot of punctuation marks and uninformative words have made their way into the Word Cloud. In order to eliminate this, we have to use the Preprocess Text widget. Drag and drop a Preprocess Text widget from the Text Mining section onto the workspace and connect it to the Corpus widget.





* We now add another Word Cloud widget to see our updated cloud.





# Practical 6

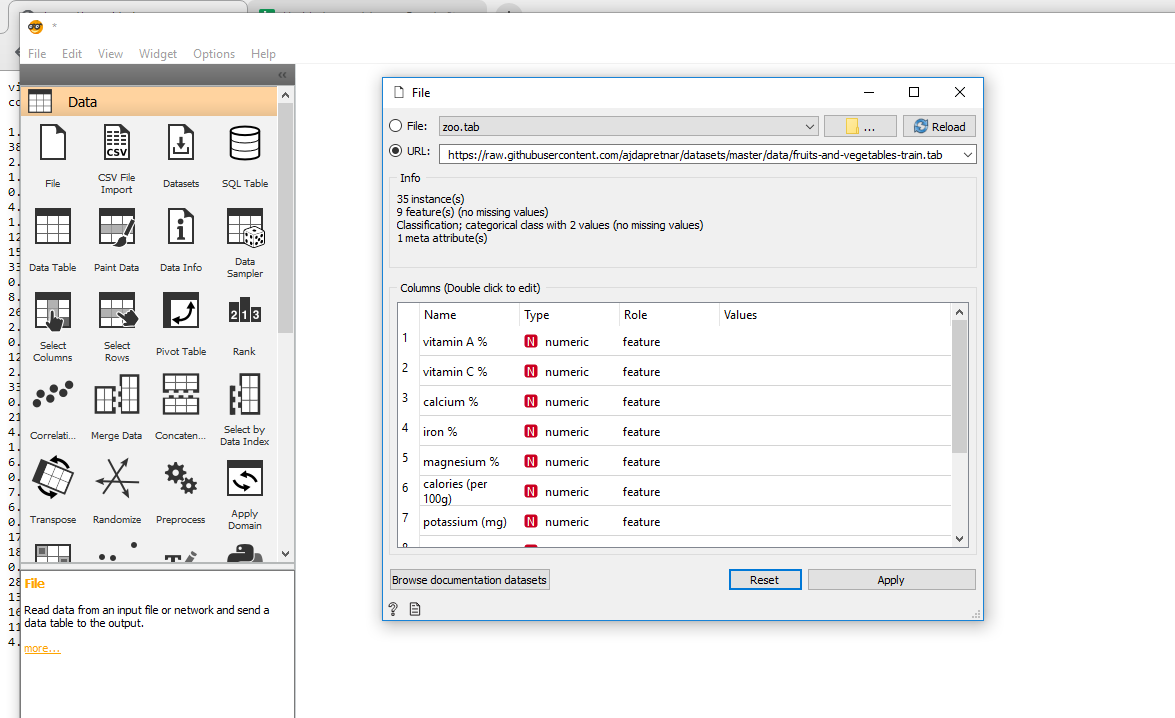
**Aim**: To predict whether the given dataset is of fruit or vegetable using orange.

## Description:

This dataset contain numerous feature which diffrenciate whether the given dataset is of vegetable or fruit. The target variable is classification i.e Is it fruit or vegetable.

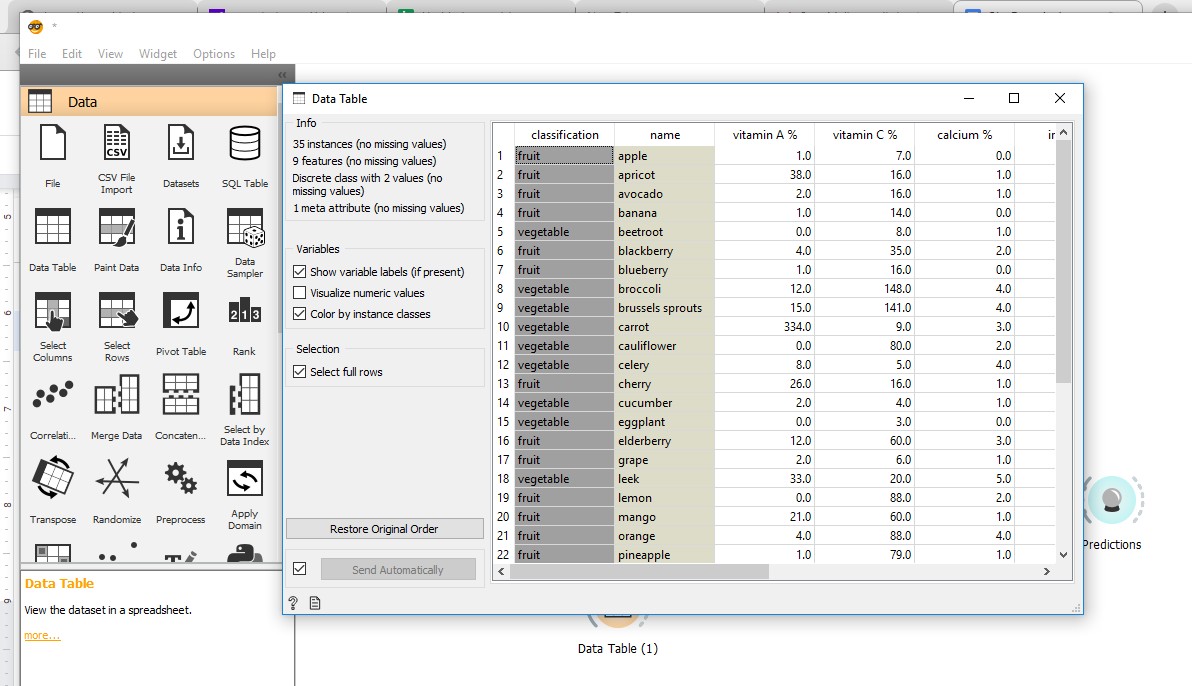
## Data selection:

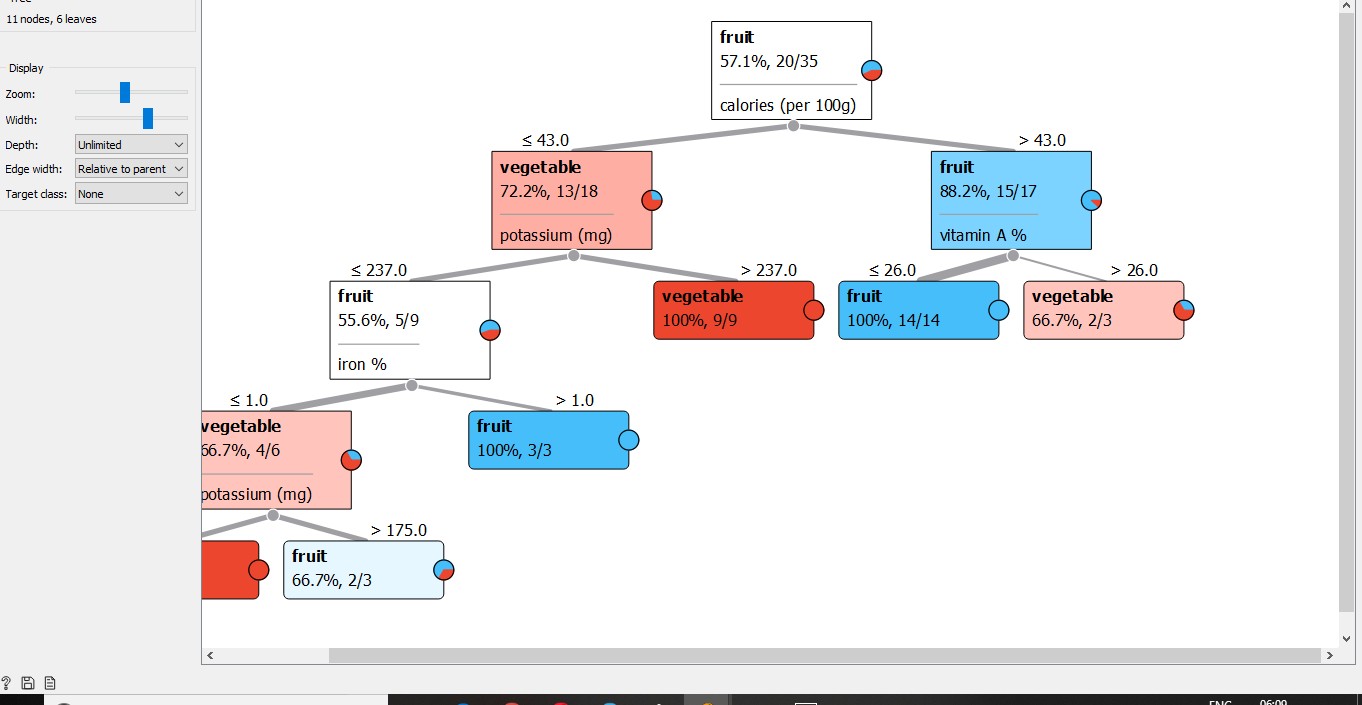
We have given the url of the dataset which contain 35 instances and 9 features of fruit and vegetables.



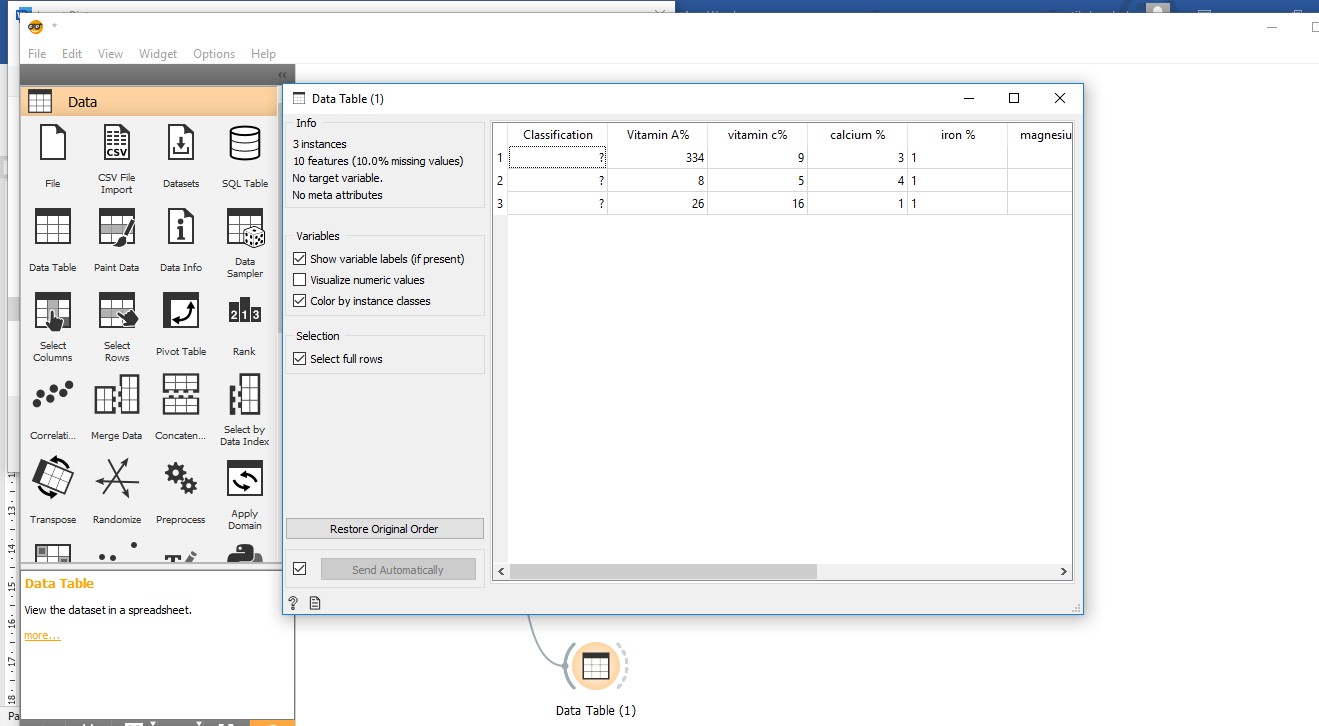
Data visualization:

We visualize the data by using data table and classification tree.

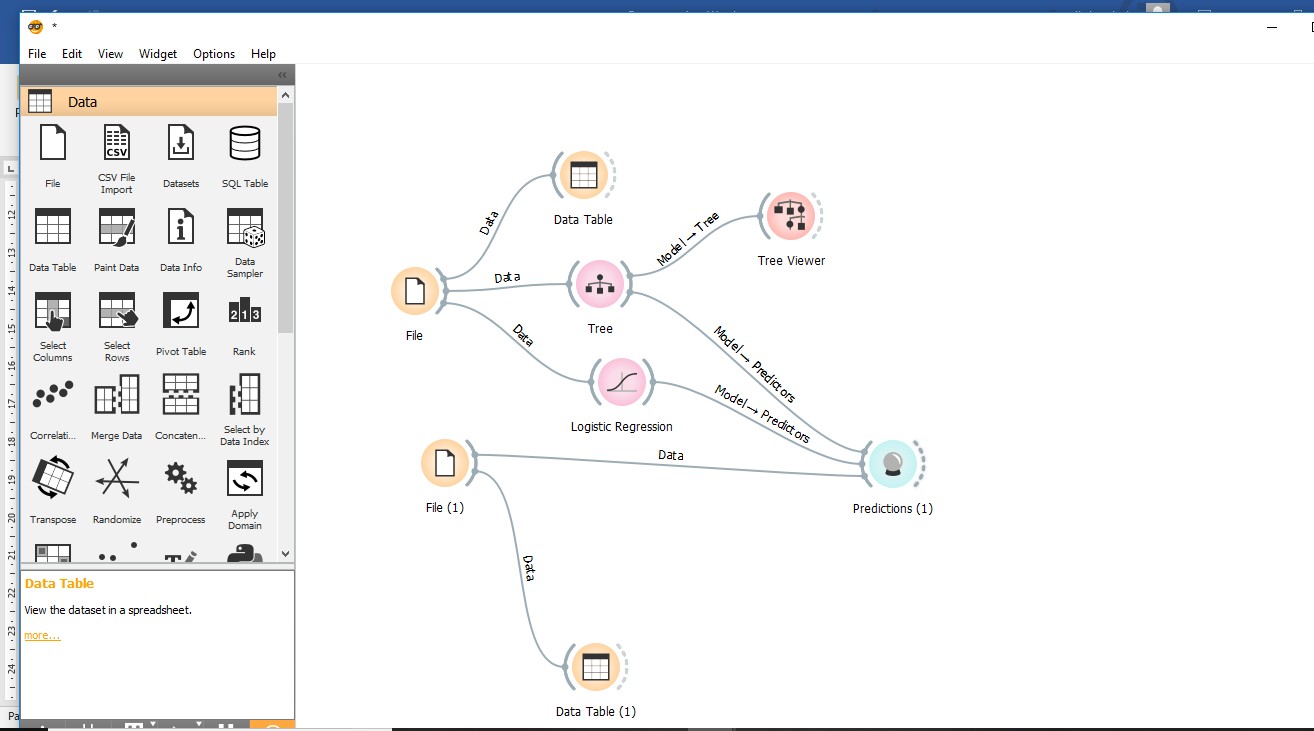


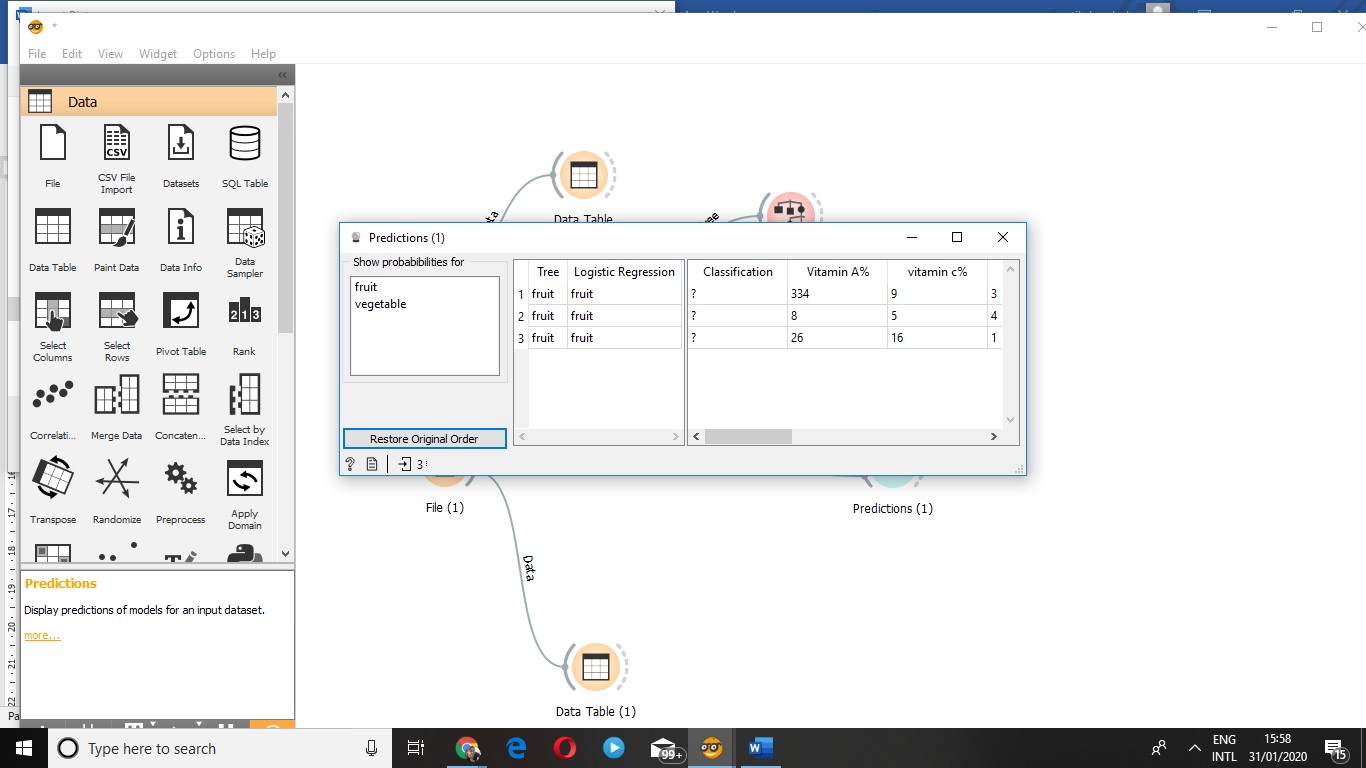


Again ,You have to create the the dataset of plants which you need to predict and visualize it in data table.



By using prediction widget you need to predict whether the plant is vegetable or fruit .





The given prediction is again checked by logistic regression.

# Practical 7

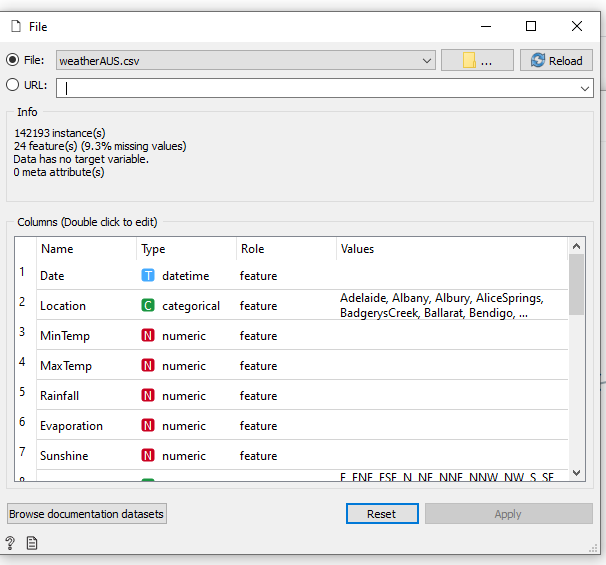
**Aim**: Predict whether using orange.

**Description**: This dataset contains daily weather observations from numerous Australian weather stations.

The target variable Rain Tomorrow means: Did it rain the next day? Yes or No.

## Data Selection:

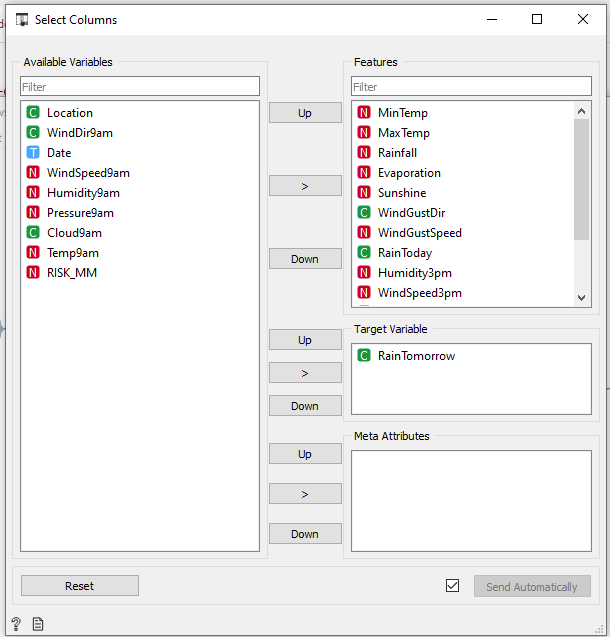
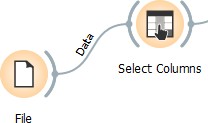
We have downloaded the dataset ( [weather dataset](https://www.kaggle.com/jsphyg/weather-dataset-rattle-package) ) of Australian weather stations which contains about 10 years of daily weather observations from numerous Australian weather stations.



## Data Cleaning:

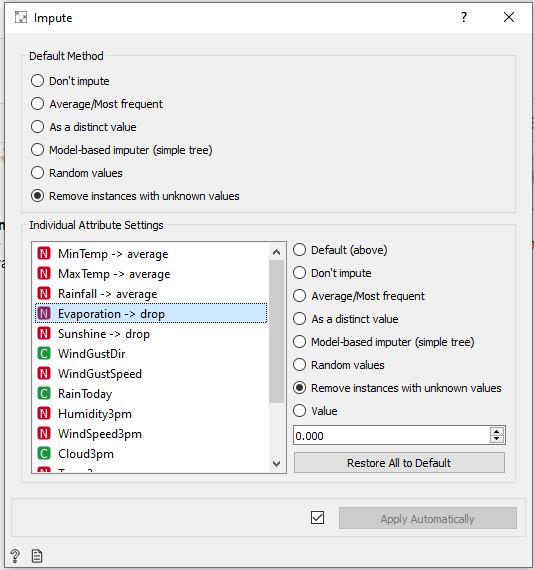
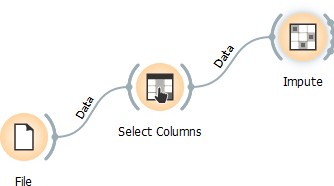
In data cleaning, we will select the required column and remove the rows with missing values as shown below:

## Column Selection



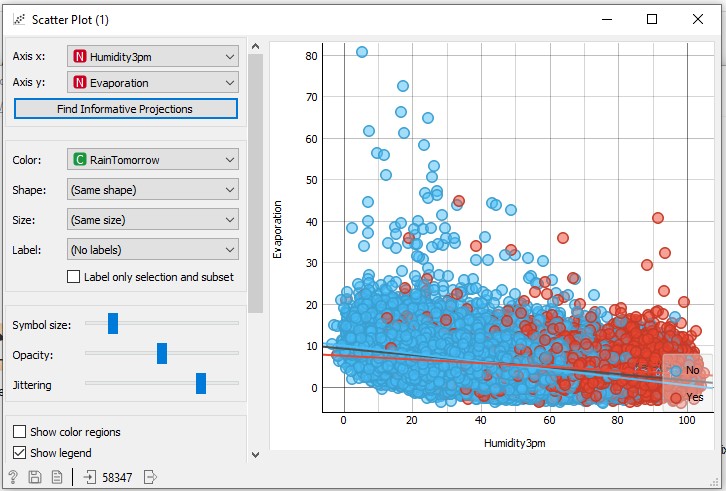
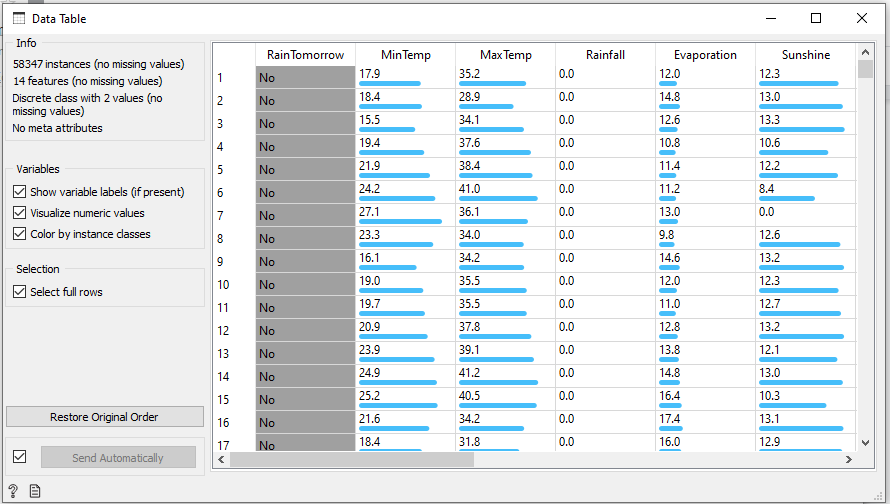
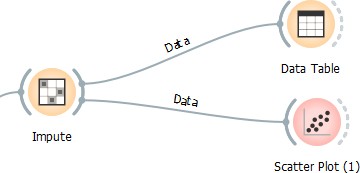
**Removing rows with missing values**

We will use the impute function to remove rows with missing values for columns “Sunshine” and “Evaporation”



## Data Visualization

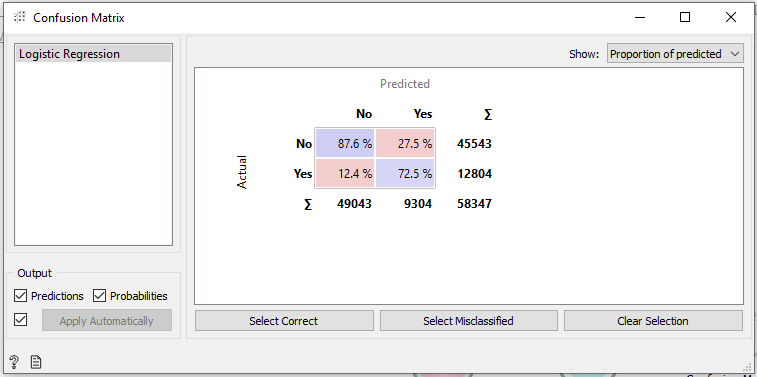
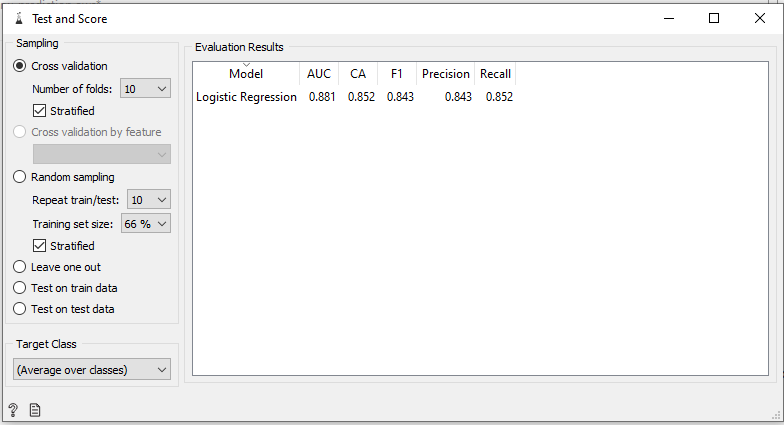
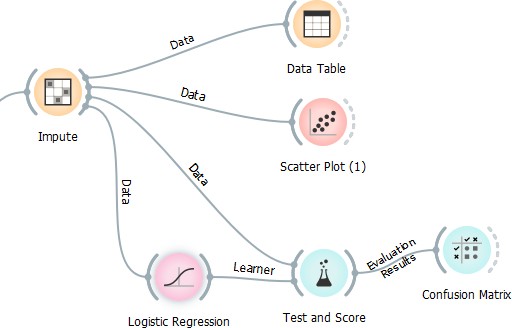
In this step, we see the reflected data selection in data table. Then, we use scatter plot to visualize the data for better understanding of our dataset.



In the above scatter plot we see, as the humidity level increases and evaporation rates are low the chances of rainfall increases (red dots)

## Data Modeling

In this step, we use **Logistic regression** to model our data. Then, we use “test and score” function to evaluate the performance of our model. Finally, we use “confusion matrix” to understand the prediction.



# Practical 8

**Aim:** Write a java program to Calculate the Alon Matias Szegedy Algorithm for given stream.

## Input:

**import java.io.\*; import java.util.\*;**

**public class AMSA {**

**public static int findCharCount(String stream, char XE, int random, int length) { int countOccurence = 0;**

**for(int i = random; i < length; i++)**

**{**

**if(stream.charAt(i) == XE)**

**{**

**countOccurence++;**

**}**

**}**

**return countOccurence;**

**}**

**public static int estimateValue(int XV1, int n) { int expValue;**

**expValue = n \* (2 \* XV1 - 1); return expValue;**

**}**

**public static void main(String[] args) { int n = 15;**

**String stream = "abcbdacdabdcaab";**

**int randomValues[] = {3, 8, 13}; char[] XE = new char[3];**

**int[] XV = new int[3];**

**int[] expValue = new int[3]; int apprSecondMomentValue;**

**for(int i = 0; i < randomValues.length; i++)**

**{**

**XE[i] = stream.charAt(randomValues[i] - 1);**

**}**

**for(int i = 0; i < randomValues.length; i++)**

**{**

**XV[i] = findCharCount(stream, XE[i], randomValues[i] - 1, n);**

**}**

**System.out.println(XE[0] + "=" + XV[0] + " " + XE[1] + "=" + XV[1] + " " + XE[2] + "=" + XV[2]);**

**for(int i = 0; i < randomValues.length; i++)**

**{**

**expValue[i] = estimateValue(XV[i], n);**

**}**

**for(int i = 0; i < randomValues.length; i++)**

**{**

**System.out.println("Expected value for "+XE[i]+" is :: "+expValue[i]);**

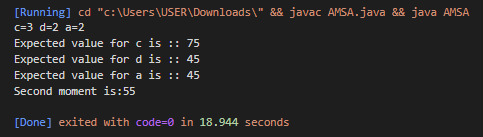
**}**

**apprSecondMomentValue = Arrays.stream(expValue).sum() / 3; System.out.println("Second moment is:" + apprSecondMomentValue);**

**}**

**}**

**Output:**



# Practical 9

**Aim:** Write a Program to Construct different types of K-shingles for a given document

## Source code:

require("tm")

kshingle<-function(){

k<- as.integer(readline("Enter a value for k - 1")) u1<- readLines("BIBD\_1.txt")

shingle<-i<-0 while(i<nchar(u1)-k+1)

{

shingle[i]<-substr(u1,i,i+k) print(shingle[i])

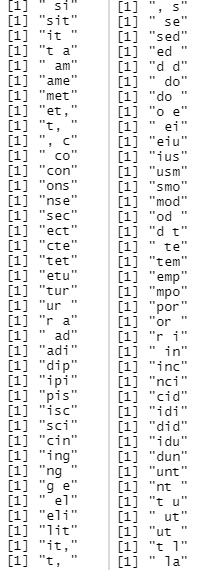
i<-i+1

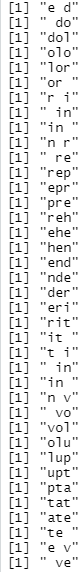
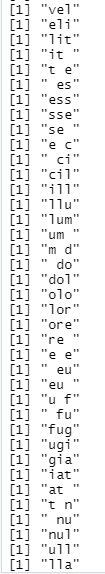
}

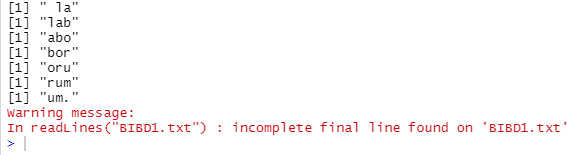
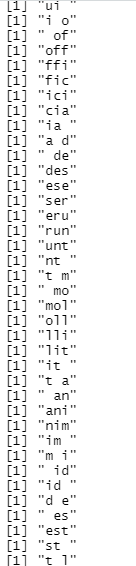
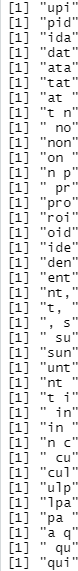
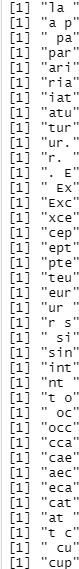
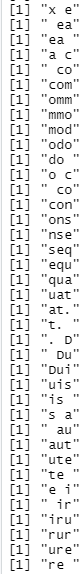
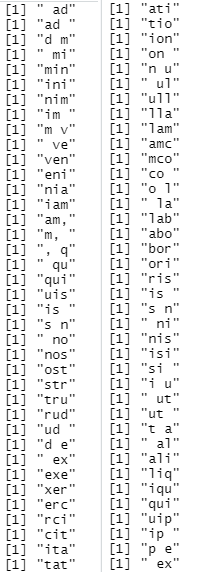
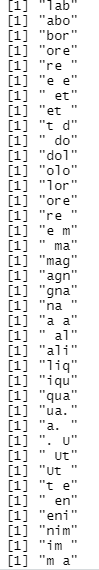
}

if(interactive())kshingle()

## Output:





# Practical 10

**Aim:** Write a program for measuring similarity among documents and detecting passages which have been reused.

## Installation of required packages before executing program:-

install.packages("tm") require("tm") install.packages("ggplot2") install.packages("textreuse") install.packages("devtools")

**Source Code a**:- require("tm")

my.corpus<-Corpus(DirSource("files"))

my.corpus<-tm\_map(my.corpus,removeWords,stopWords("english")) my.tdm<-TermDocumentMatrix(my.corpus)

#inspect(my.tdm) my.dtm<-

DocumentTermMatrix(my.corpus,control=list(weighting=weightTfldf,stopw ords=TRUE))

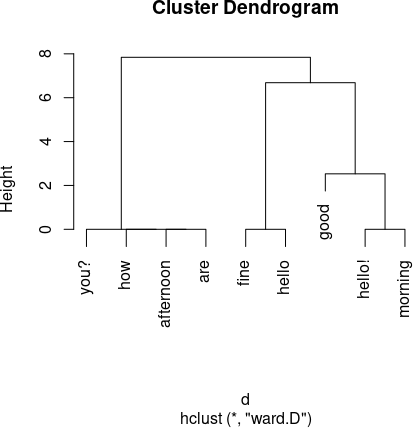
#inspect(my.dtm)

my.df<-as.data.frame(inspect(my.tdm)) my.df.scale<-scale(my.df)

d<-dist(my.df.scale,method = "euclidean") fit<-hclust(d,method = "ward.D")

plot(fit)

## 1 a.pngOutput:



**Source code b (using bar plot with and without color):-**

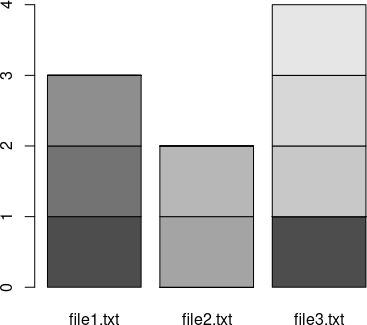
my.corpus<-Corpus(DirSource("/cloud/project/files"))

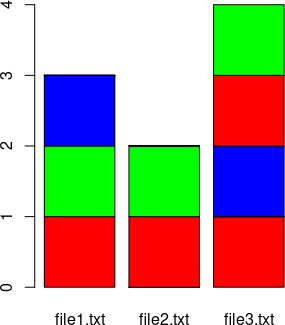
my.corpus<-tm\_map(my.corpus,removeWords,stopwords("english")) my.tdm<-TermDocumentMatrix(my.corpus)

inspect(my.tdm)

my.df<-as.data.frame(inspect(my.tdm)) barplot(as.matrix(my.tdm)) #barplot(as.matrix(my.tdm),col=color) barplot(as.matrix(my.tdm),col= c("Red","Green","Blue"))

## Output:





**Source code c (using minhash and jaccard similarity):-**

library("textreuse")

**Source Code**

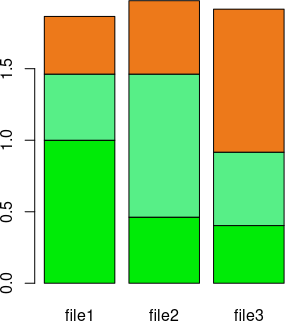
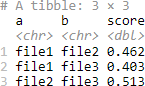
minhash <- minhash\_generator(200, seed = 235)

ats <- TextReuseCorpus(dir = "files", tokenizer = tokenize\_ngrams, n = 5, minhash\_func = minhash)

buckets <- lsh(ats, bands = 50, progress = interactive()) candidates <- lsh\_candidates(buckets)

scores <- lsh\_compare(candidates, ats, jaccard\_similarity, progress = F) scores

barplot(as.matrix(scores), col = c("#00eb07", "#57ef87", "#ed791a", "#5e5fff", "#1cf1c6", "#5e035b"))

**Output:**

# Practical 11

**Aim:** Write a java Program to demonstrate the k-moments for zeroth, first and second moments

## Input:

import java.io.\*; import java.util.\*;

class KMoments

{

public static void main(String[] args)

{

int n = 15;

String stream[] =

{"a","b","c","b","d","a","c","d","a","b","d","c","a","a","b"};

int zerothMoment = 0, firstMoment = 0, secondMoment = 0,

count = 1, flag = 0;

ArrayList<Integer> arrayList = new ArrayList(); for(String character : stream)

{

System.out.print(character + "\t\t\t");

}

System.out.println(); Arrays.sort(stream);

for(int i = 1; i < n; i++)

{

if(stream[i] == stream[i - 1]) count++;

else

{

}

}

arrayList.add(count); count = 1;

zerothMoment);

arrayList.add(count); System.out.println("Zeroth moment:\t\t\t" +

for(int i : arrayList)

{

firstMoment += i; secondMoment += i \* i;

}

secondMoment);

System.out.println("First Moment:\t\t\t" + firstMoment); System.out.println("Second Moment:\t\t\t" +

}

}

## Output:

