

Name: Medini Wakankar
Class: Msc CS Part I

Academic Year:2021-2022
Roll no:41

Subject: Business Intelligence and Big Data Analytics

Index

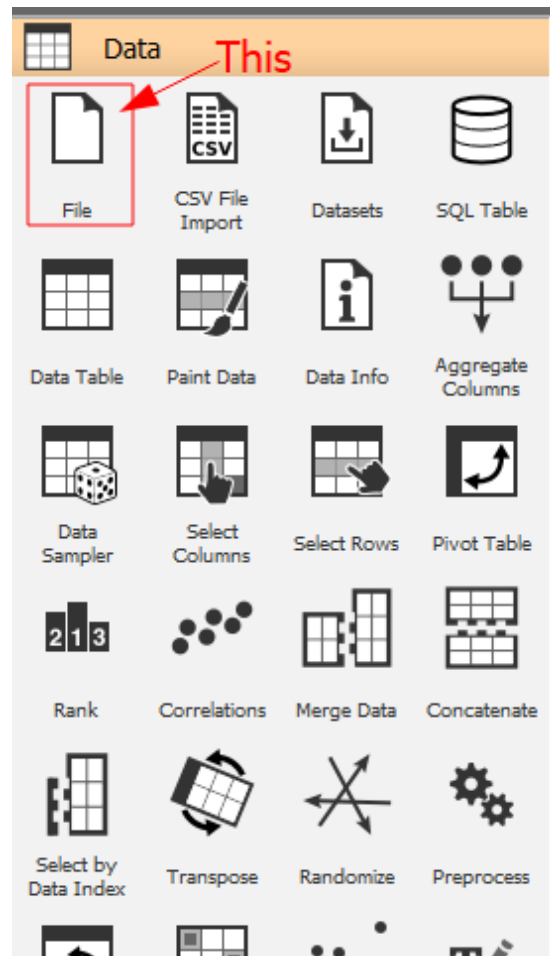
Sr.No	Title	Page No.
1	Classification using orange tool	2
2	Text Classification using orange tool	13
3	Image Classification using orange tool	29
4	Hierarchical clustering using orange tool	41
5	Text Preprocessing using orange tool	49
6	To predict whether the given dataset is of fruit or vegetable using 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.tab**dataset.

Source

☒ File: **zoo.tab**

☐ URL:

Info

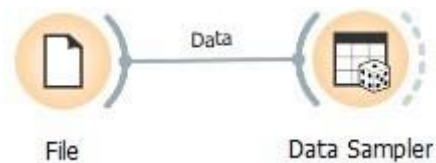
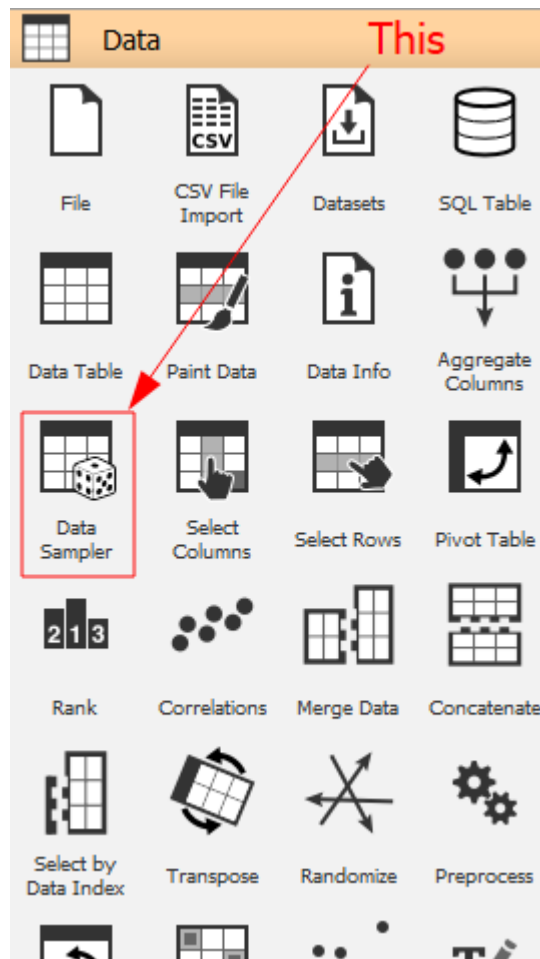
Zoo dataset
This dataset consists of 101 animals with various traits to describe them.

101 instance(s)
16 feature(s) (no missing values)
Classification; categorical class with 7 values (no missing values)
1 meta attribute(s)

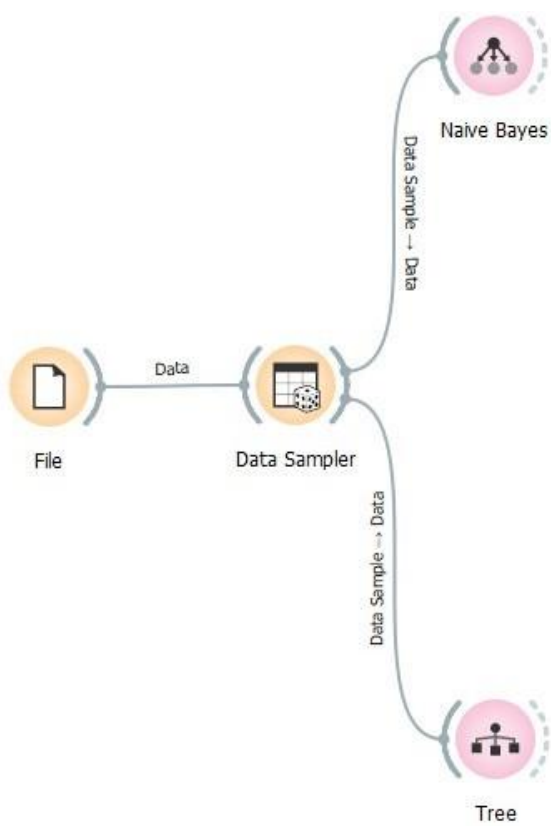
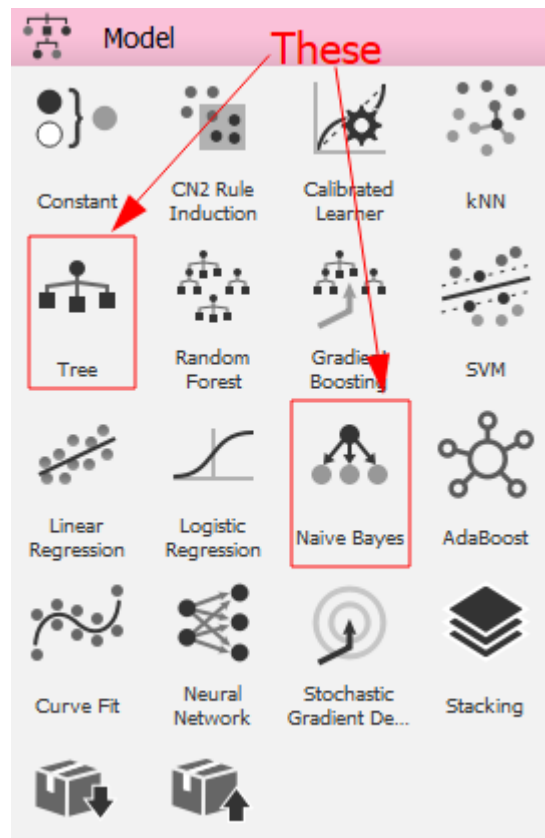
Columns (Double click to edit)

	Name	Type	Role	Values
1	hair	categorical	feature	0, 1
2	feathers	categorical	feature	0, 1
3	eggs	categorical	feature	0, 1
4	milk	categorical	feature	0, 1
5	airborne	categorical	feature	0, 1
6	aquatic	categorical	feature	0, 1

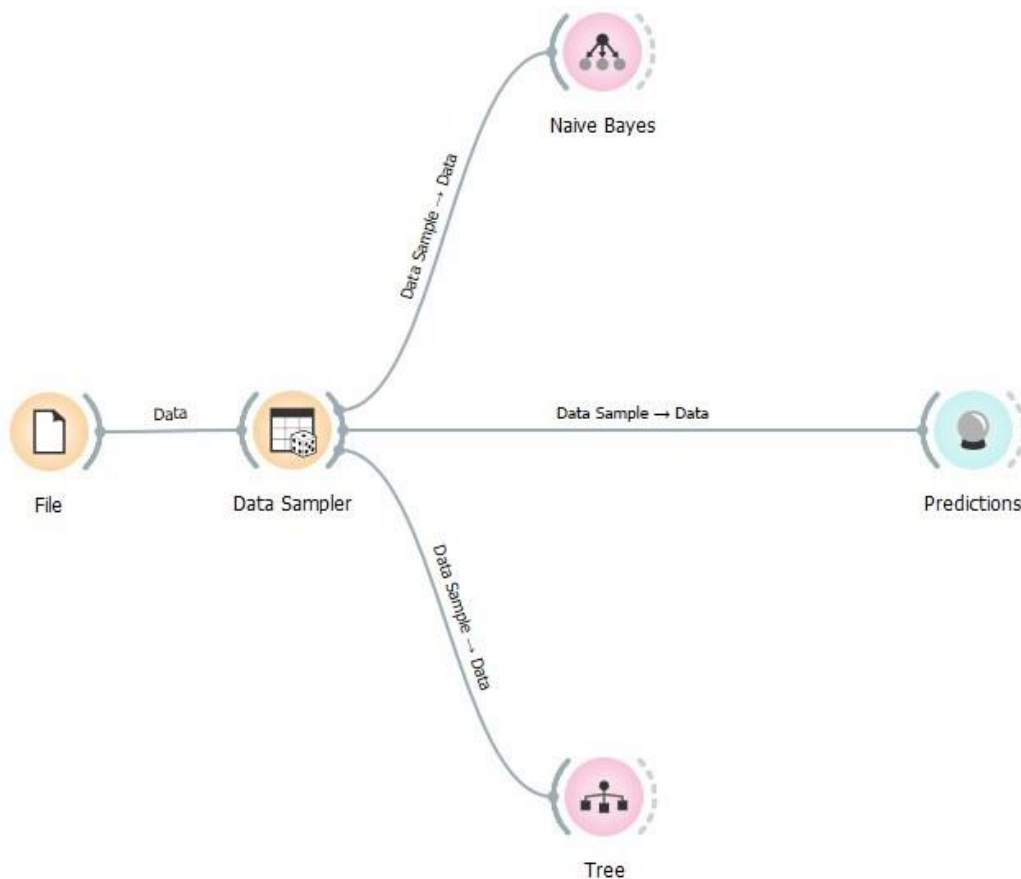
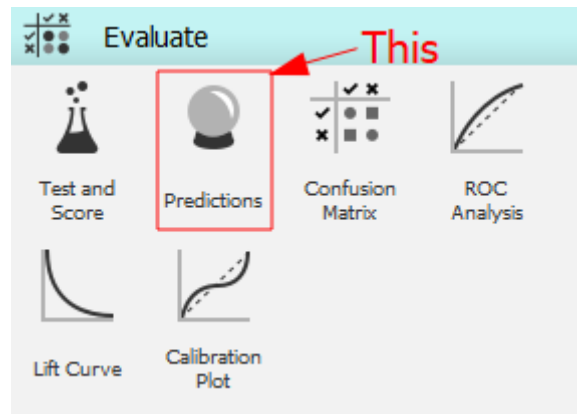
- Drag and drop a Data Sampler widget onto the workspace and connect it to the File widget. This widget is found in the Data section.



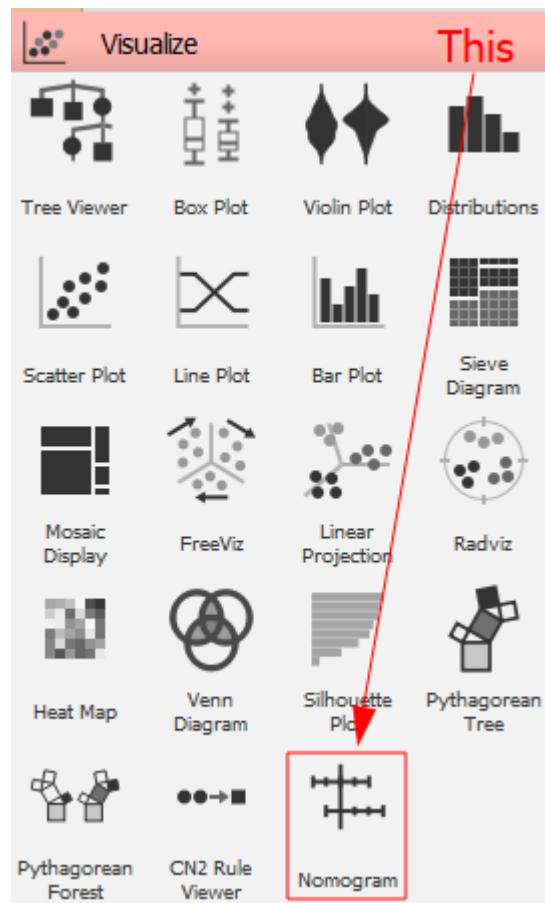
- We choose the Naive Bayes' and Classification Tree models to classify our data. Drag and drop these widgets to the workspace. They are found under the Models section.

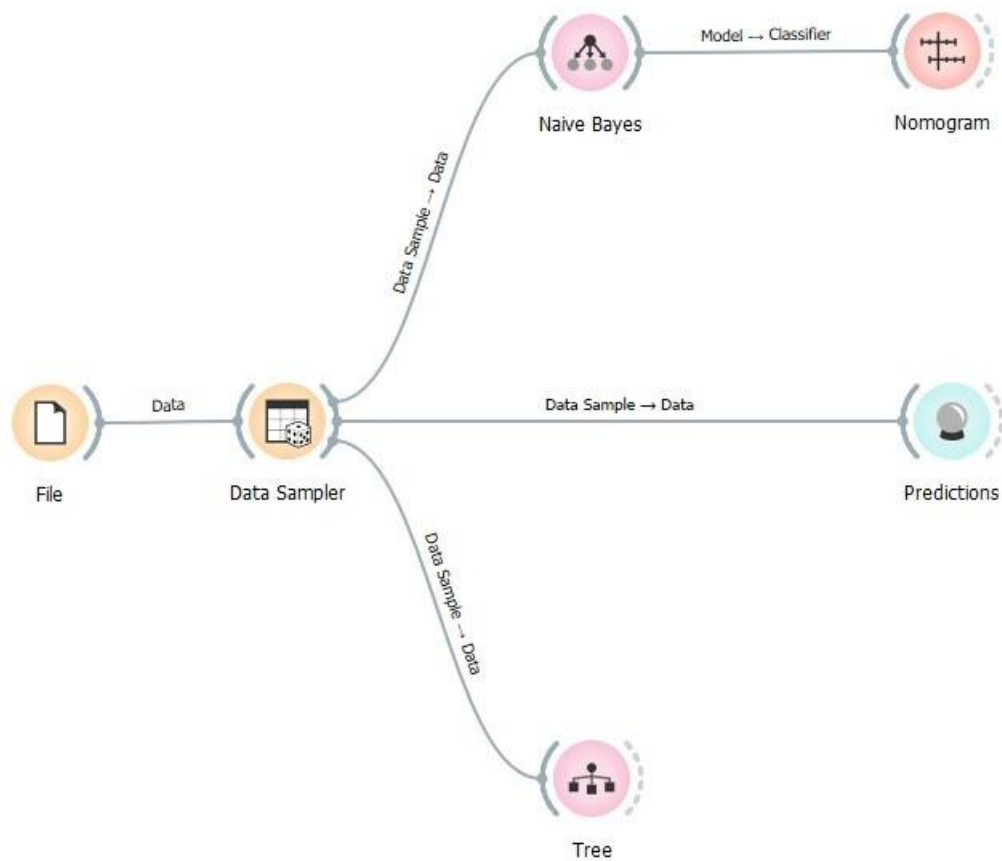


- To perform predictions on the data, we drag and drop the Predictions widget onto the workspace. This widget is found in the Evaluate section.

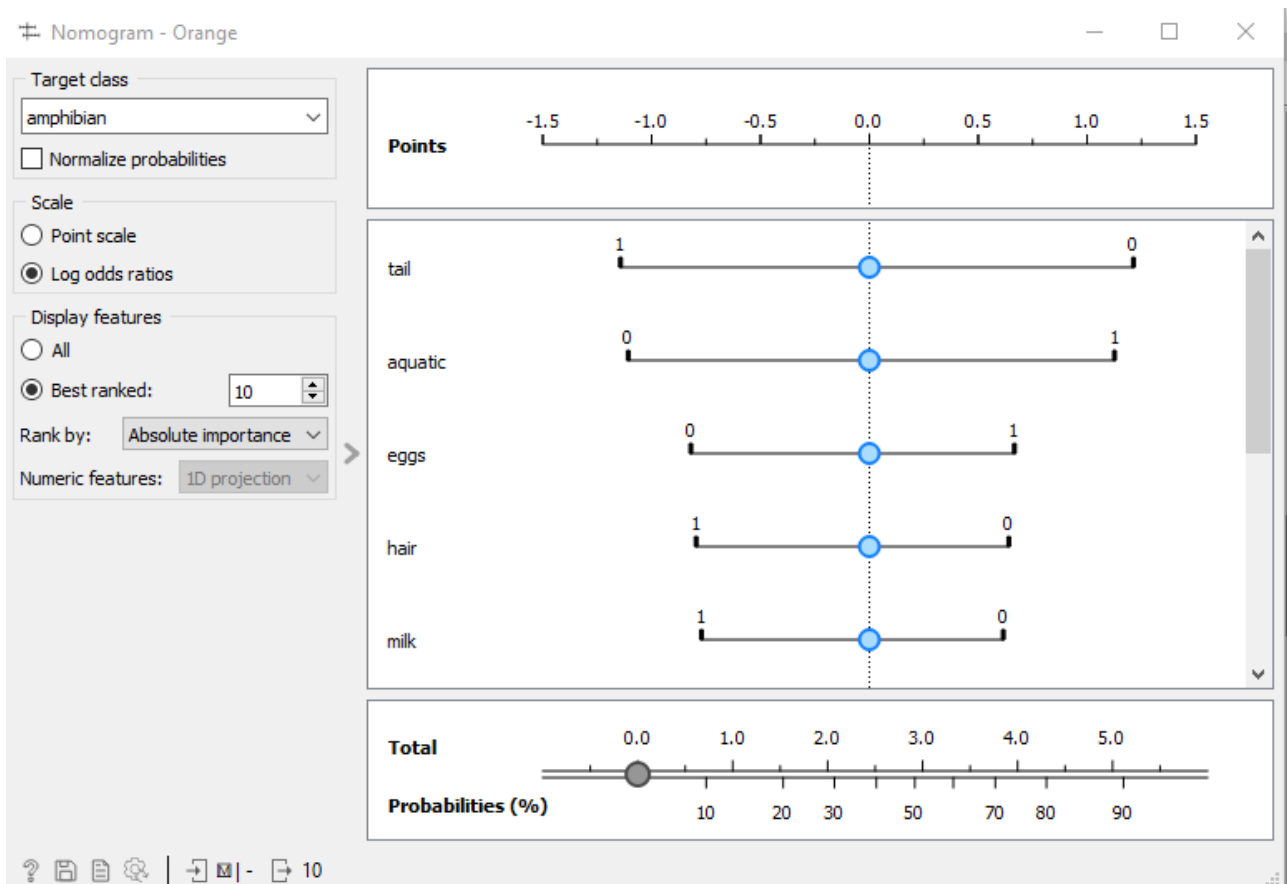
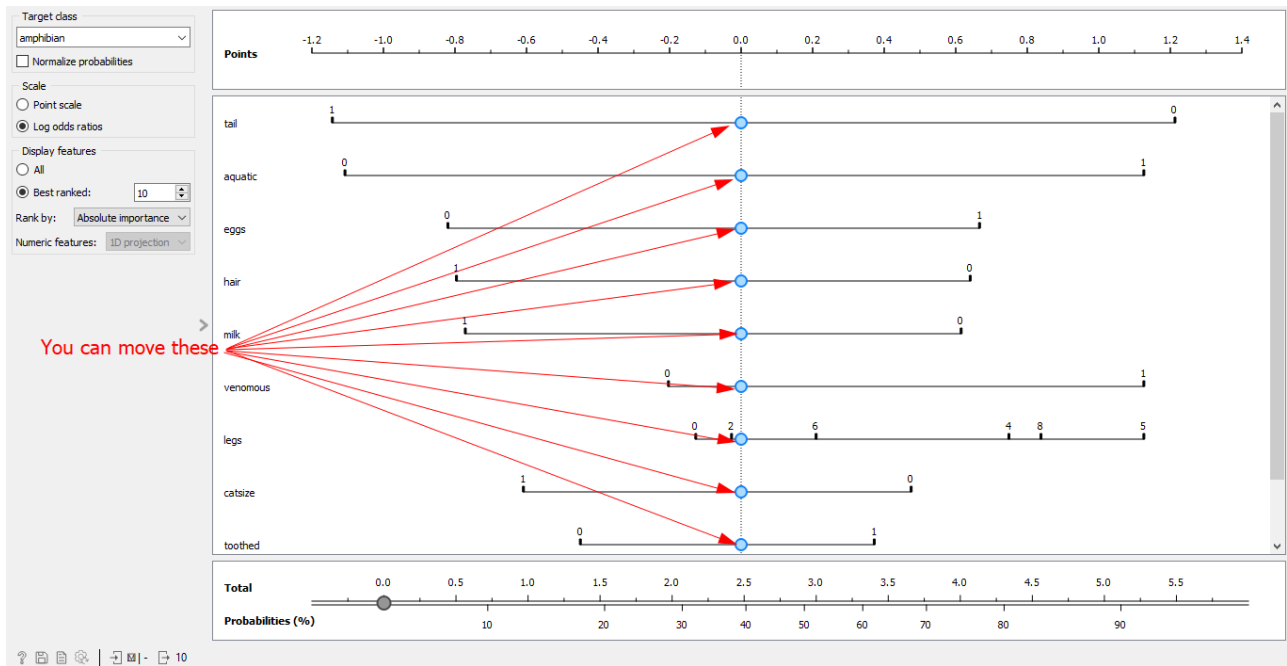


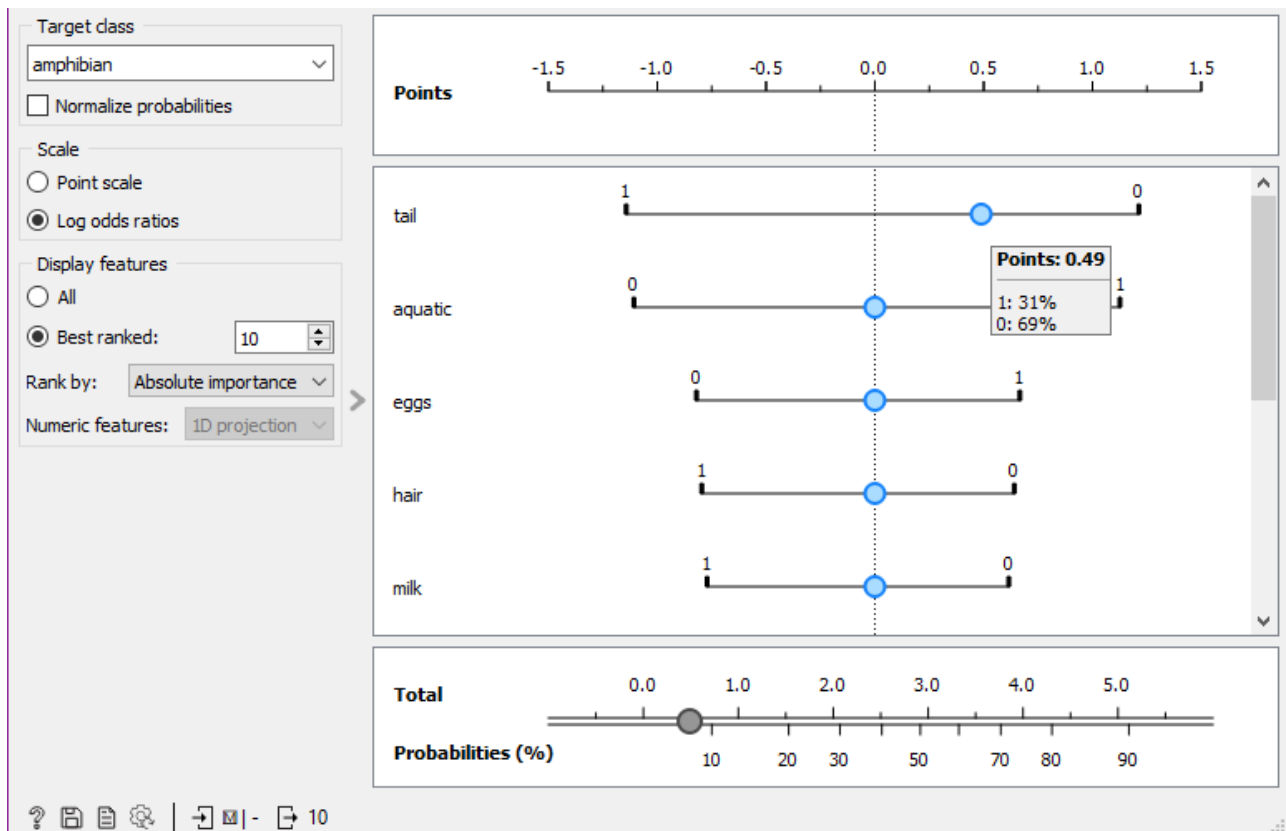
- A Nomogram is useful to view data from a Naive Bayes' model. Drag and drop this widget onto the workspace. It can be found in the Visualise section.



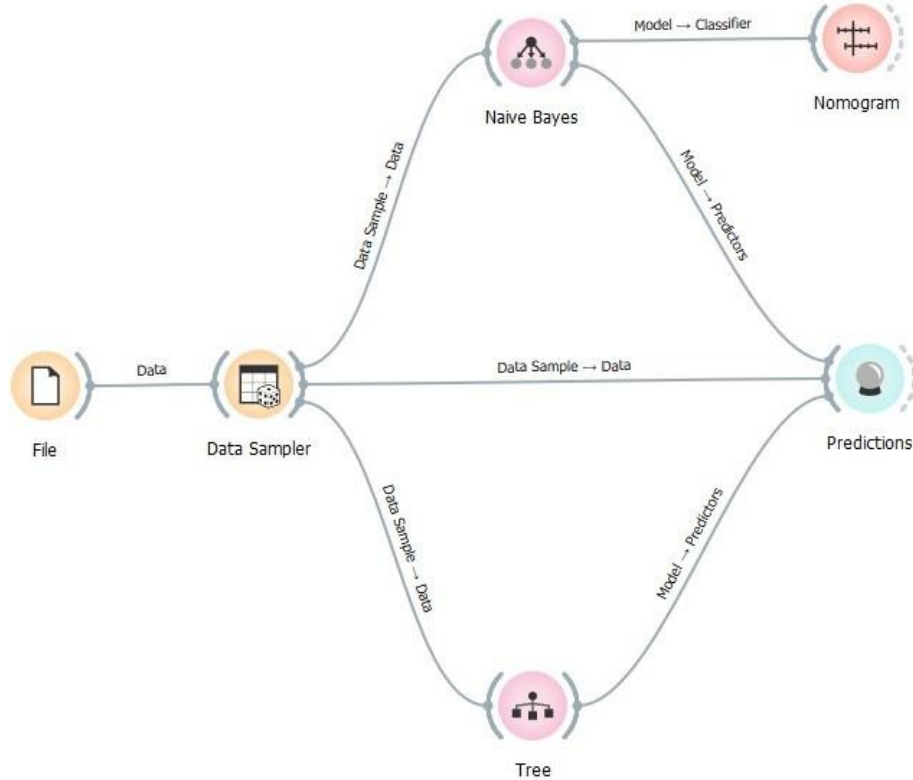


- You can move the points in the Nomogram to see the probabilities of a particular class. Here 1 indicates favourable probability while 0 indicates unfavourable probability.





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

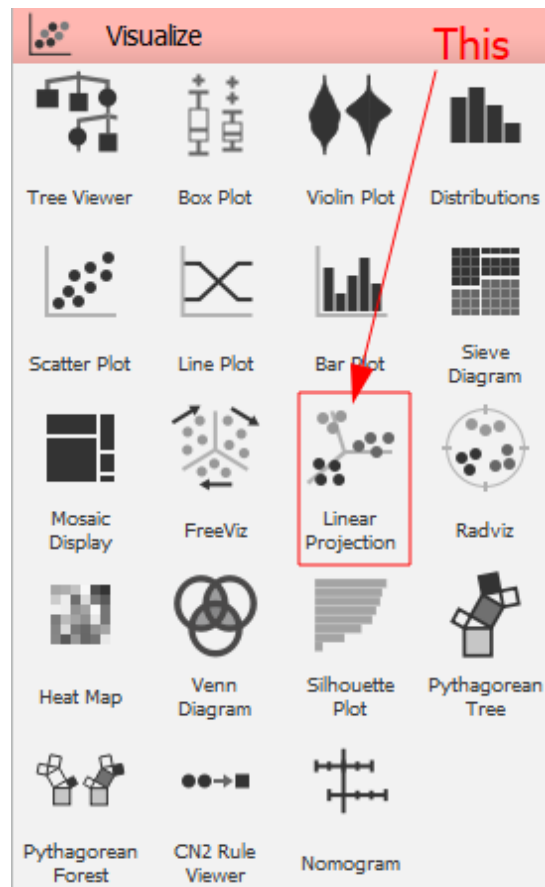


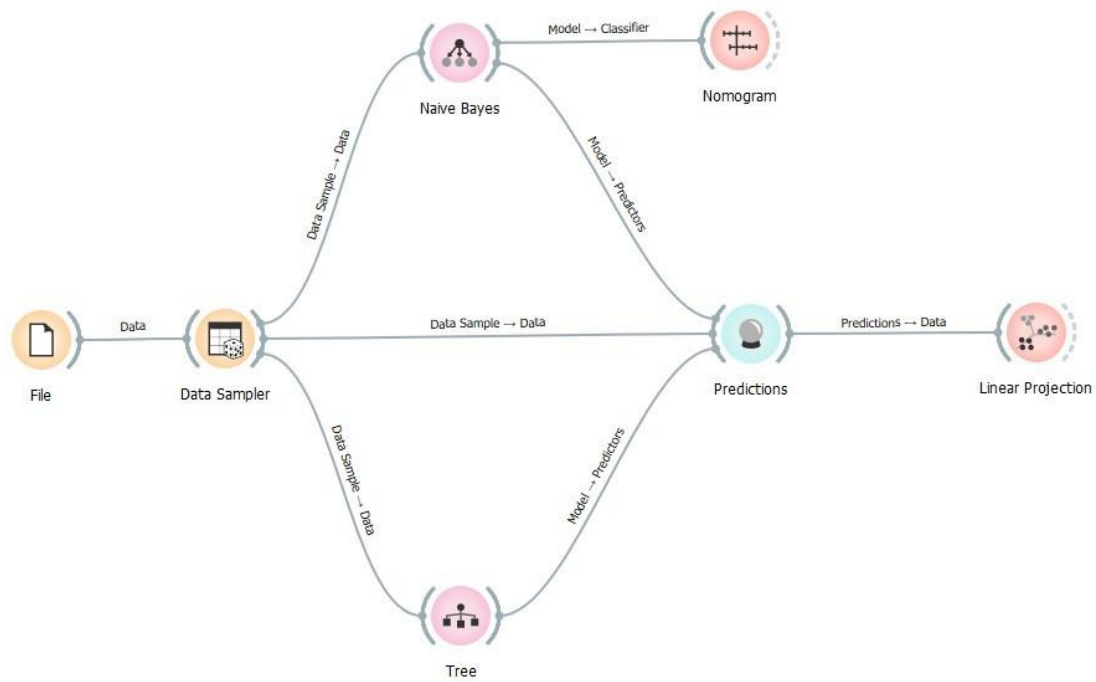
➤ Double click the Predictions widget to see the predictions.

Model	AUC	CA	F1	Precision	Recall
Naive Bayes	1.000	0.944	0.948	0.967	0.944
Tree	0.999	0.986	0.986	0.988	0.986

type	name	hair	feathers	eggs	milk
mammal	squirrel	1	0	0	1
mammal	oryx	1	0	0	1
mammal	porpoise	0	0	0	1
mammal	puma	1	0	0	1
mammal	lion	1	0	0	1
mammal	honeybee	1	0	1	0
mammal	elephant	1	0	0	1
mammal	leopard	1	0	0	1
mammal	cheetah	1	0	0	1
mammal	aardvark	1	0	0	1
fish	dogfish	0	0	1	0
insect	gnat	0	0	1	0
insect	wasp	1	0	1	0
bird	gull	0	1	1	0
invertebrate	seawasp	0	0	1	0
mammal	boar	1	0	0	1
mammal	vampire	1	0	0	1
bird	skimmer	0	1	1	0
fish	chub	0	0	1	0
mammal	goat	1	0	0	1
reptile	seasnake	0	0	0	0
amphibian	toad	0	0	1	0
amphibian	frog	0	0	1	0
insect	beetle	0	0	1	0

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



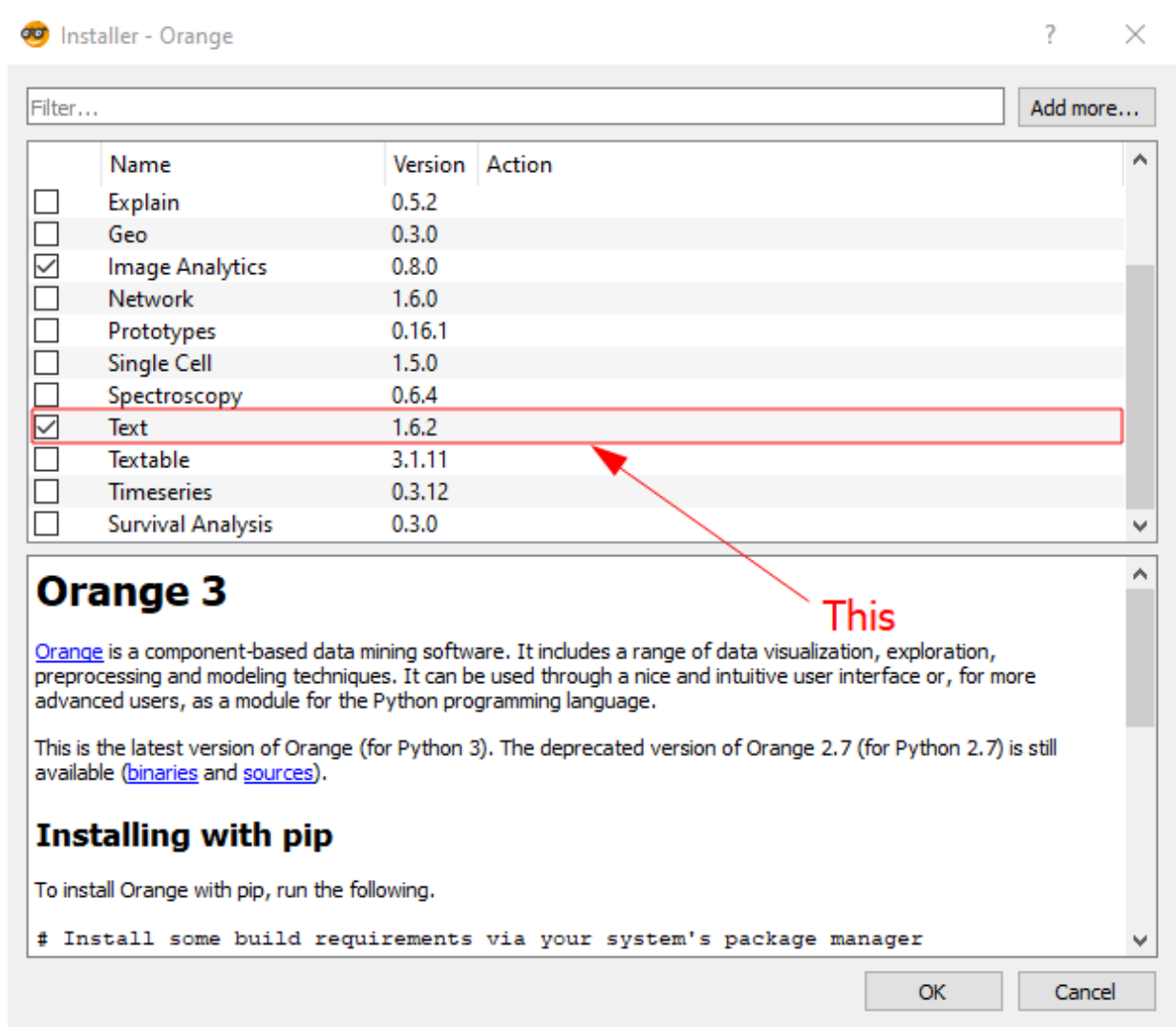


Practical 2

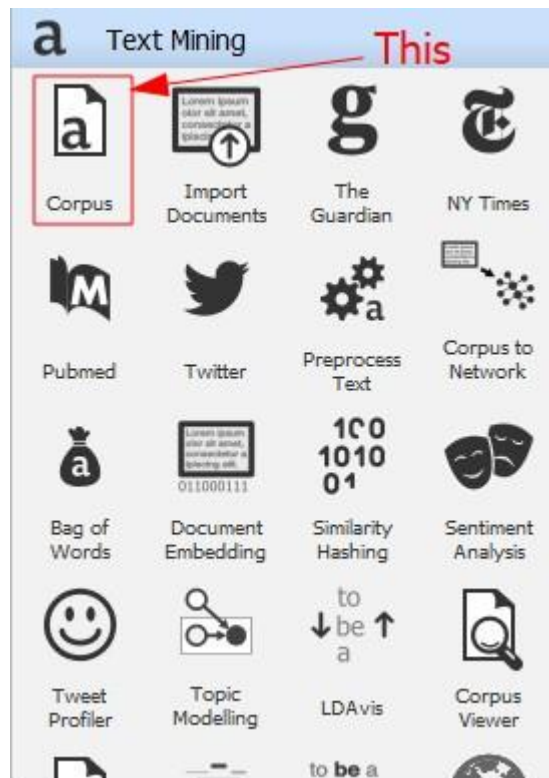
Aim: Text classification using Orange.

Steps:

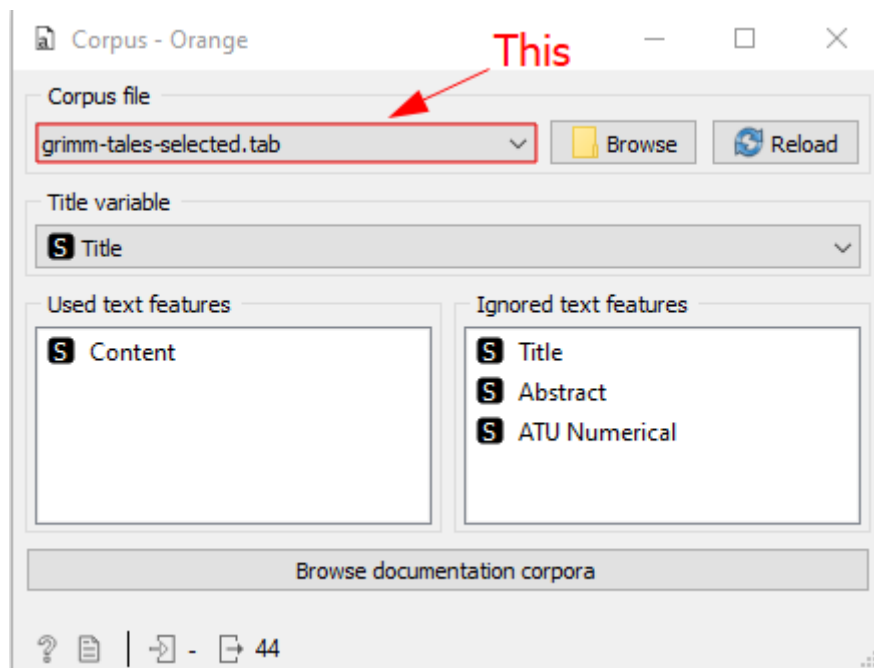
- Before starting, we need to install the **Text** add-on. Install it by navigating to **Options > Add ons...**. You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.



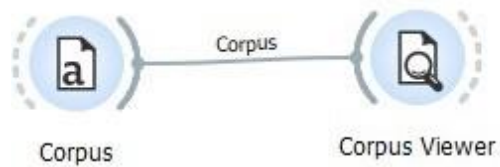
- Drag and drop a Corpus widget to the workspace. It can be found in the newly added Text Mining section.



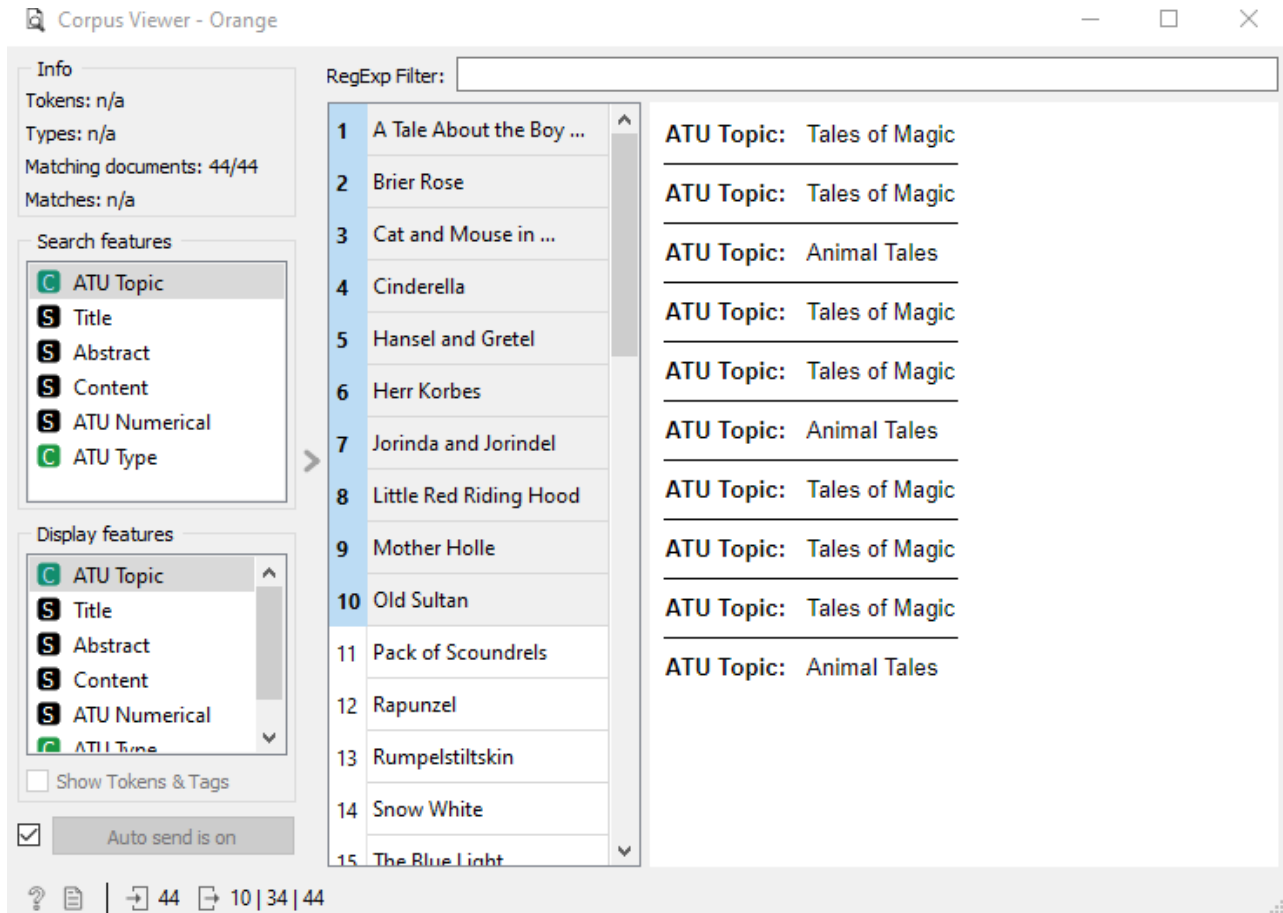
- Double click the Corpus widget and select the **grimm-tales-selected.tab** corpus file.



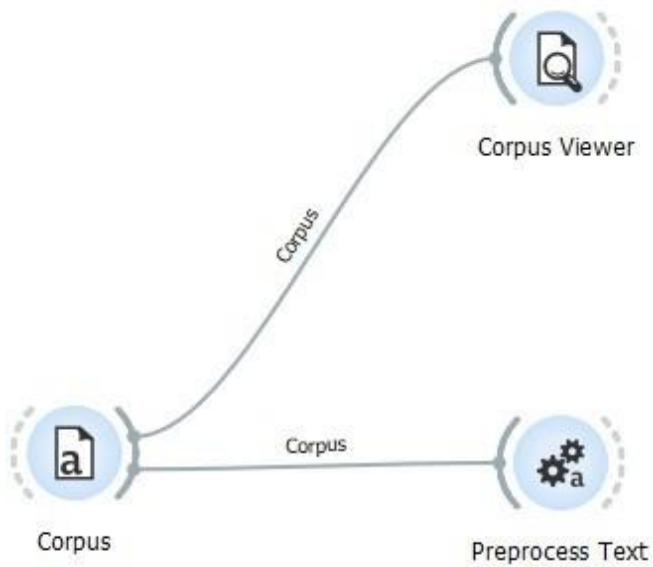
- Drag and drop a Corpus Viewer widget and connect it to the Corpus widget. The Corpus Viewer widget can be found in the Text Mining section.



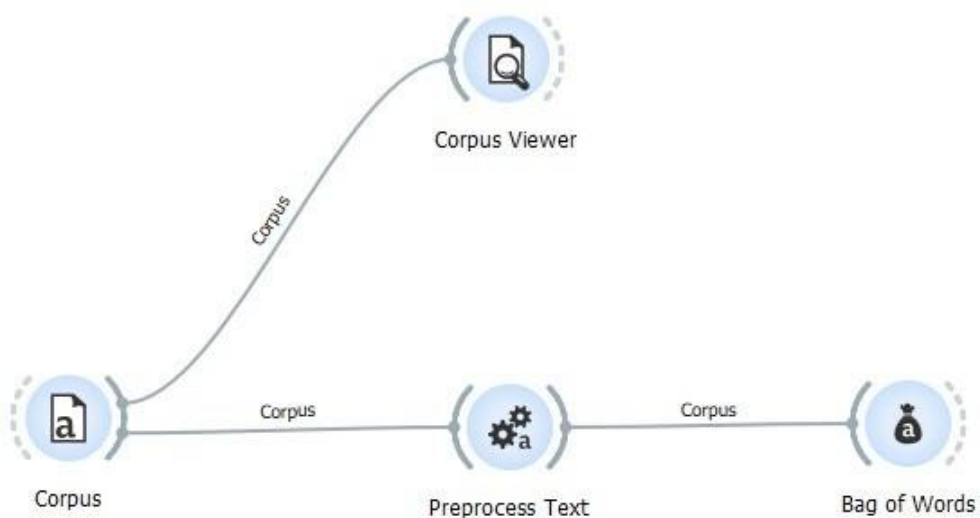
- Double click the Corpus Viewer widget to visualize the corpus in a tabular format. Select the first 10 entries to use as training set.



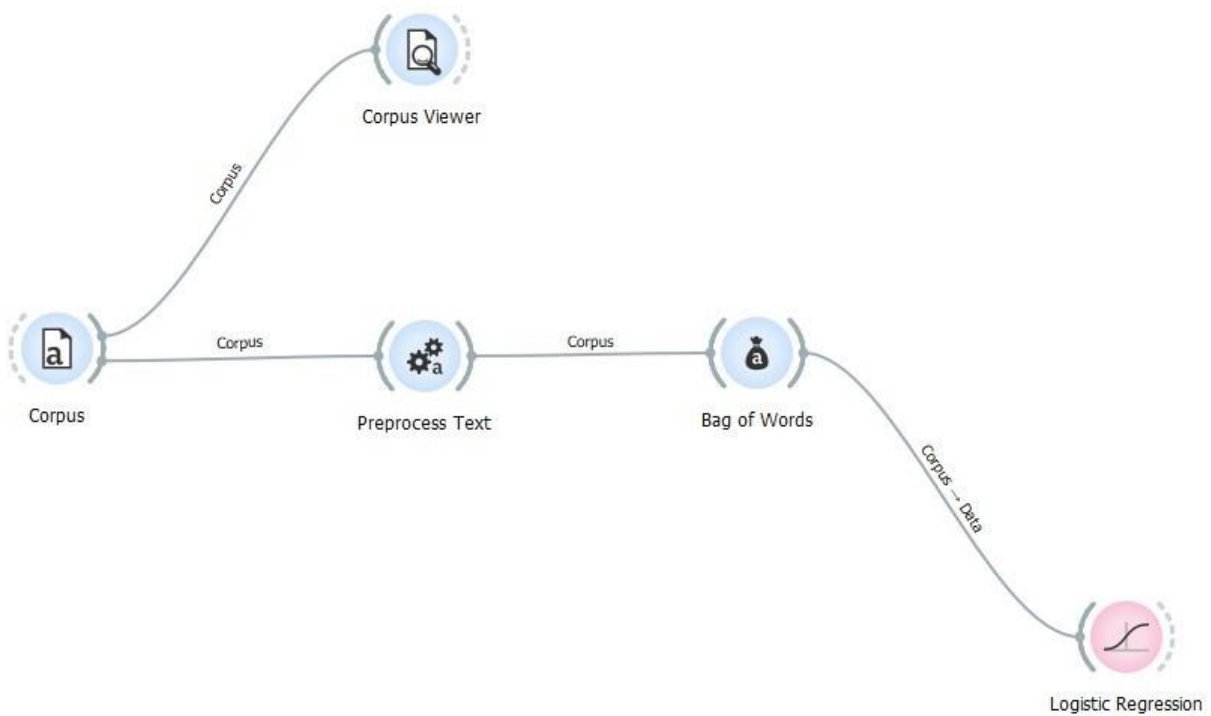
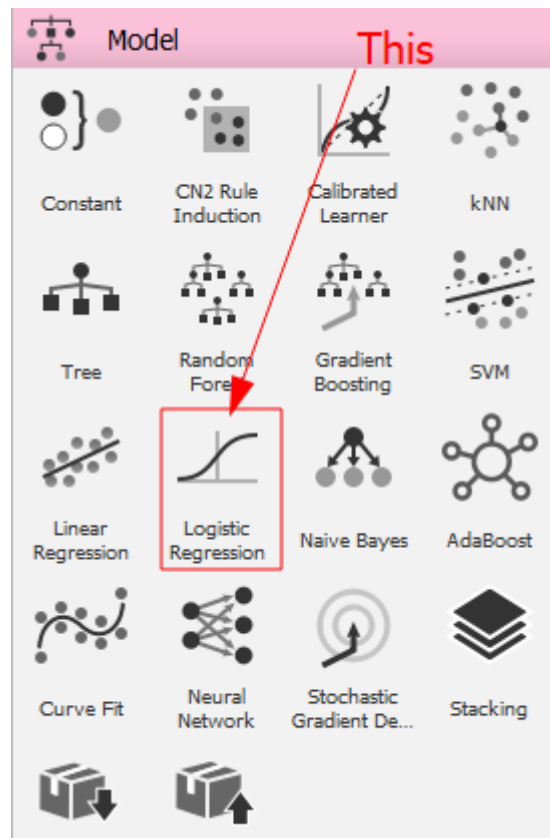
- Add a Preprocess Text widget to the workspace. This widget can also be found in the Text Mining section.



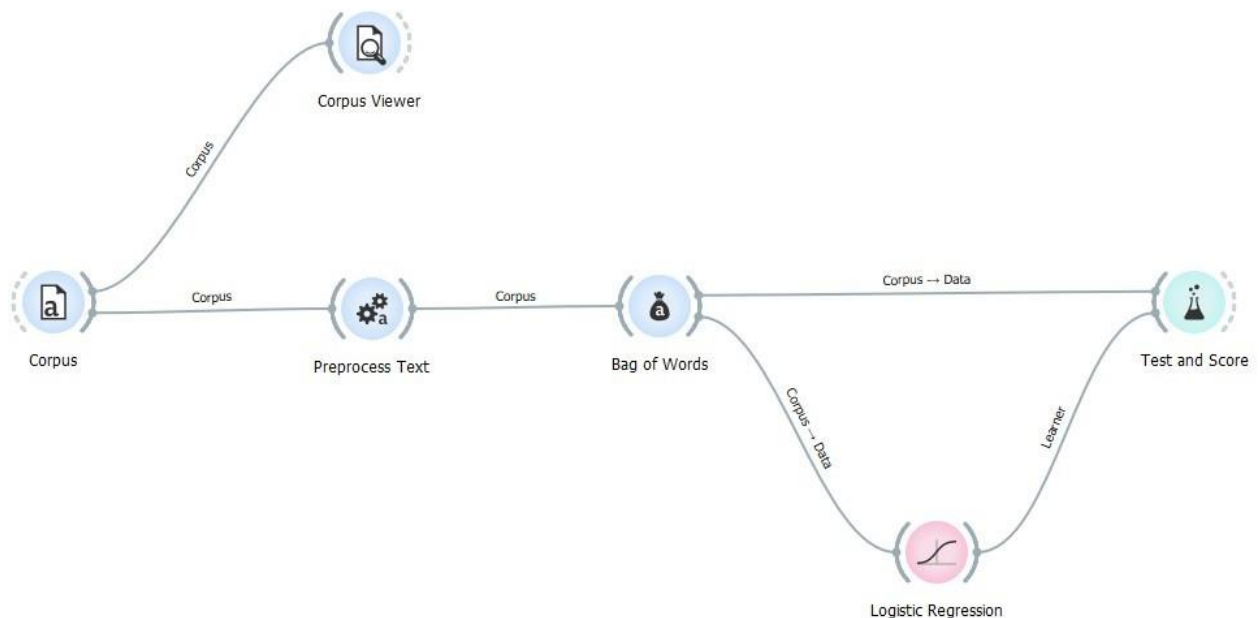
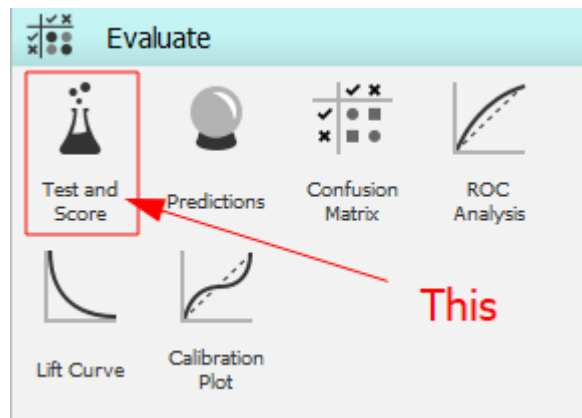
- Add a Bag Of Words widget to the workspace and connect it to the Preprocess Text widget. This widget is also found in the Text Mining section.



- 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 Evaluate section 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.

The screenshot shows two widgets from a machine learning software interface. The left widget is titled 'Sampling' and contains several configuration options. The right widget is titled 'Evaluation Results' and displays a table of performance metrics for a Logistic Regression model. Below this is a 'Model Comparison by AUC' widget showing a comparison between the Logistic Regression model and itself.

Sampling Configuration:

- ☒ Cross validation
 - Number of folds: 10
 - ☒ Stratified
- ☐ Cross validation by feature
 - ATU Type
- ☐ Random sampling
 - Repeat train/test: 10
 - Training set size: 66 %
 - ☒ Stratified
- ☐ Leave one out
- ☐ Test on train data
- ☐ Test on test data
- Target Class: (Average over classes)
- Model Comparison: Area under ROC curve
- ☐ Negligible difference: 0.1

Evaluation Results Table:

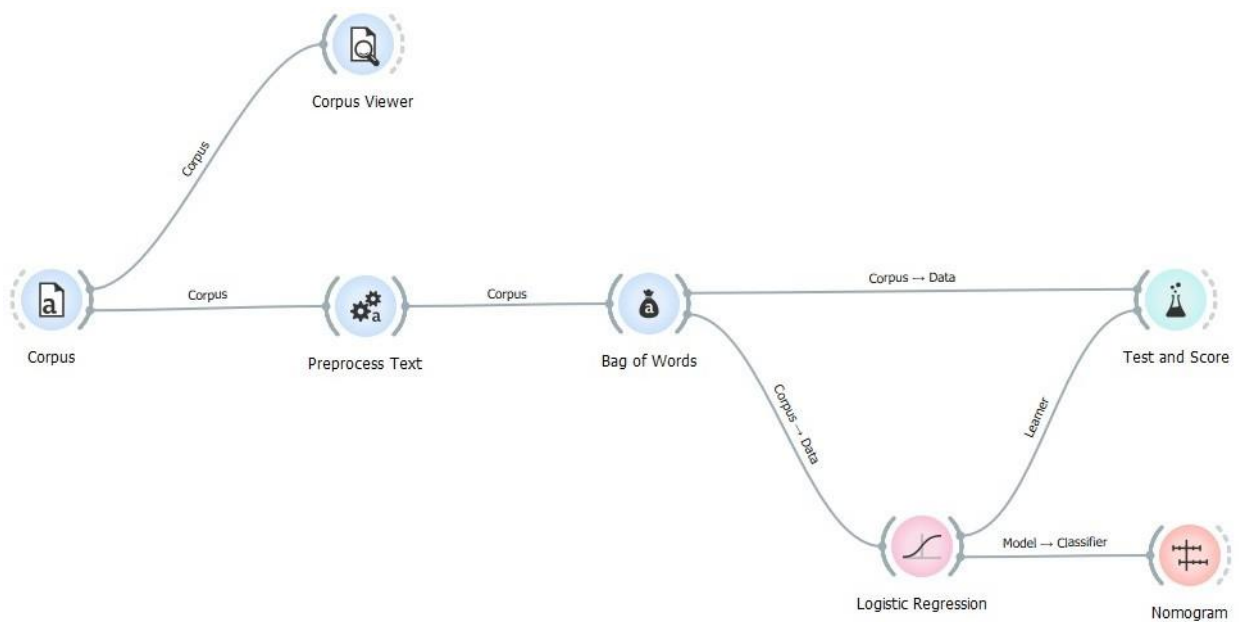
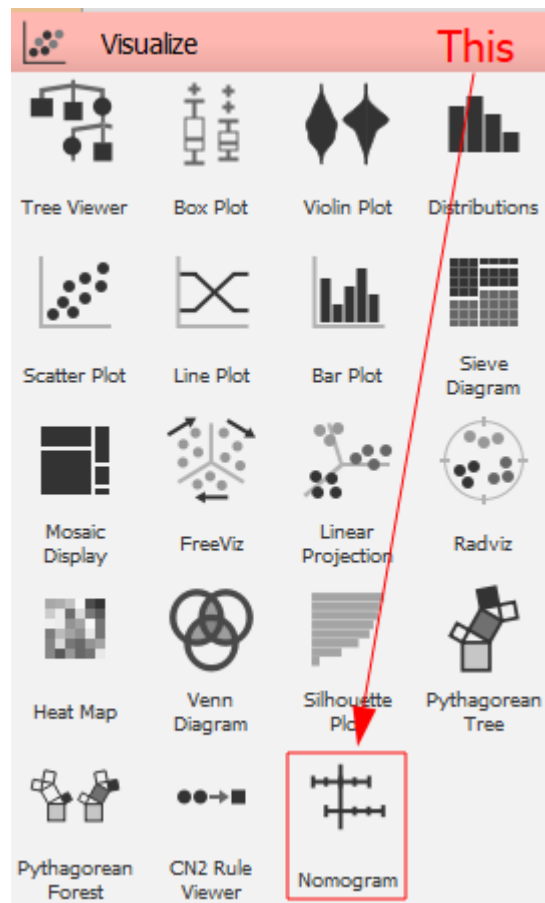
Model	AUC	CA	F1	Precision	Recall
Logistic Regression	0.968	0.909	0.910	0.926	0.909

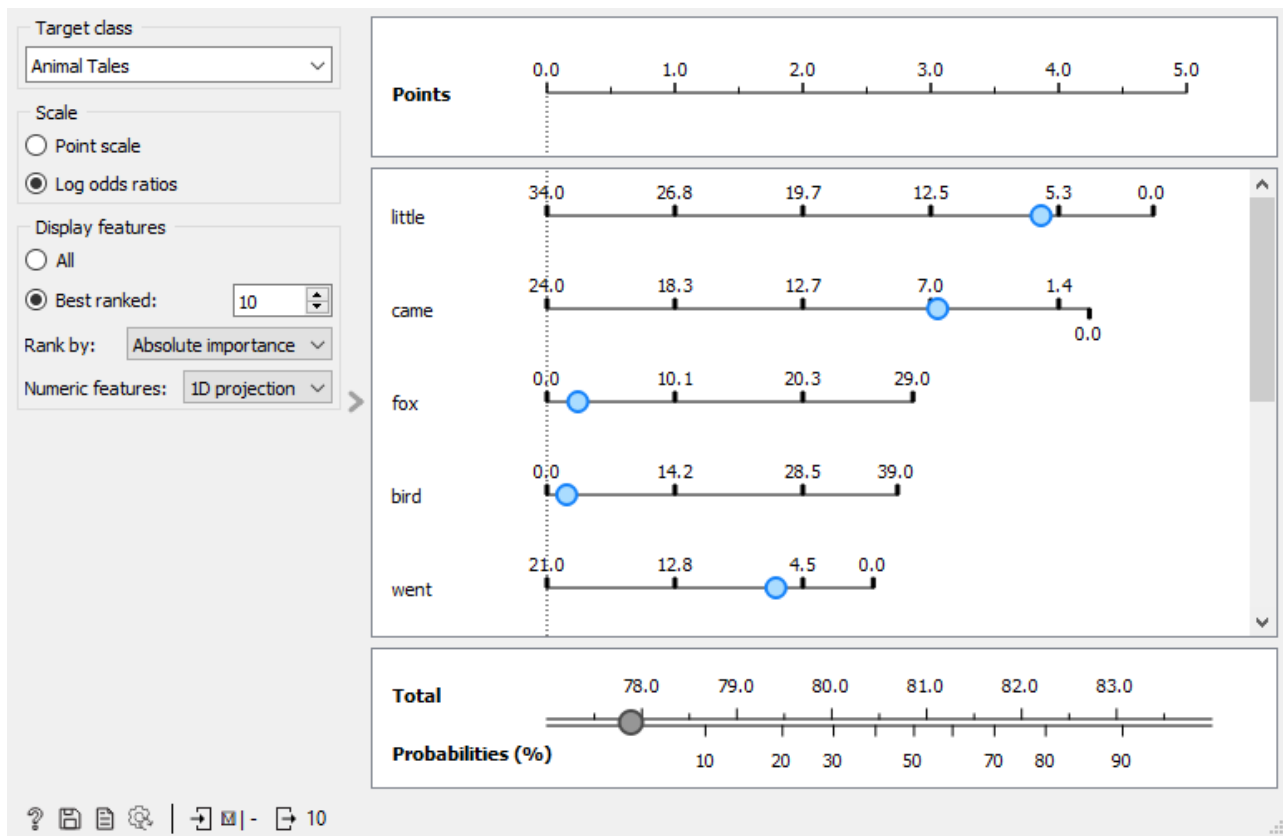
Model Comparison by AUC:

	Logistic ...
Logistic Regression	

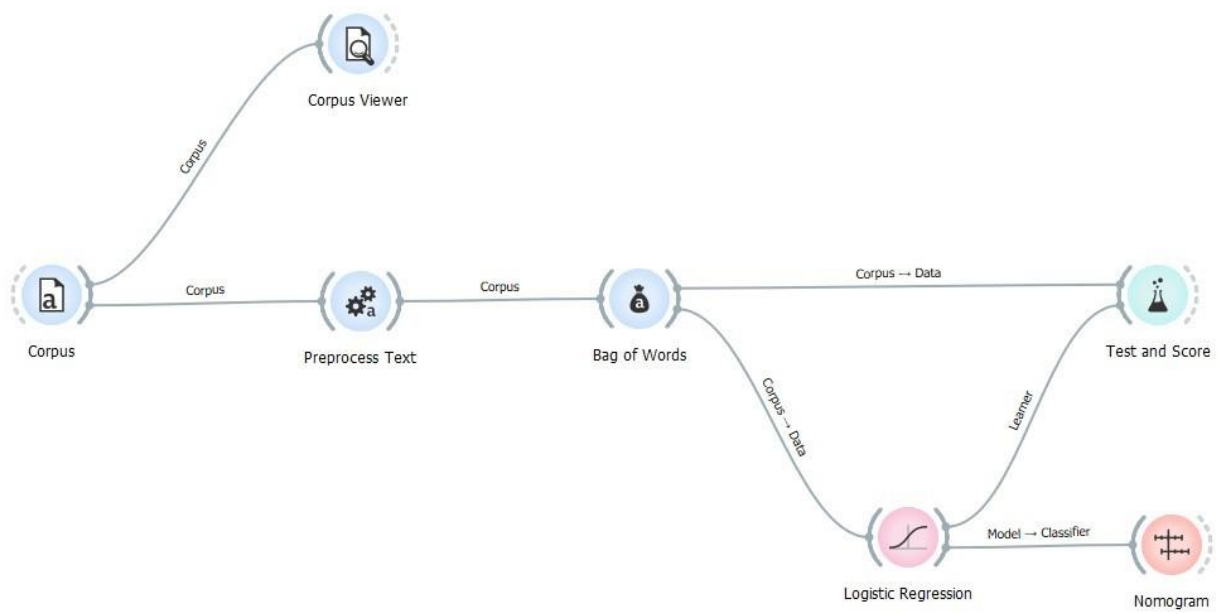
Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

- 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**.



Corpus file

andersen.tab

Browse

Reload

Title variable

Title

Used text features

Content

Ignored text features

Title

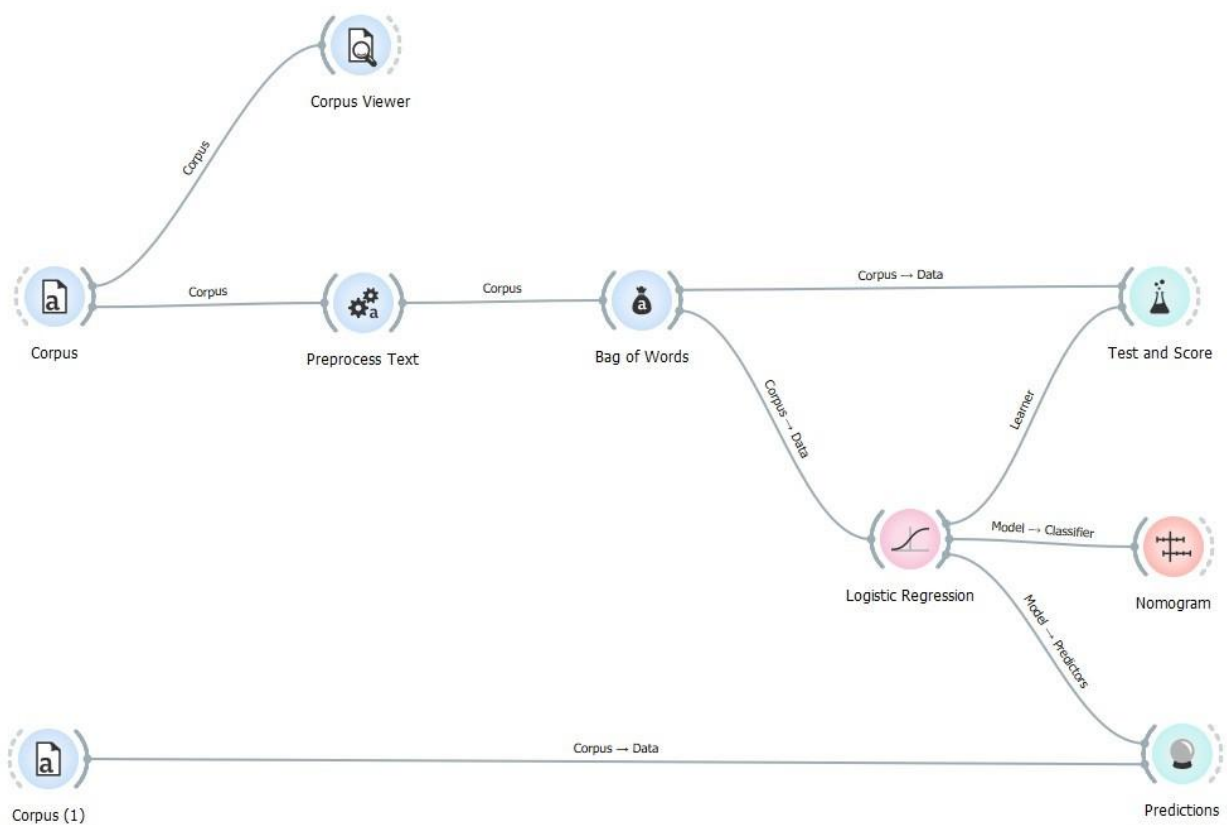
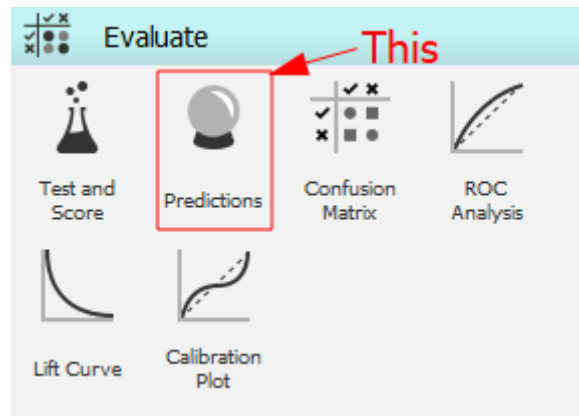
Browse documentation corpora

?

-

3

- 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.



Show probabilities for

Animal Tales

Tales of Magic

Restore Original Order

Logistic Regression

1 0.18 : 0.82 → Tales of Ma...

2 0.00 : 1.00 → Tales of Ma...

3 0.00 : 1.00 → Tales of Ma...

Title

Content

The Little Matc...It was terribly c...

The Philosophie...Far away towar...

The Ugly Duckli...It was lovely su...

?

3

3

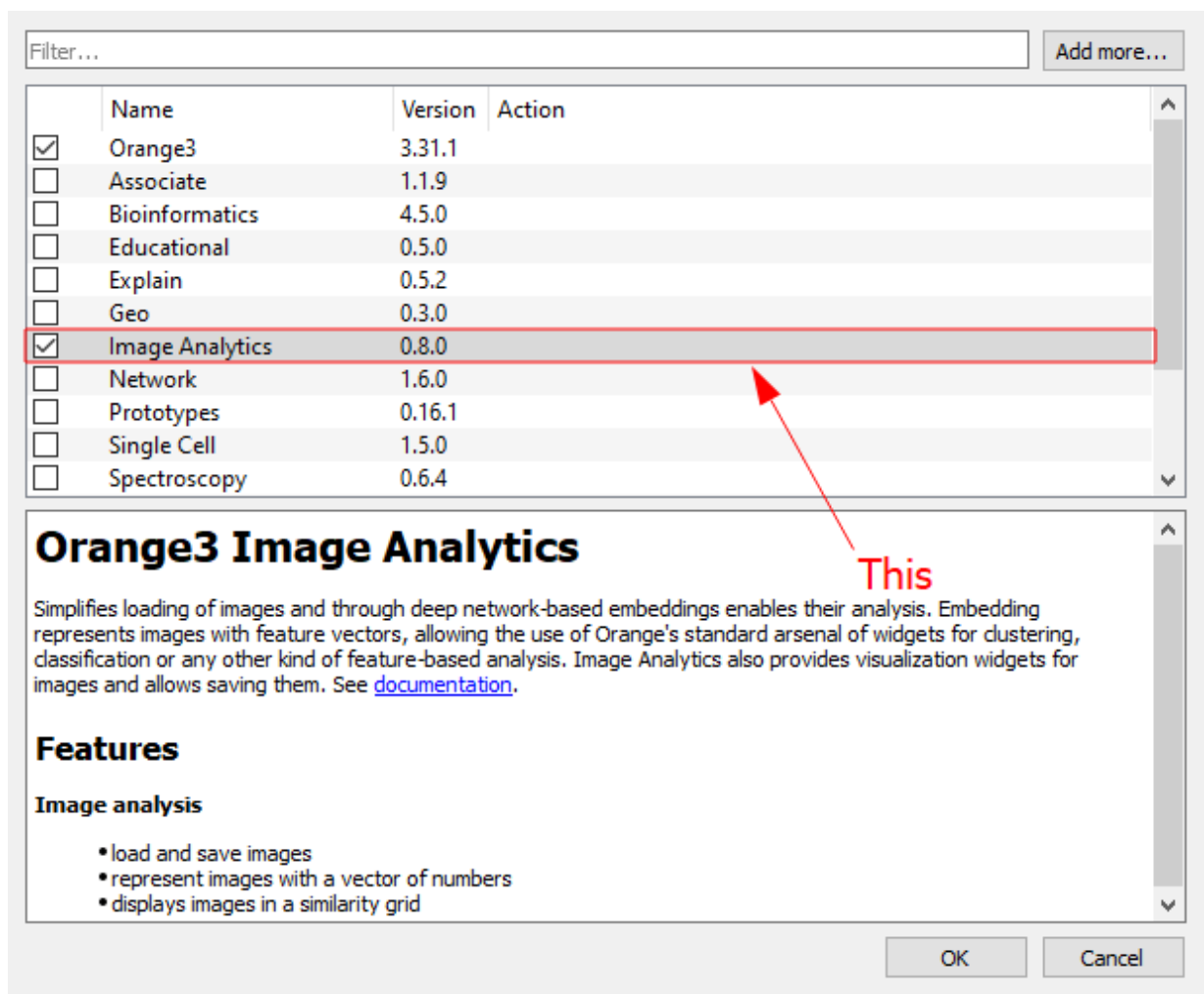
-

Practical 3

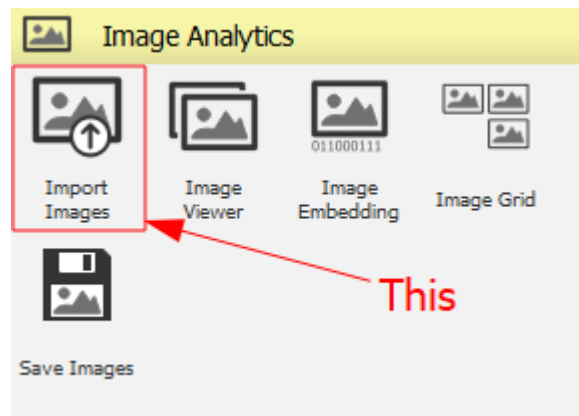
Aim: Image classification using orange.

Steps:

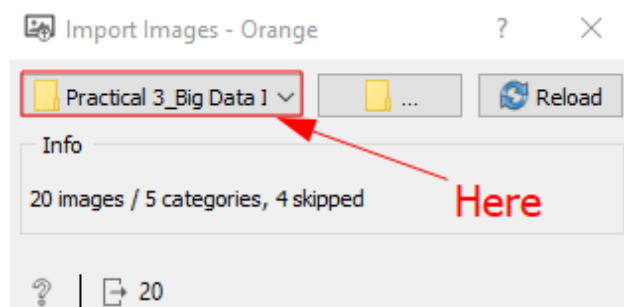
- Before starting, we need to install the ImageProcessing add-on. Install it by navigating to Options > Add ons.... You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.



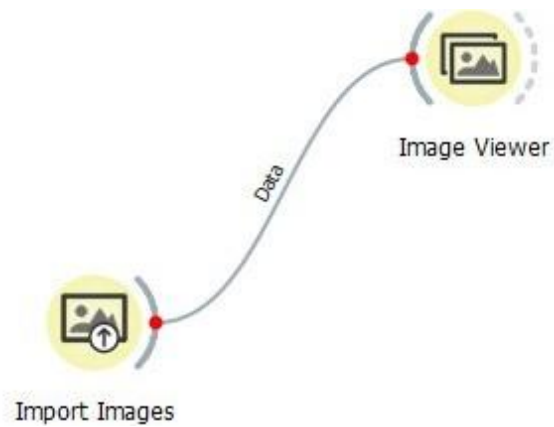
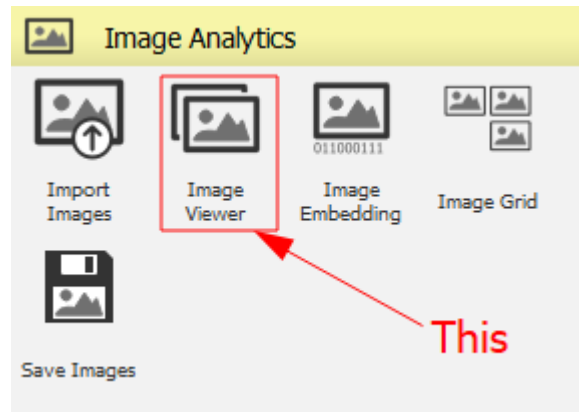
- Add a Import Images widget from the newly added Image Analytics section to the workspace.

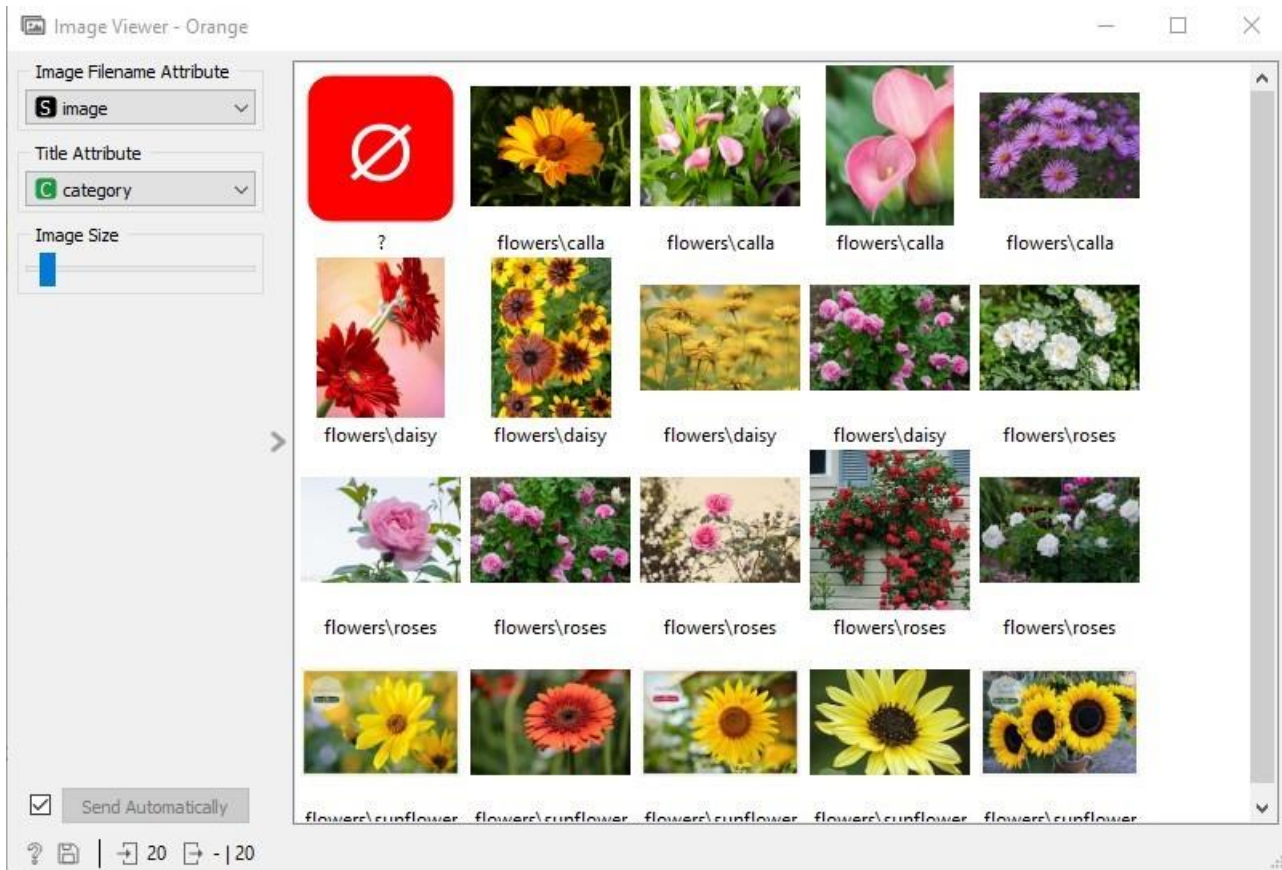


- Load the directory containing the images in the newly added Import Images widget by double clicking the widget.

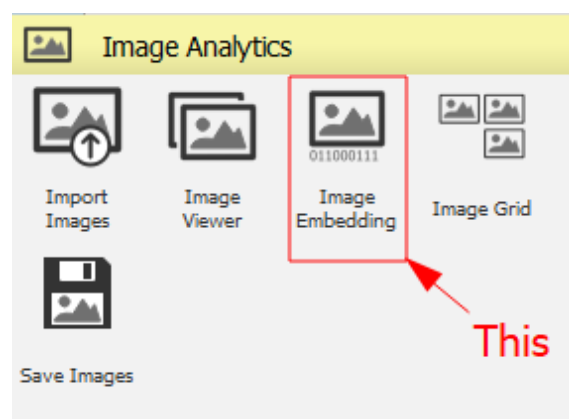


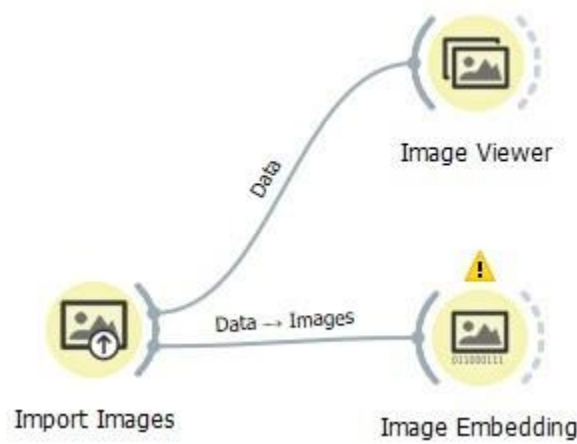
- We can view these images using the Image Viewer widget found in the Image Analytics section. Drag and drop this widget onto the workspace and connect it to the Import Images widget.



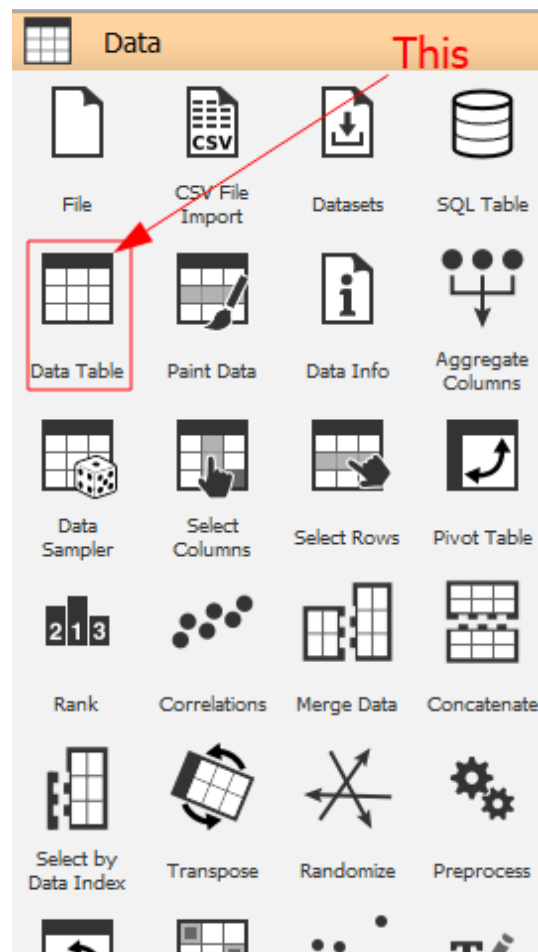


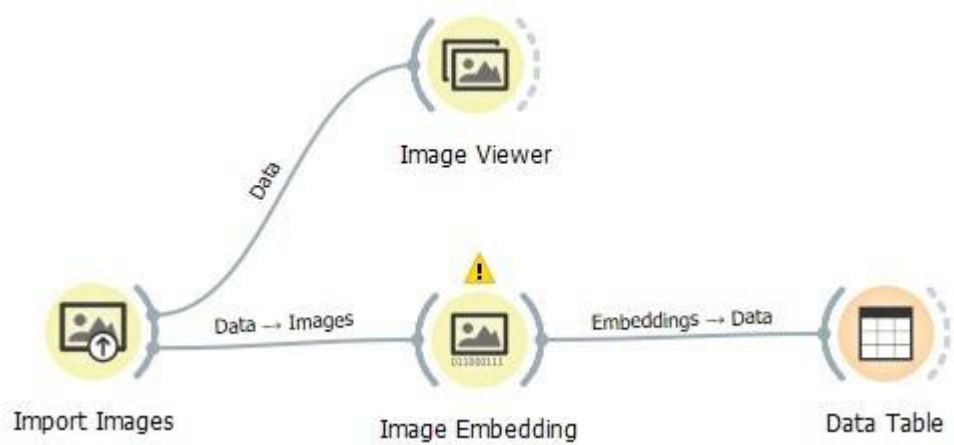
- As our models can only process numbers, we need to convert the images into numerical data. This is where Image Embedding comes into play. Drag and drop a Image Embedding widget from the Image Analytics section onto the workspace and connect it to the Import Images widget.





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





Info

18 instances (no missing data)
1000 features
Target with 5 values
5 meta attributes

Variables

☒ Show variable labels (if present)

☐ Visualize numeric values

☒ Color by instance classes

Selection

☒ Select full rows

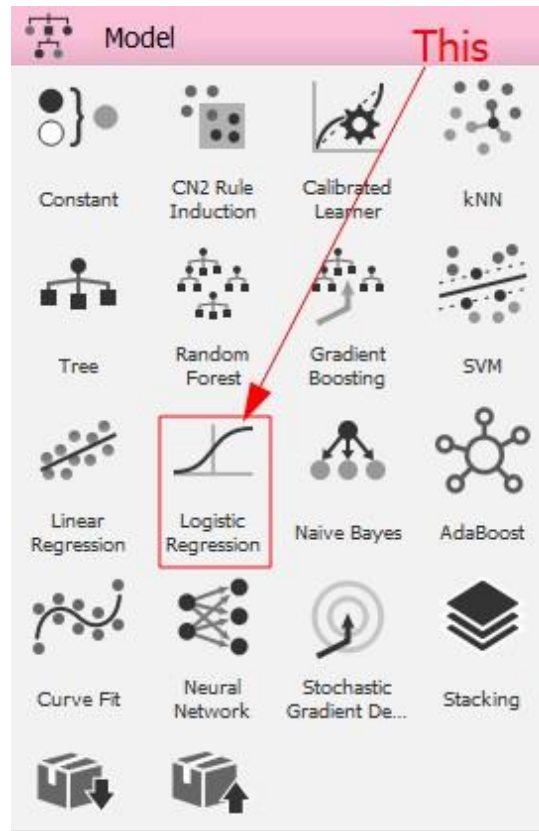
Restore Original Order

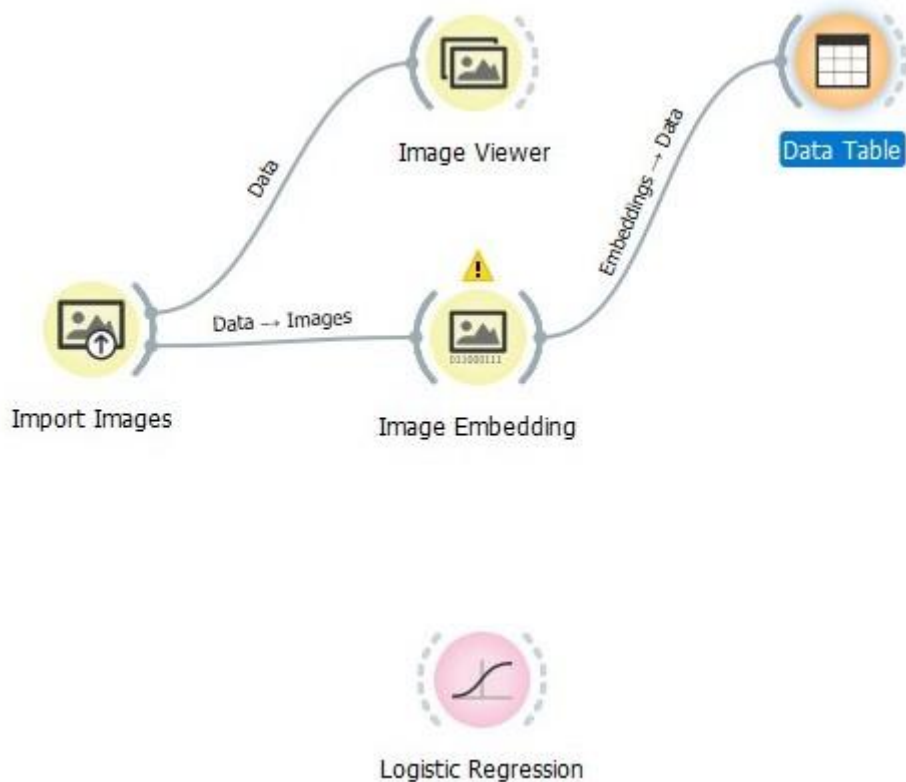
☒ Send Automatically

? | 18 | 18 | 18

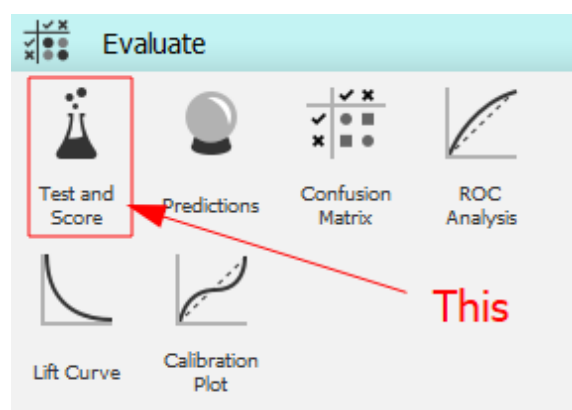
	category	image name	image	size	width
1	flowers\calla	calla	flowers\calla\c...	3429626	2560
2	flowers\calla	calla1	flowers\calla\c...	29712	400
3	flowers\calla	calla2	flowers\calla\c...	2890	105
4	flowers\daisy	daisy1	flowers\daisy\d...	39660	400
5	flowers\daisy	daisy4	flowers\daisy\d...	71325	375
6	flowers\daisy	daisy6	flowers\daisy\d...	3206304	2084
7	flowers\daisy	daisy8	flowers\daisy\d...	526653	933
8	flowers\roses	rose1	flowers\roses\r...	535368	933
9	flowers\roses	rose2	flowers\roses\r...	354357	933
10	flowers\roses	rose3	flowers\roses\r...	526653	933
11	flowers\roses	rose4	flowers\roses\r...	429411	933
12	flowers\roses	rose6	flowers\roses\r...	841733	700
13	flowers\roses	rose7	flowers\roses\r...	523616	933
14	flowers\sunflo...	sun1	flowers\sunflo...	6649	170
15	flowers\sunflo...	sun2	flowers\sunflo...	5481	176
16	flowers\sunflo...	sun3	flowers\sunflo...	7061	170
17	flowers\sunflo...	sun4	flowers\sunflo...	5808	177
18	flowers\sunflo...	sun5	flowers\sunflo...	222011	800

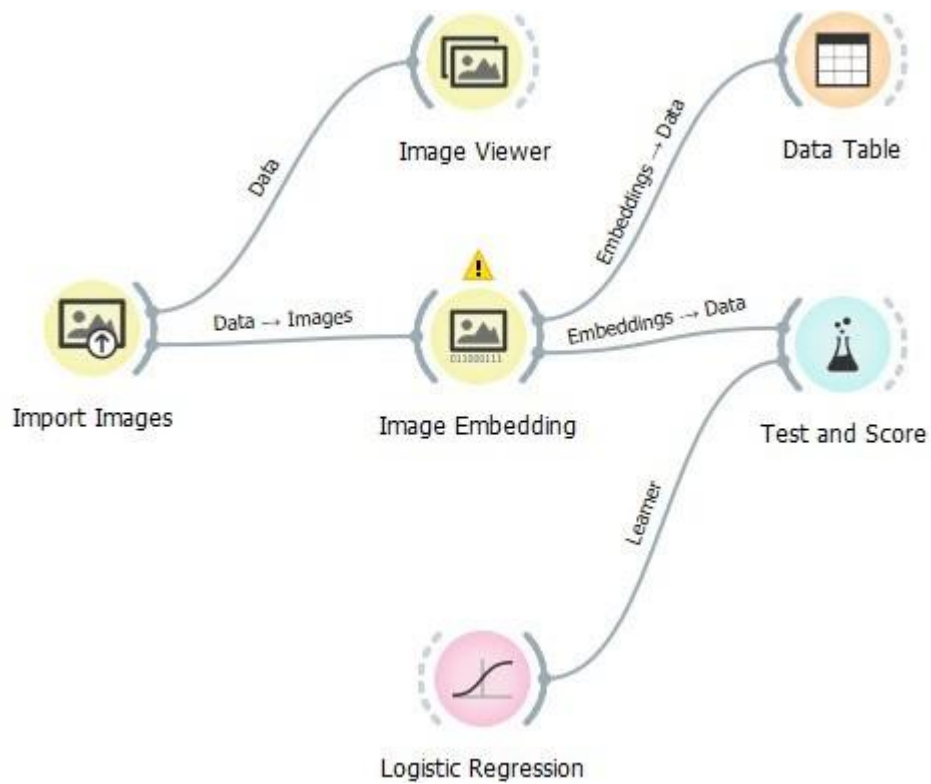
- We will use Logistic Regression as our model. Drag and drop a Logistic Regression widget from the Model section onto the workspace.





- Drag and drop a Test and Score widget from the Evaluate section onto the workspace. This widget will allow us to verify our model. Connect this widget to the Image Embedding and Logistic Regression widgets.





Sampling

☒ Cross validation

Number of folds:

☒ Stratified

☐ Cross validation by feature

☐ Random sampling

Repeat train/test:

Training set size:

☒ Stratified

☐ Leave one out

☐ Test on train data

☐ Test on test data

Target Class

Model Comparison

☐ Negligible difference:

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Logistic Regression	0.718	0.556	0.519	0.543	0.556

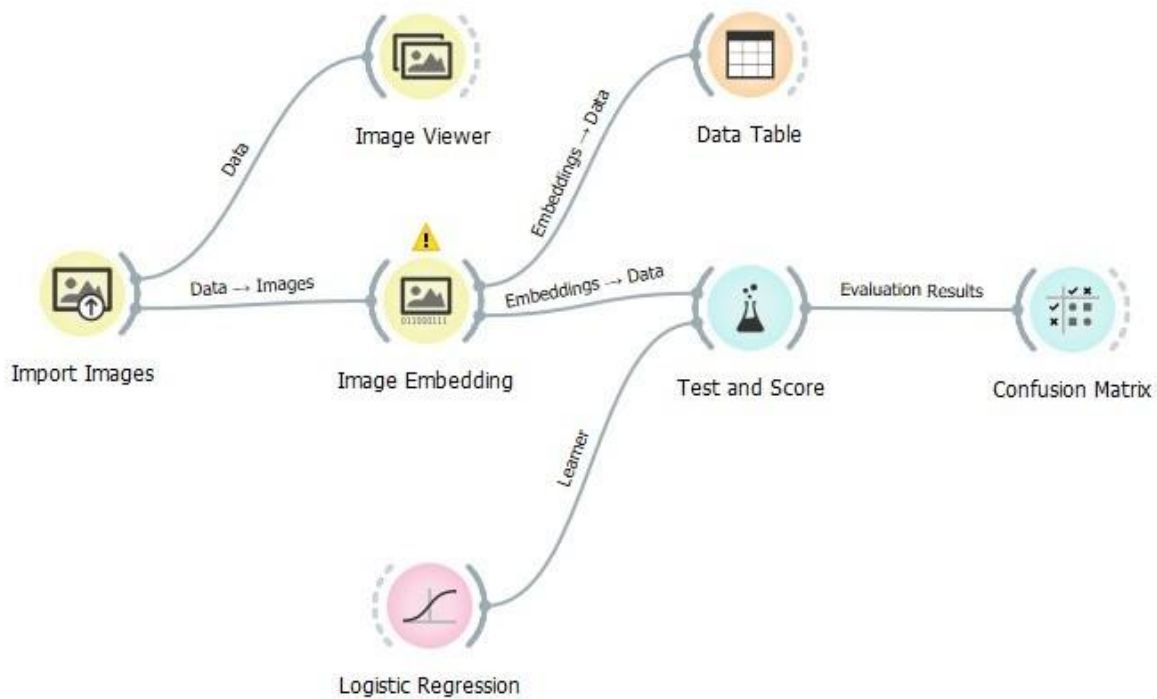
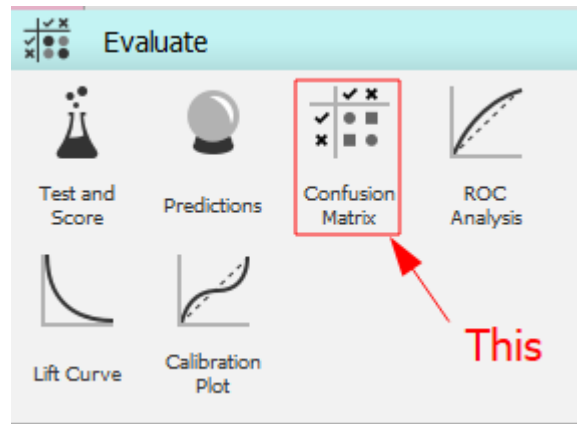
Model Comparison by AUC

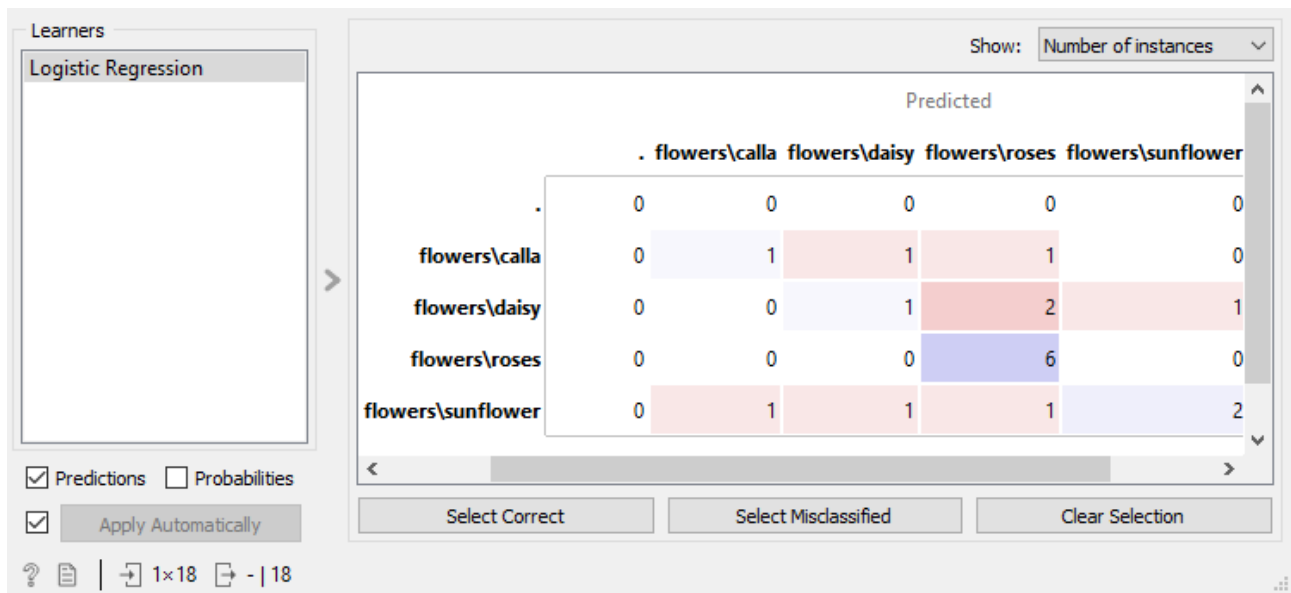
Model	Logistic ...
Logistic Regression	

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

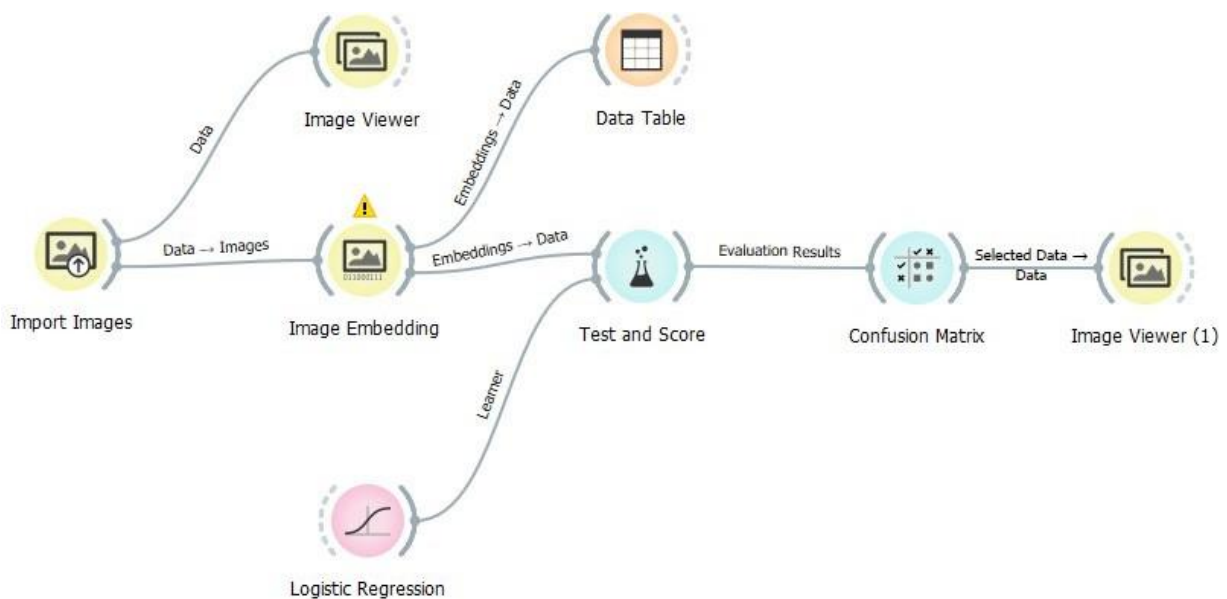
? | 18 | - | 18 | 1x18

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





- Drag and drop another Image Viewer widget to view the selected cell(s) from the Confusion Matrix.



Learners

Logistic Regression

Show: Number of instances

	flowers\calla	flowers\daisy	flowers\roses	flowers\sunflower
flowers\calla	0	1	1	1
flowers\daisy	0	0	1	2
flowers\roses	0	0	0	6
flowers\sunflower	0	1	1	1

☒ Predictions
 ☐ Probabilities

☒ Apply Automatically

Select Correct
 Select Misclassified
 Clear Selection

? | 1x18 | 6 | 18

Image Viewer (1) - Orange

Image Filename Attribute: image

Title Attribute: category

Image Size: [Slider]

☒ Send Automatically

flowers\roses flowers\roses flowers\roses flowers\roses flowers\roses

flowers\roses

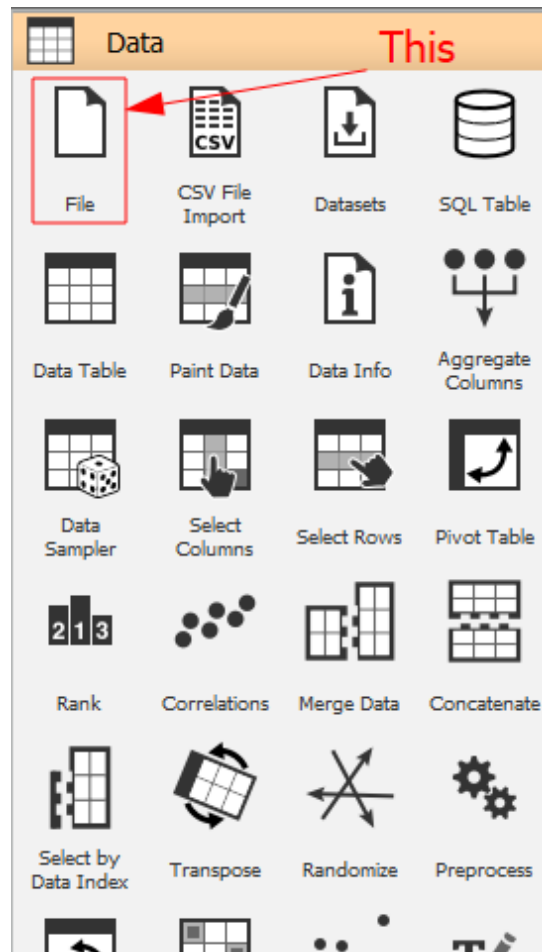
6 - 6

Practical 4

Aim: Hierarchical clustering using orange.

Steps:


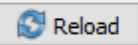
- Drag and drop a Filewidget from the Data section onto the workspace.





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

Source

☒ File:  

☐ URL:


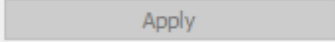
Info

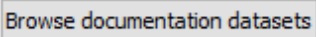
Iris flower dataset
Classical dataset with 150 instances of Iris setosa, Iris virginica and Iris versicolor.

150 instance(s)
4 feature(s) (no missing values)
Classification; categorical class with 3 values (no missing values)
0 meta attribute(s)

Columns (Double click to edit)

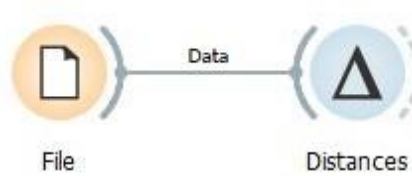
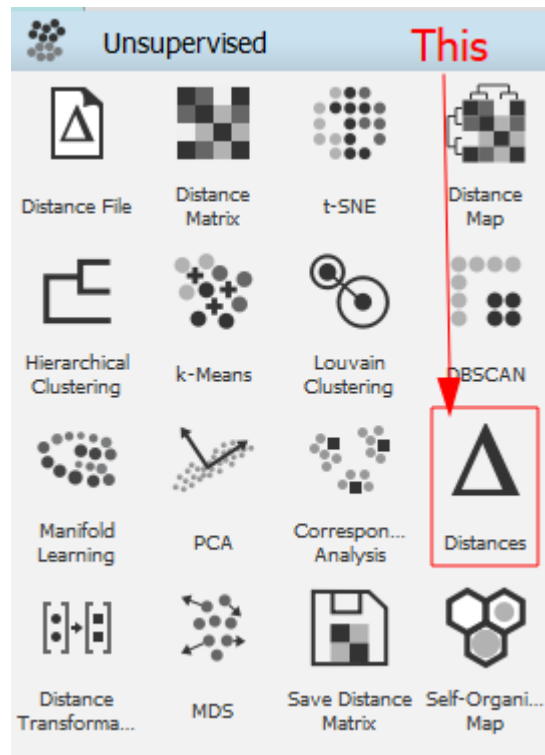
	Name	Type	Role	Values
1	sepal length	N numeric	feature	
2	sepal width	N numeric	feature	
3	petal length	N numeric	feature	
4	petal width	N numeric	feature	
5	iris	C categorical	target	Iris-setosa, Iris-versicolor, Iris-virginica

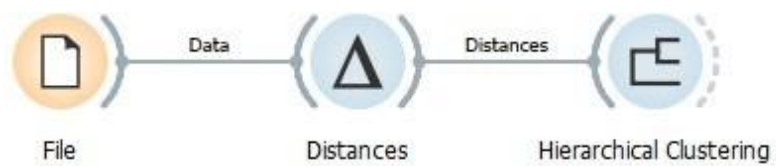
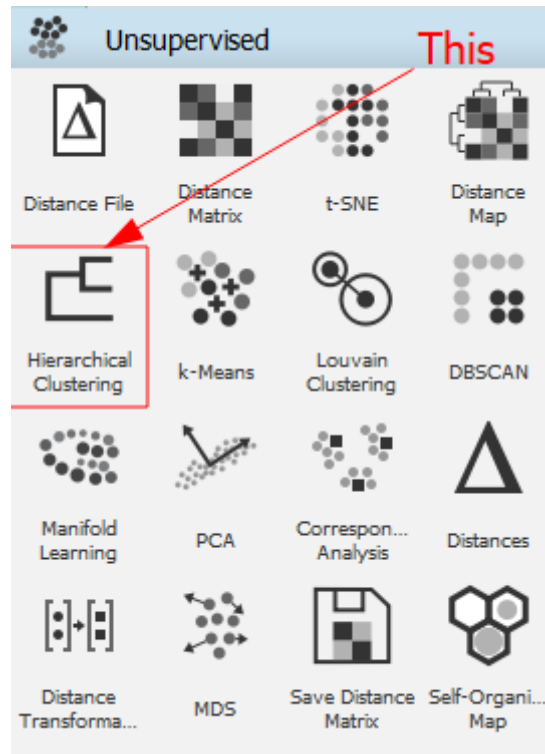


? | 150

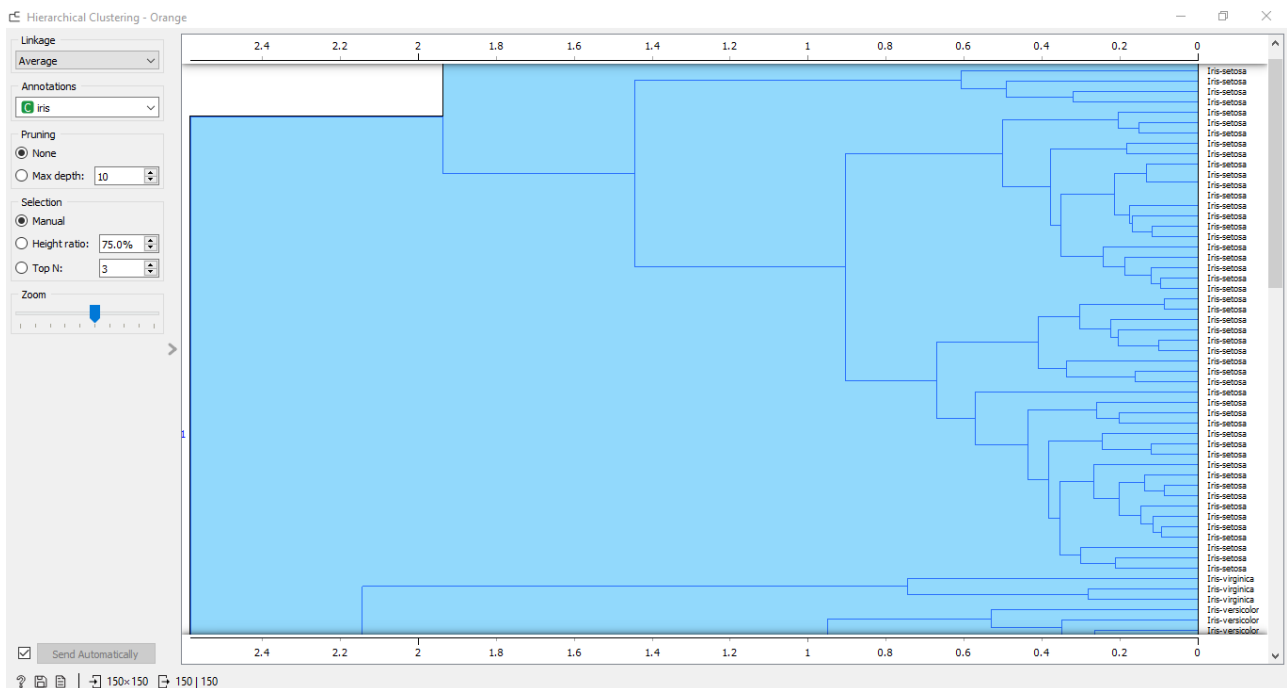
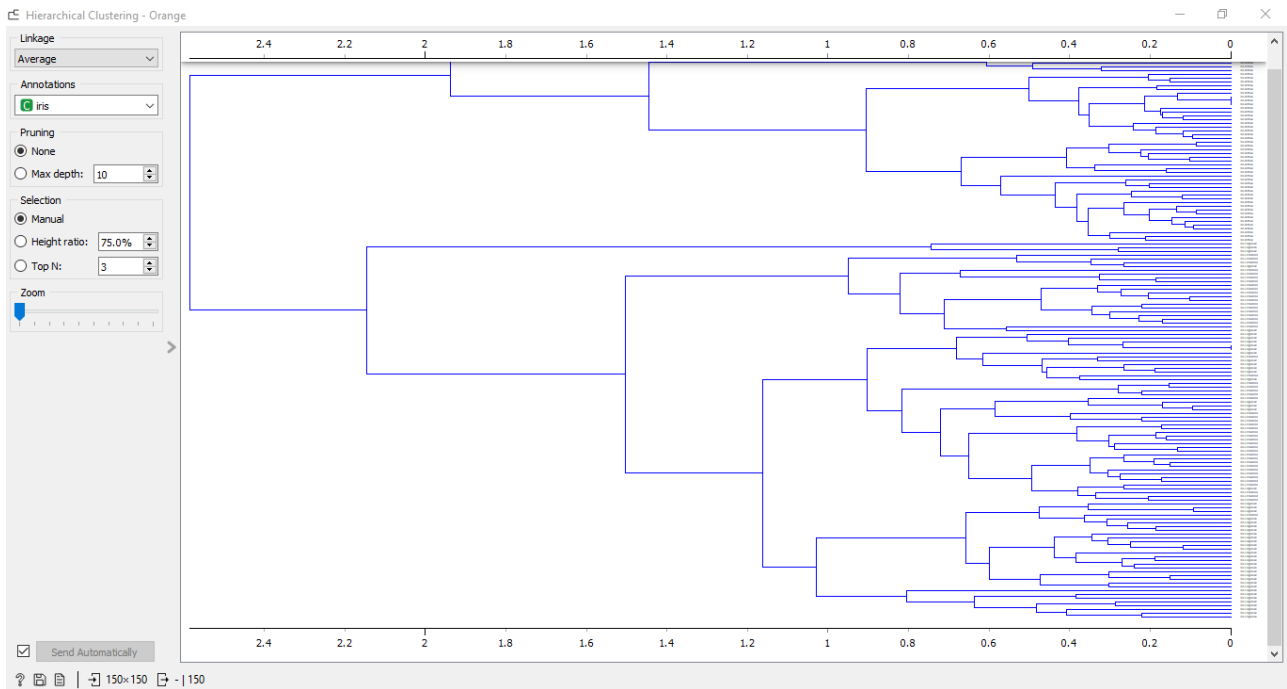
- Drag and drop the Distances widget from the Unsupervised section onto the workspace and connect it to the File widget.



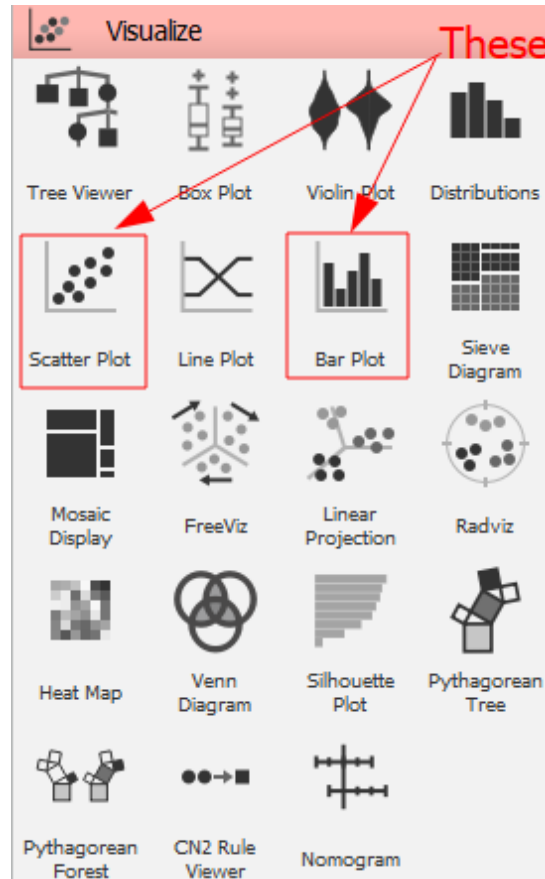
- Drag and drop a Hierarchical Clustering widget from the Unsupervised section to the workspace and connect it to the Distances widget.

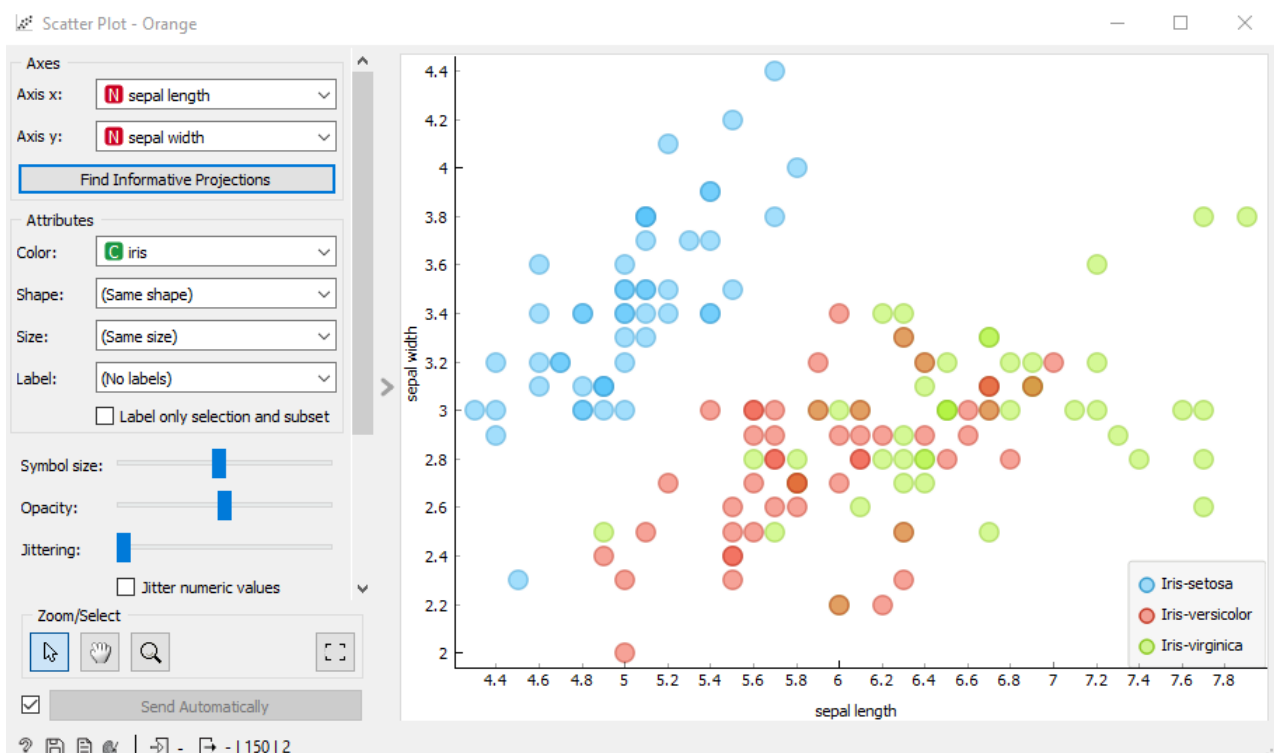
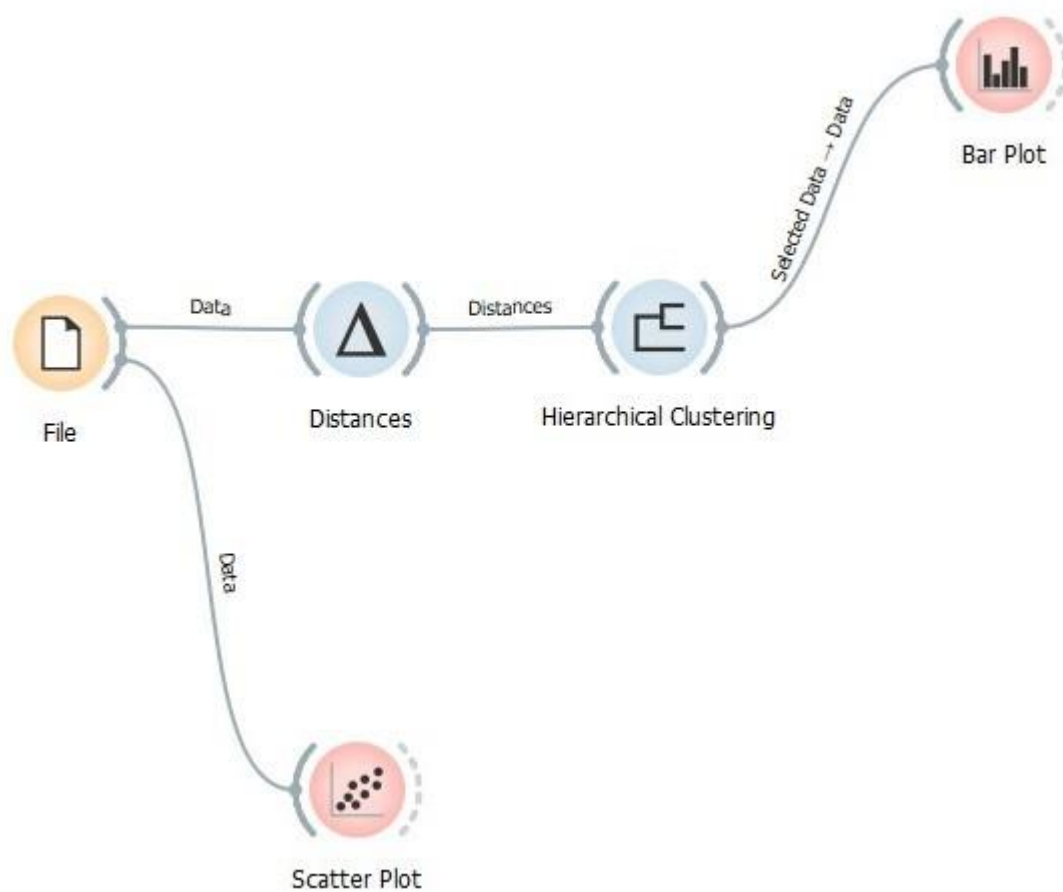


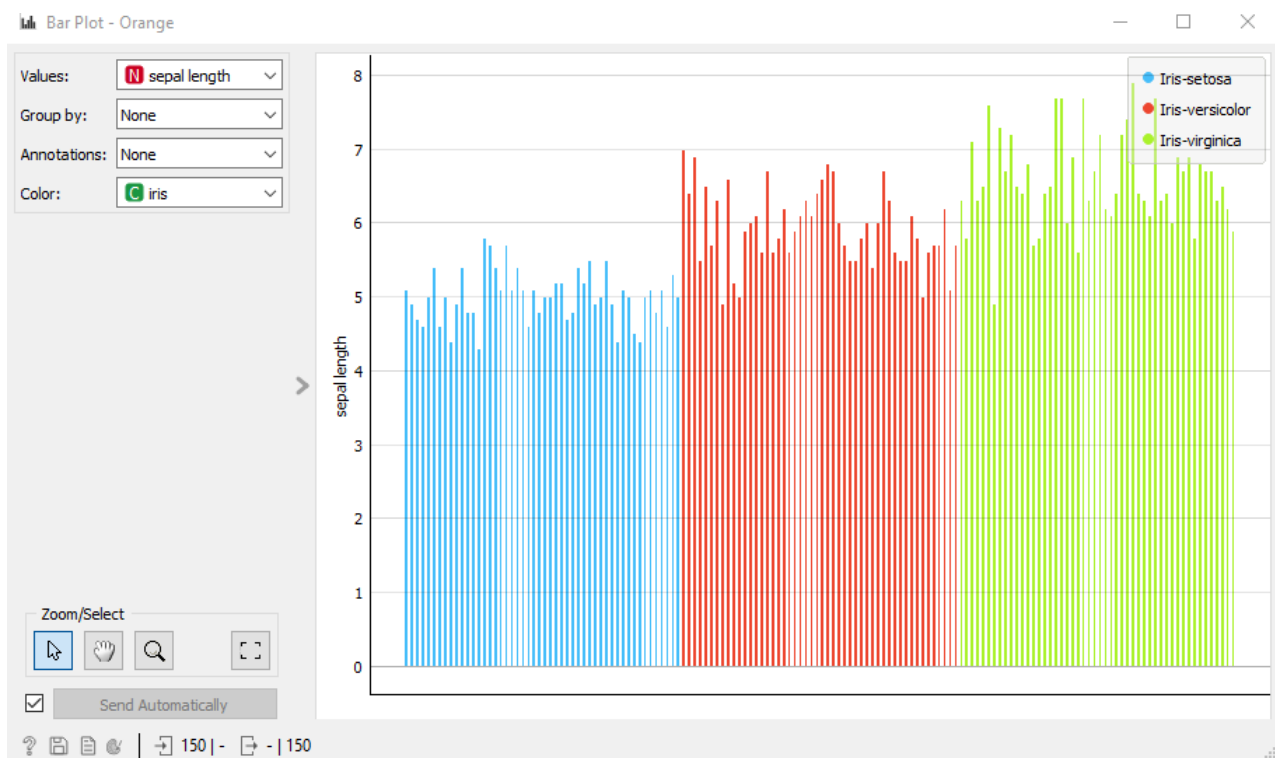
- Double click the Hierarchical Clustering widget to view the dendrogram. Select any sub-cluster as per convenience.



- Choose any visualization method of your choice. This example assumes a Bar Plot for the sub-cluster and a Scatter Plot for 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 Plot to the File widget.





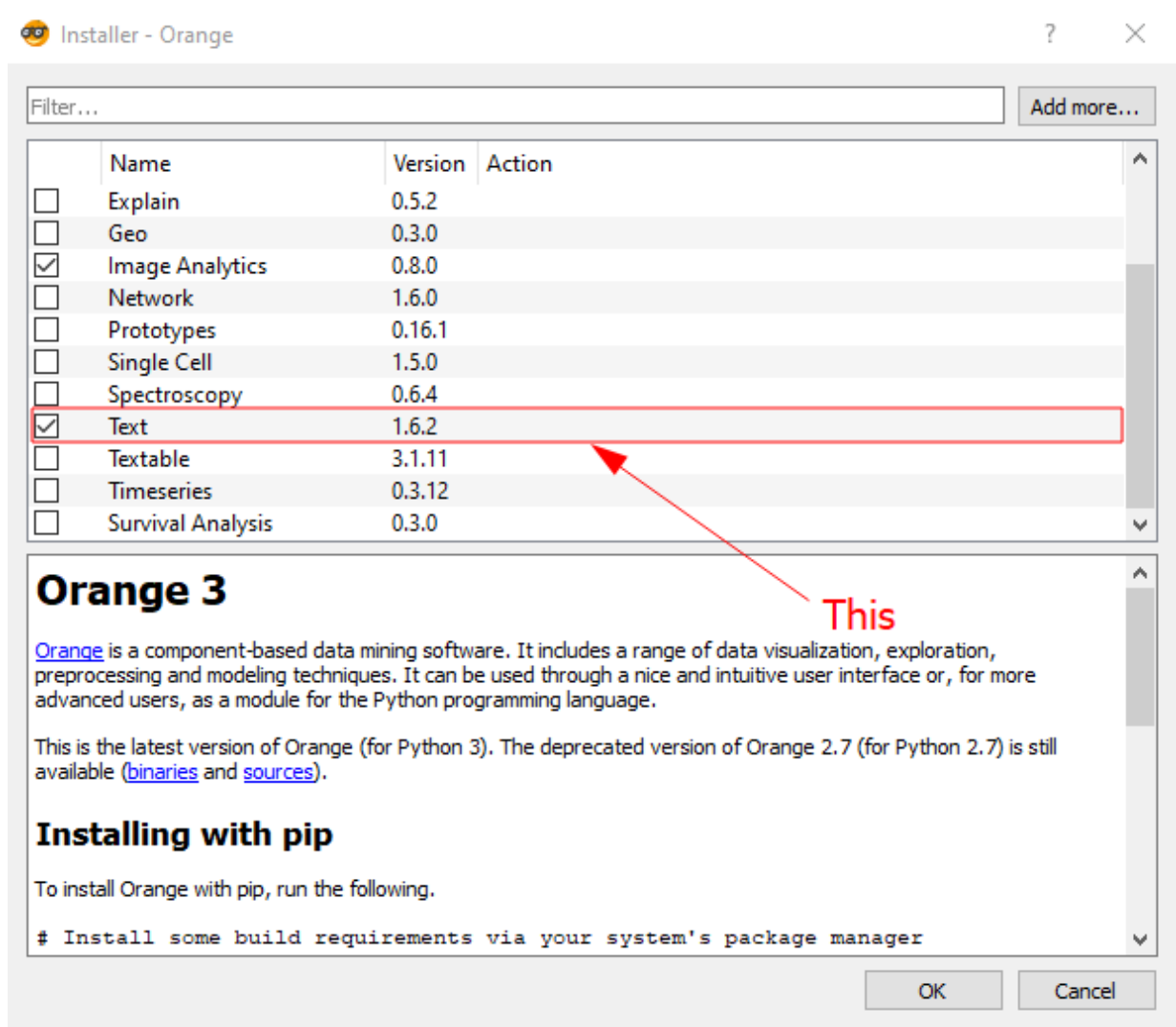


Practical 5

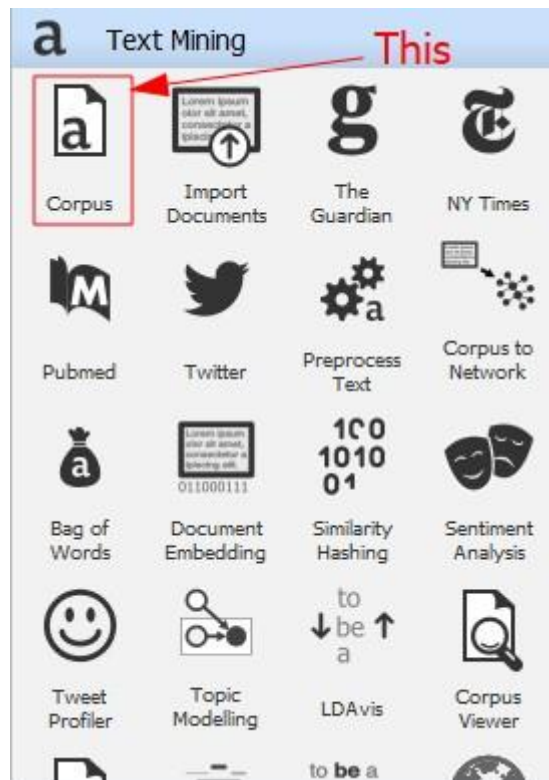
Aim: Text Preprocessing using orange.

Steps:

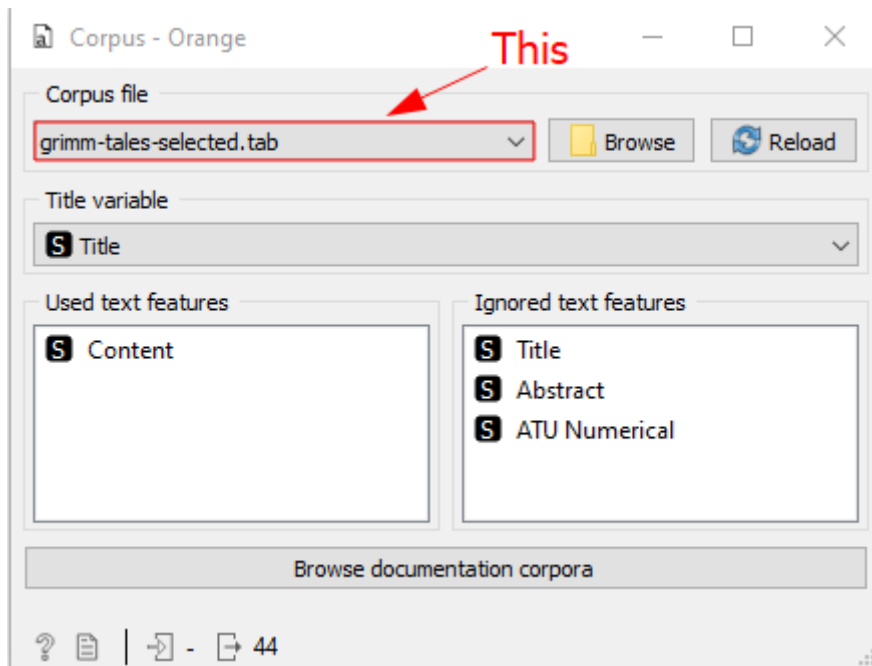
- Before starting, we need to install the **Text** add-on. Install it by navigating to **Options > Add ons...**. You will be prompted to restart Orange after the installation completes. If it is already installed, skip this step.



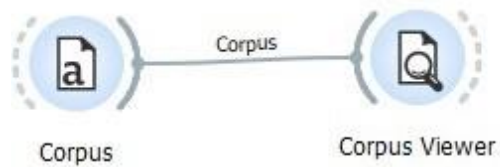
- Drag and drop a Corpuswidget to the workspace. It can be found in the newly added Text Mining section.



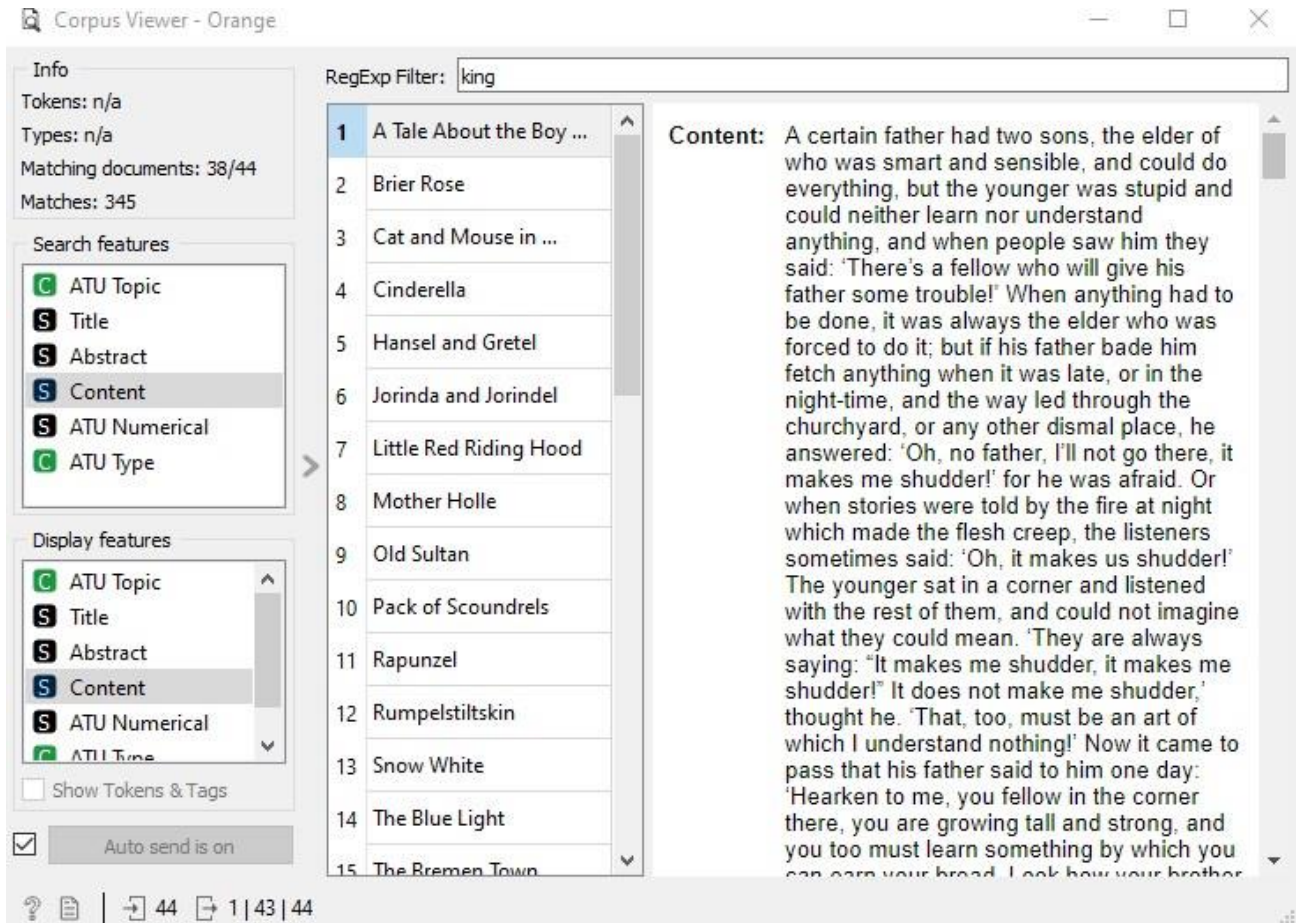
- Double click the Corpus widget and select the **grimm-tales-selected.tab** corpus file.



- 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.

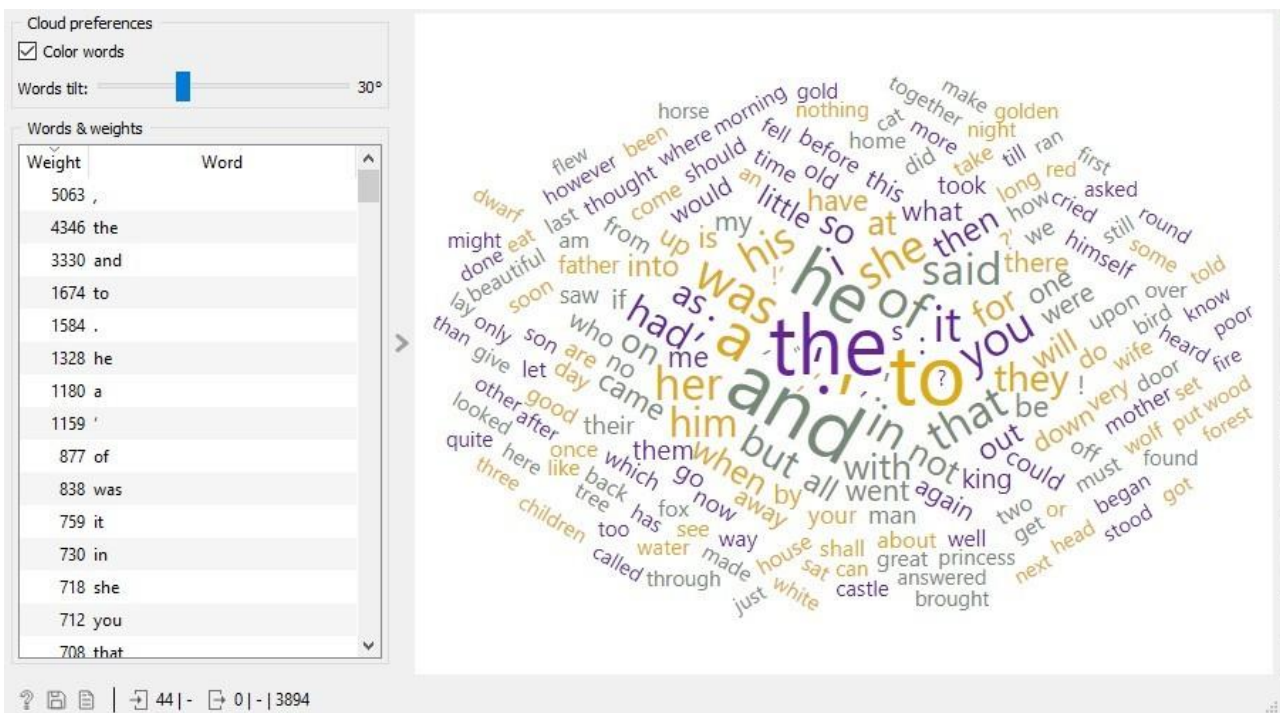
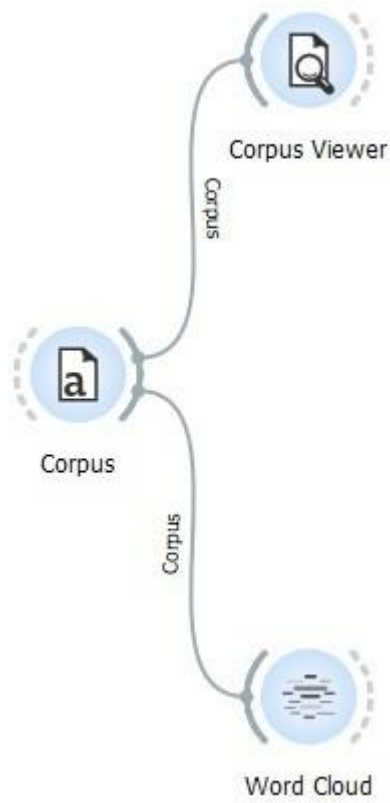


- Double-clicking the Corpus Viewer allows us to peek through the corpus and also allows us to filter text.

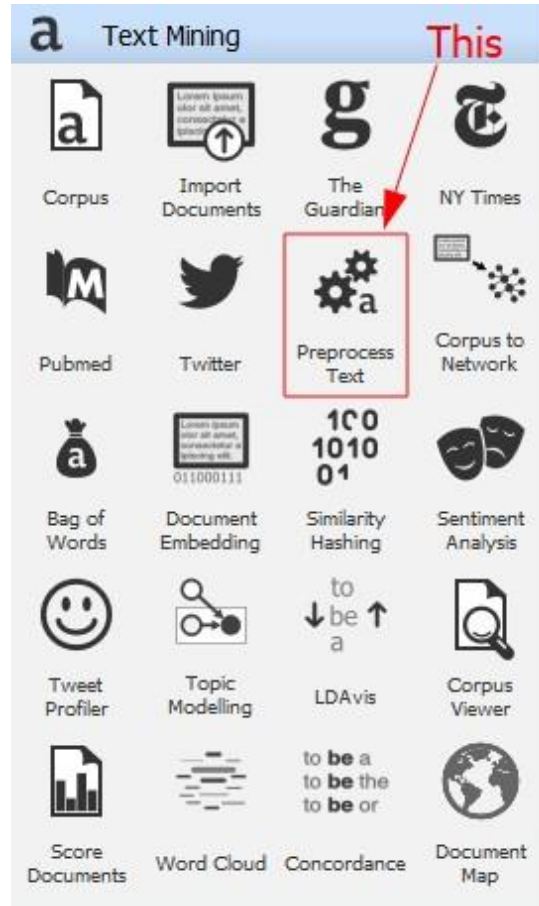


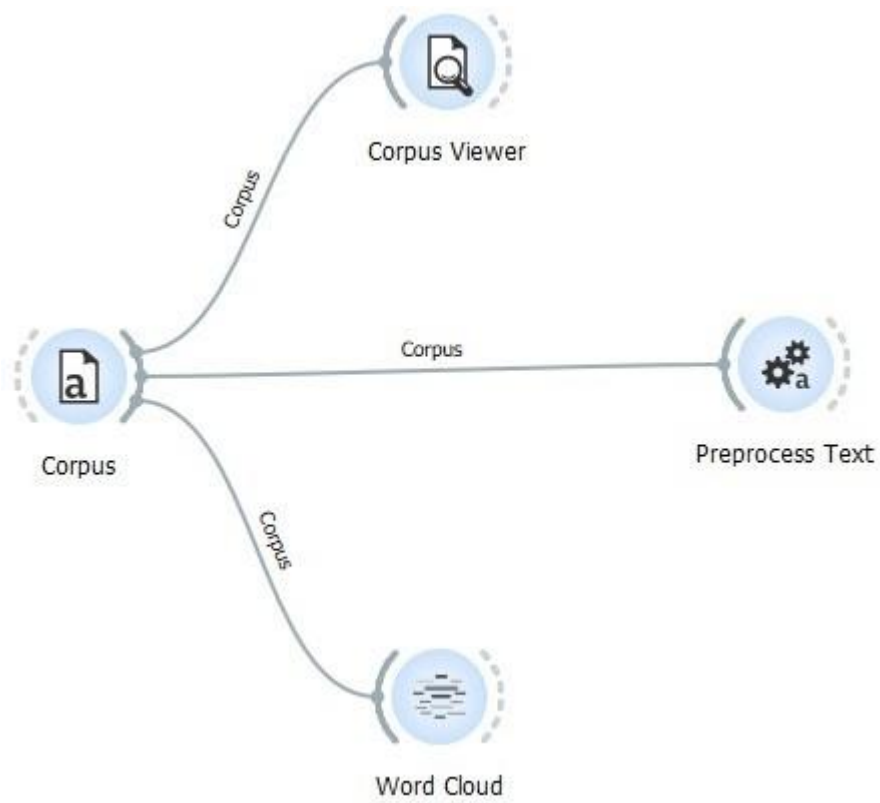
- Another method of visualizing data is though Word Clouds. Drag and drop a Word Cloud widget from the Text Mining section and connect it to the Corpuswidget.



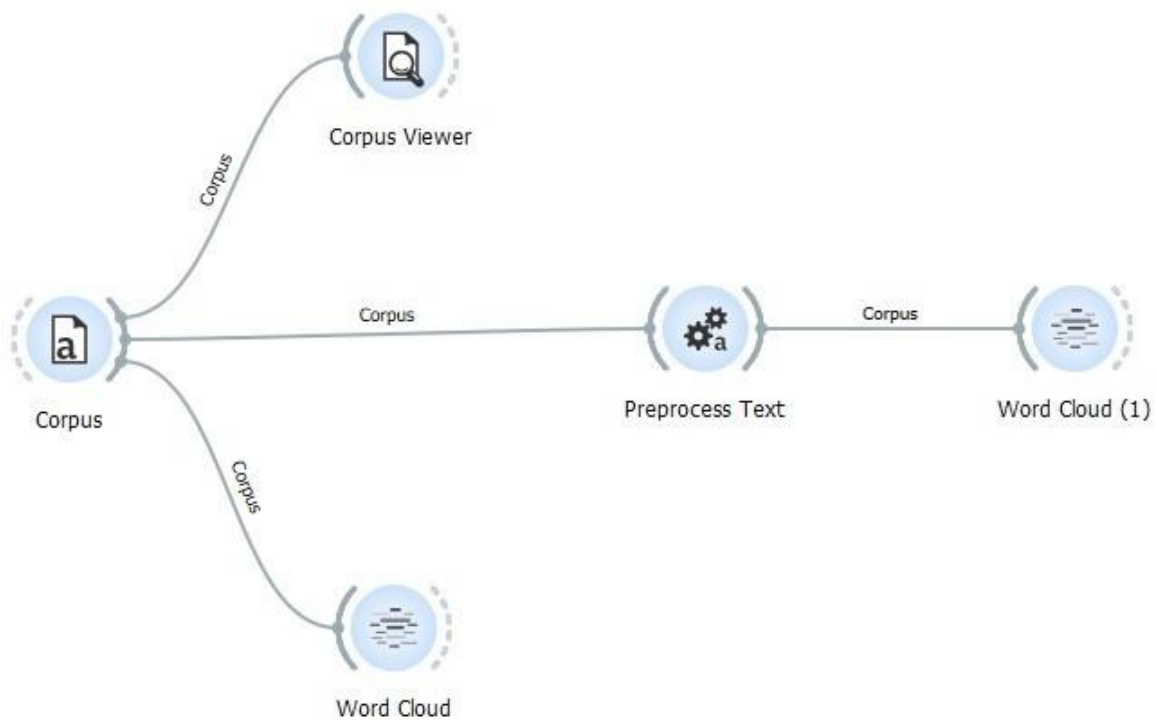


- We see that a lot of punctuation marks and uninformative words have made their way into the Word Cloud. In order to eliminate this, we have to use the Preprocess Text widget. Drag and drop a Preprocess Text widget from the Text Mining section onto the workspace and connect it to the Corpus widget.

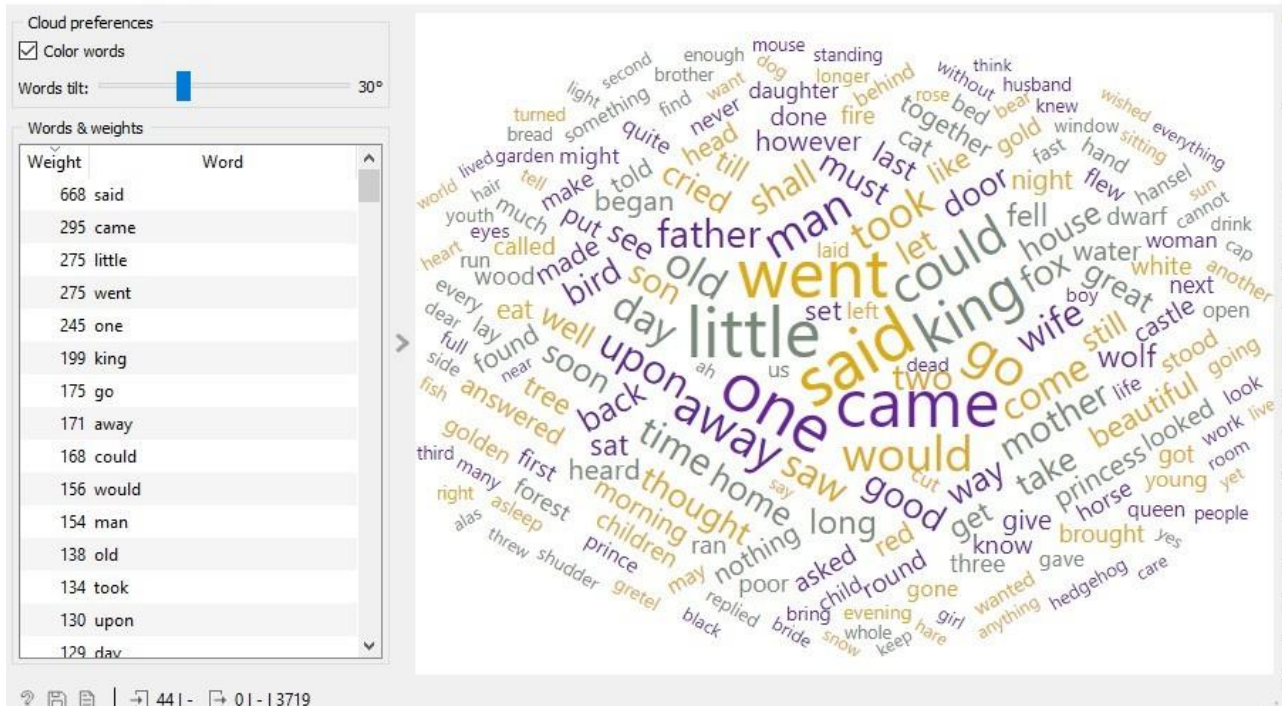




➤ We now add another Word Cloud widget to see our updated cloud.



Word Cloud (1) - Orange



Practical 6

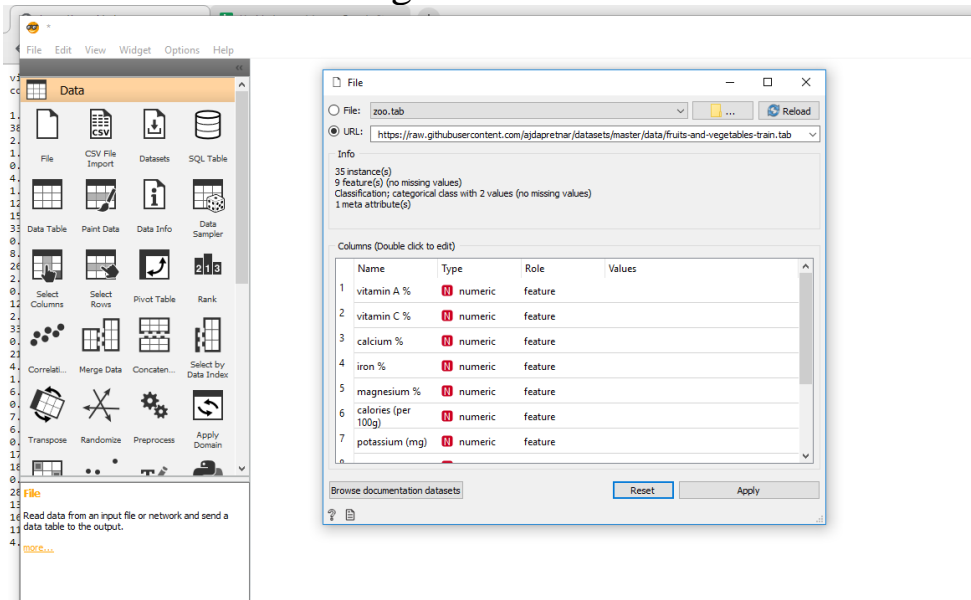
Aim: To predict whether the given dataset is of fruit or vegetable using orange.

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.

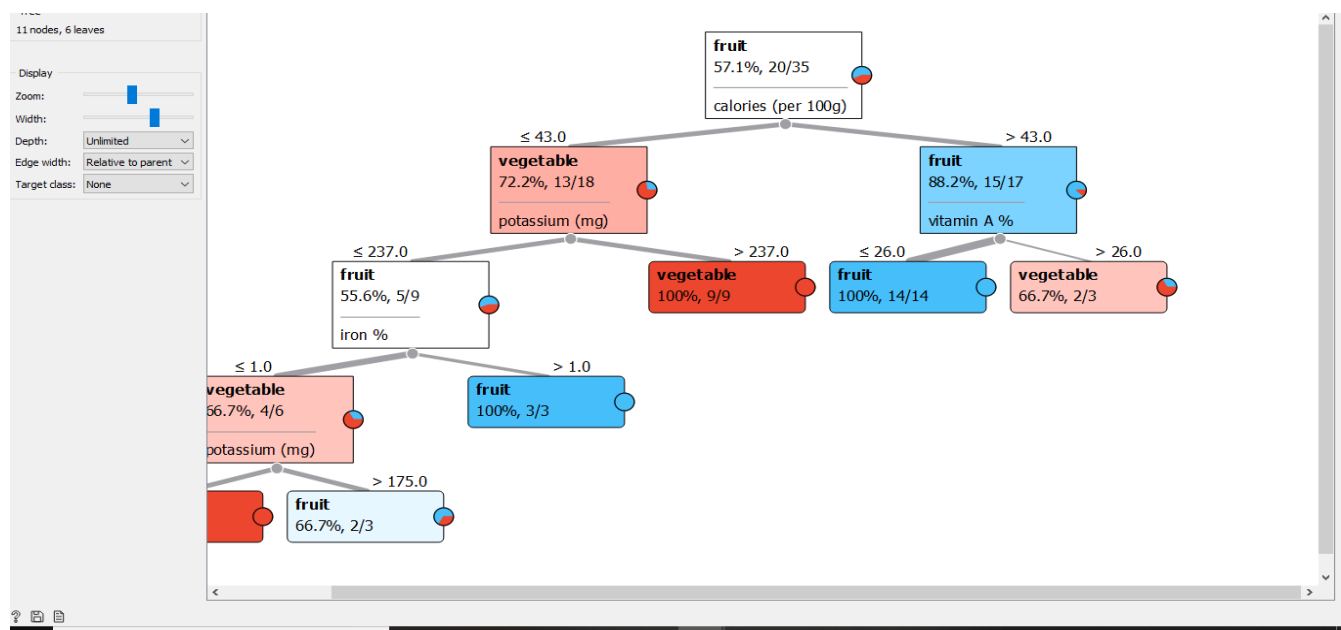


Data visualization:

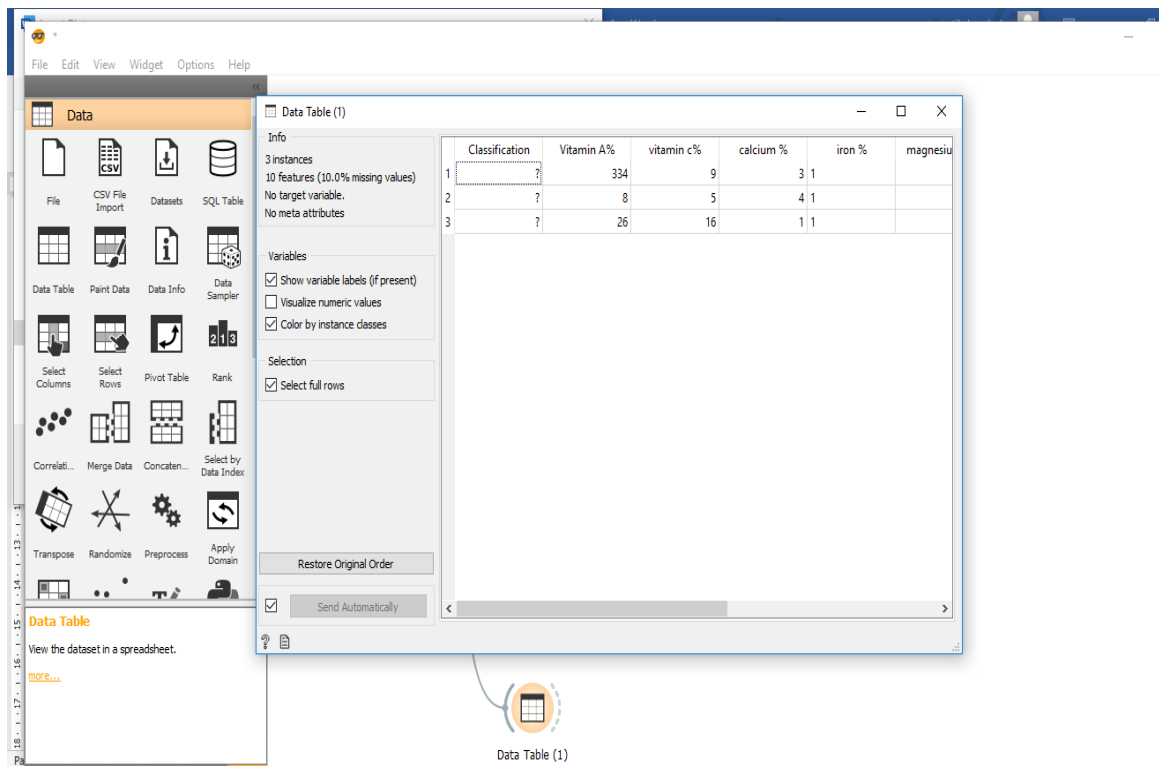
We visualize the data by using data table and classification tree.

The screenshot shows the Orange3 Data Table widget. The left sidebar contains various data processing widgets. The main window displays the 'Data Table' widget with the following data:

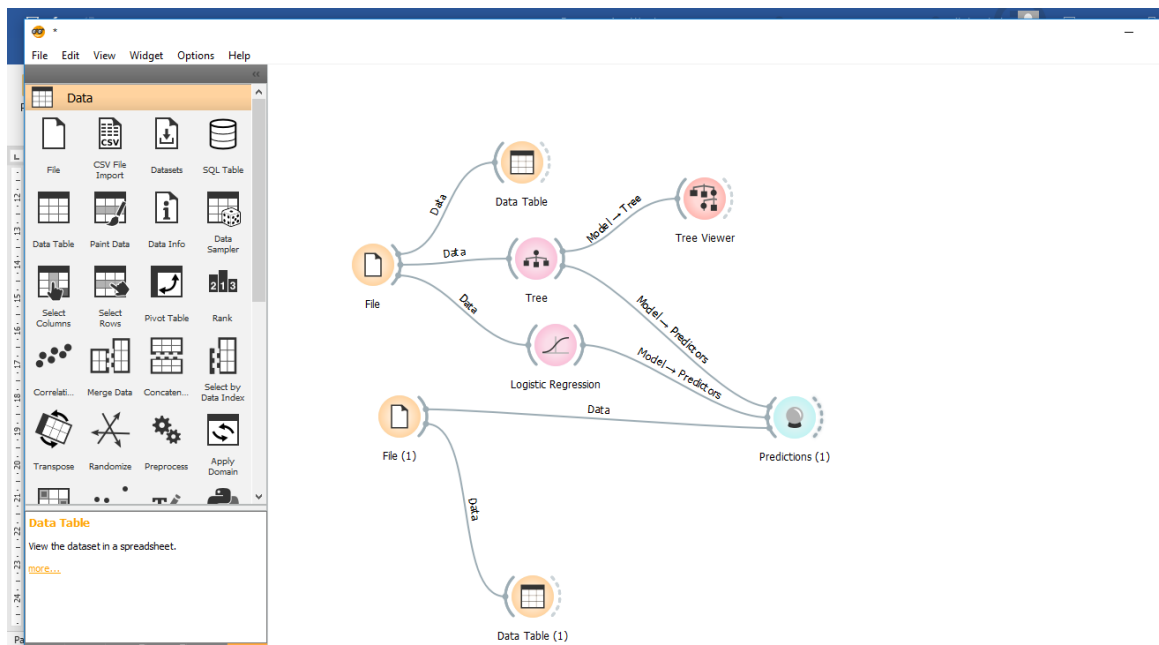
	classification	name	vitamin A %	vitamin C %	calcium %	iron %
1	fruit	apple	1.0	7.0	0.0	0.0
2	fruit	apricot	38.0	16.0	1.0	0.0
3	fruit	avocado	2.0	16.0	1.0	0.0
4	fruit	banana	1.0	14.0	0.0	0.0
5	vegetable	beetroot	0.0	8.0	1.0	0.0
6	fruit	blackberry	4.0	35.0	2.0	0.0
7	fruit	blueberry	1.0	16.0	0.0	0.0
8	vegetable	broccoli	12.0	148.0	4.0	0.0
9	vegetable	brussels sprouts	15.0	141.0	4.0	0.0
10	vegetable	carrot	334.0	9.0	3.0	0.0
11	vegetable	cauliflower	0.0	80.0	2.0	0.0
12	vegetable	celery	8.0	5.0	4.0	0.0
13	fruit	cherry	26.0	16.0	1.0	0.0
14	vegetable	cucumber	2.0	4.0	1.0	0.0
15	vegetable	eggplant	0.0	3.0	0.0	0.0
16	fruit	elderberry	12.0	60.0	3.0	0.0
17	fruit	grape	2.0	6.0	1.0	0.0
18	vegetable	leek	33.0	20.0	5.0	0.0
19	fruit	lemon	0.0	88.0	2.0	0.0
20	fruit	mango	21.0	60.0	1.0	0.0
21	fruit	orange	4.0	88.0	4.0	0.0
22	fruit	pineapple	1.0	79.0	1.0	0.0



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 screenshot shows the Orange3 data mining software interface. On the left is a 'Data' widget panel with various data sources and processing widgets. The main workspace contains a workflow: 'File (1)' connects to 'Data Table (1)', which connects to 'Predictions (1)'. The 'Predictions (1)' window is open, displaying a table of model predictions for 'fruit' and 'vegetable' classes. The table includes columns for the model type (Tree, Logistic Regression), the predicted class, and several numerical features (Classification, Vitamin A%, vitamin c%).

	Tree	Logistic Regression	Classification	Vitamin A%	vitamin c%
1 fruit	fruit	fruit	?	334	9
2 fruit	fruit	fruit	?	8	5
3 fruit	fruit	fruit	?	26	16

The given prediction is again checked by logistic regression.

Practical 7

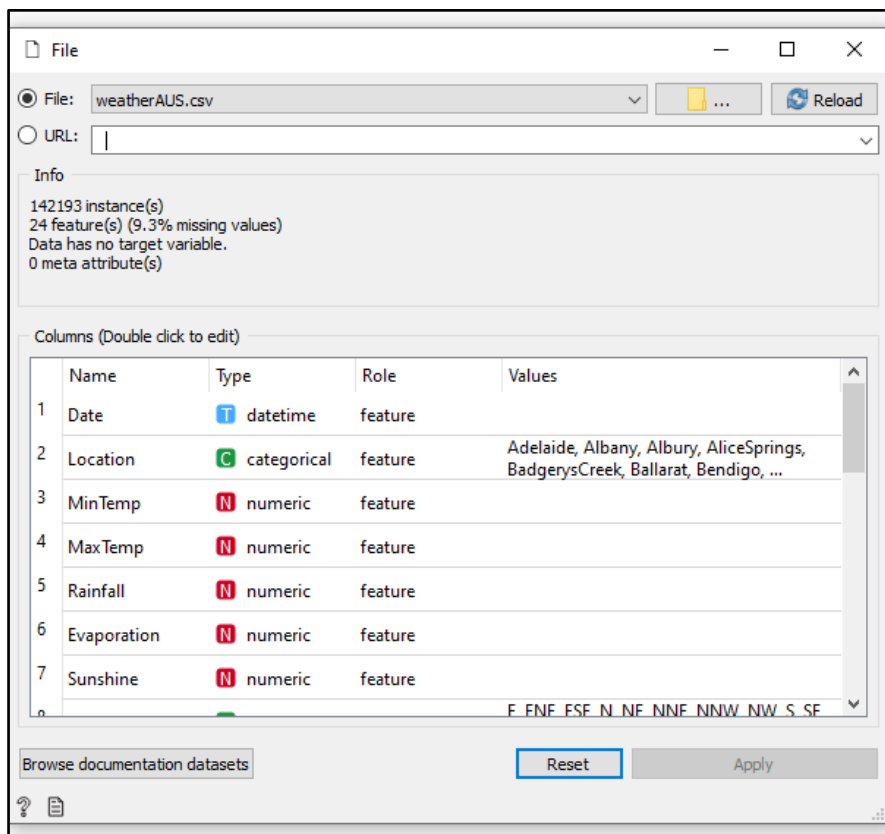
Aim: Predict whether using orange.

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:

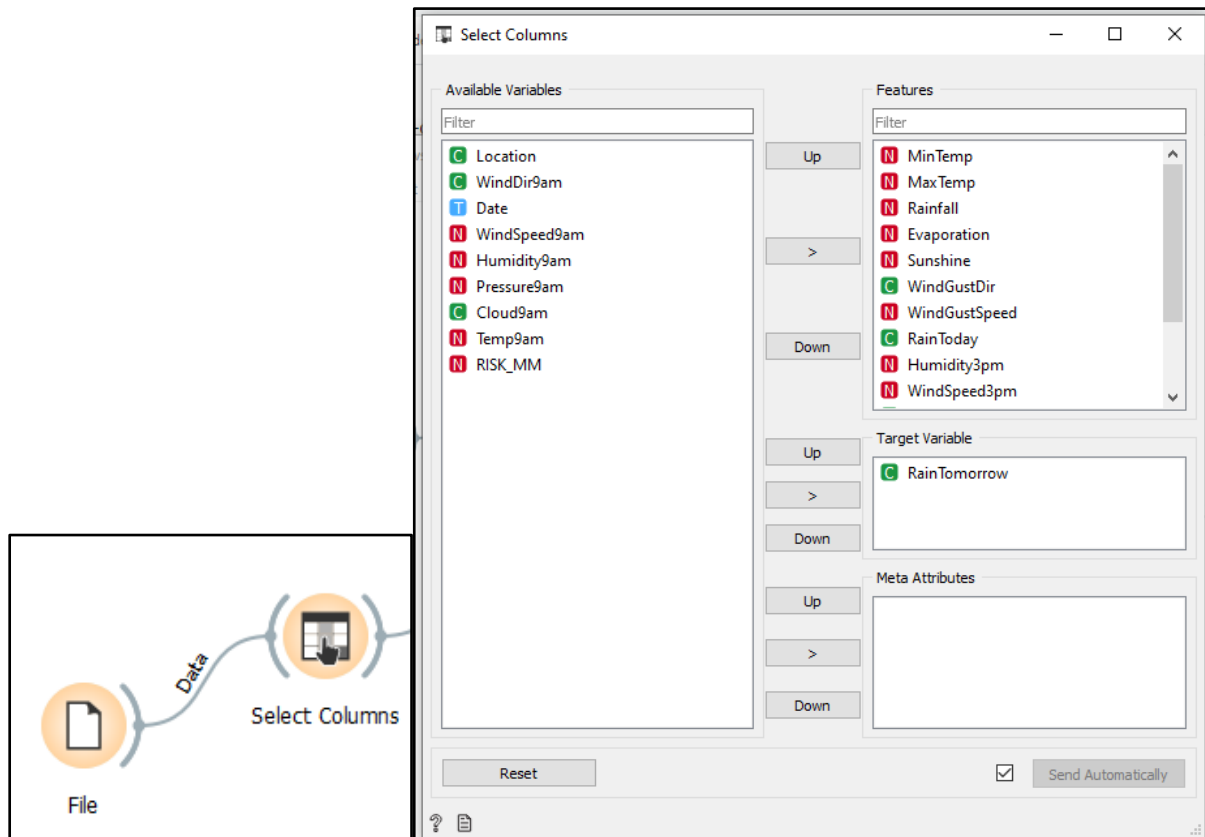
We have downloaded the dataset (weather dataset) of Australian weather stations which contains about 10 years of daily weather observations from numerous Australian weather stations.



Data Cleaning:

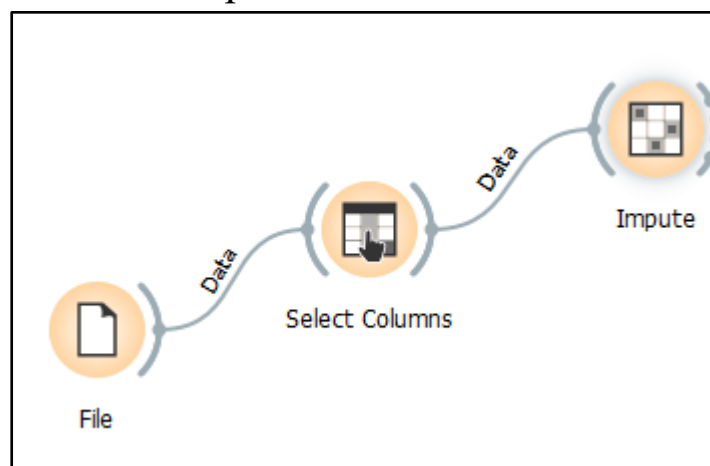
In data cleaning, we will select the required column and remove the rows with missing values as shown below:

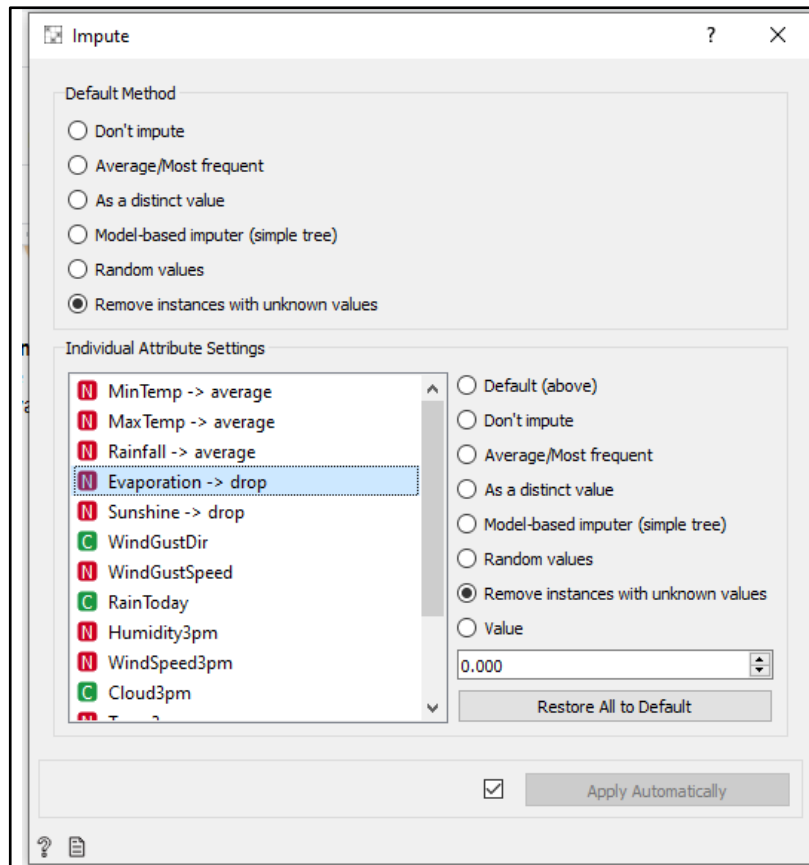
Column Selection



Removing rows with missing values

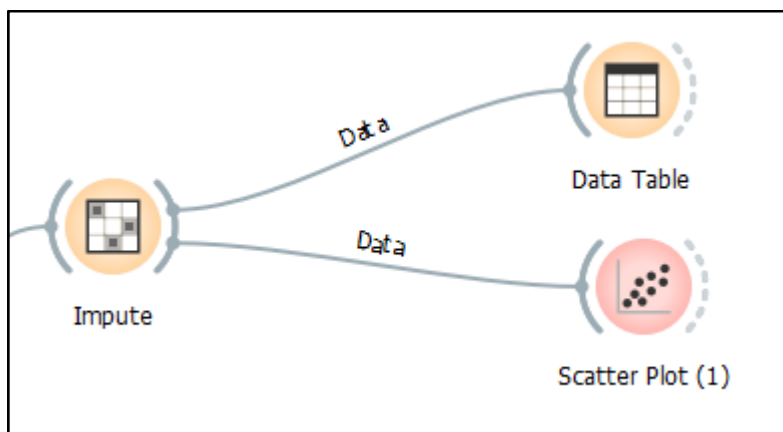
We will use the impute function to remove rows with missing values for columns “Sunshine” and “Evaporation”



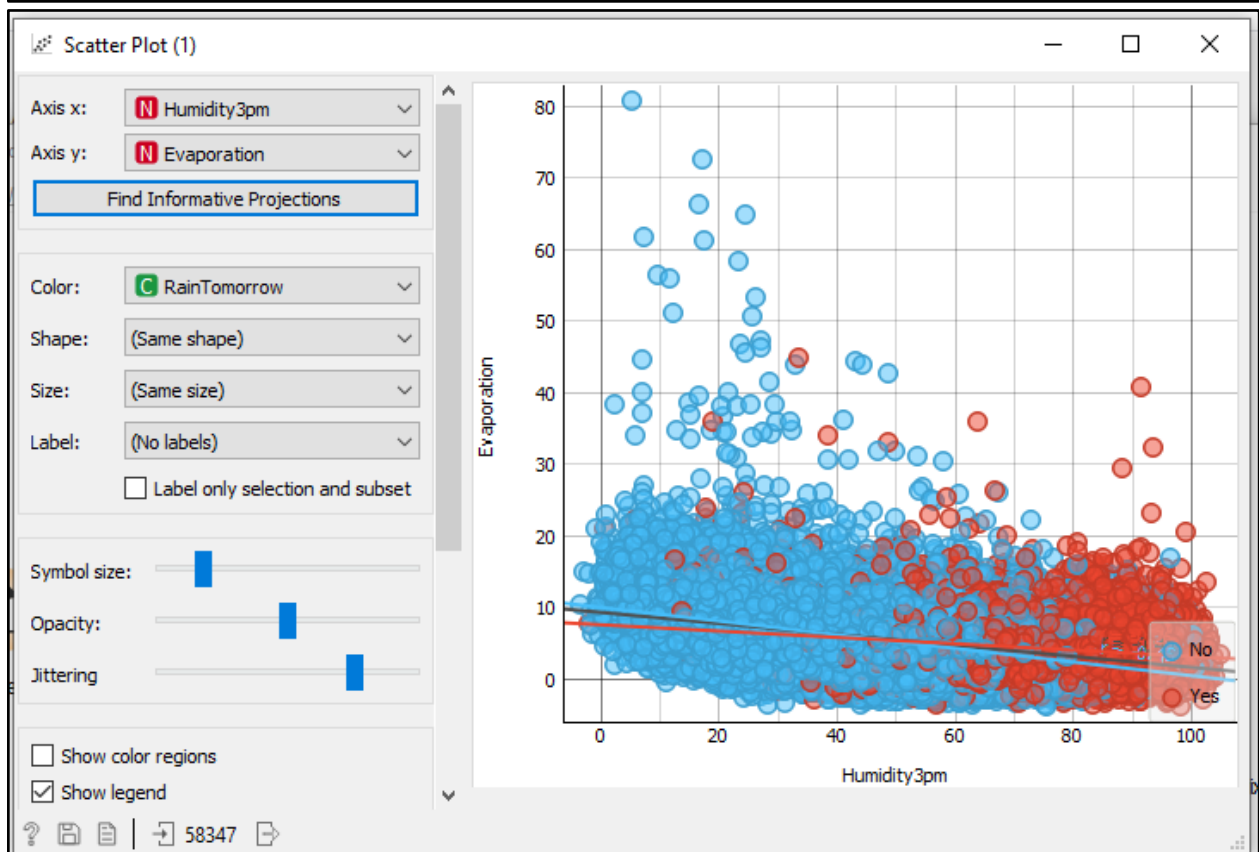


Data Visualization

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.



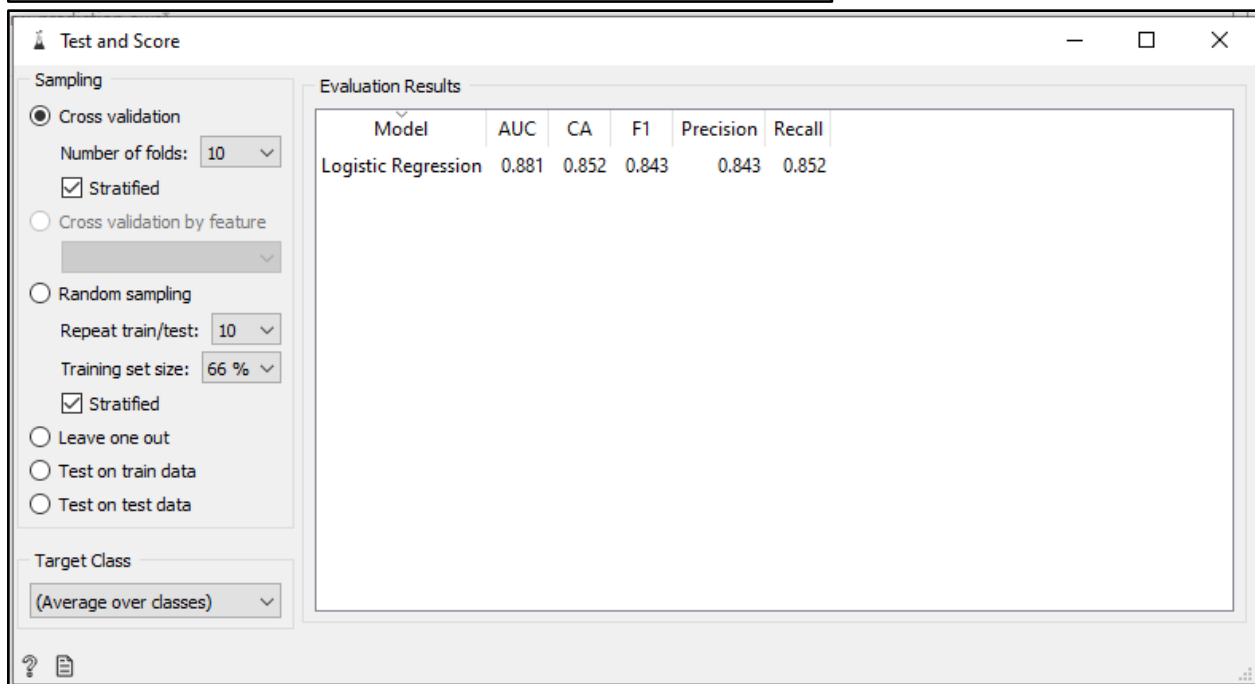
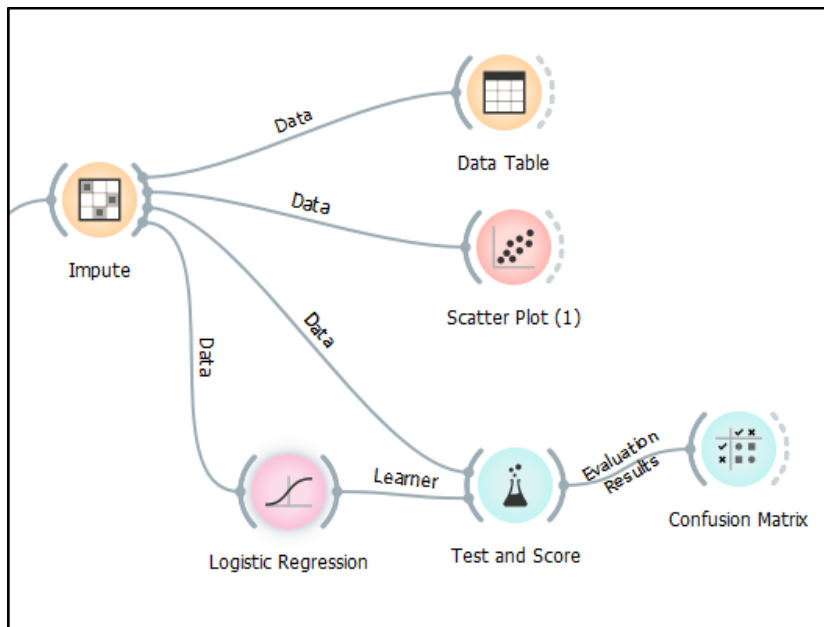
Data Table							
<div>Info</div> <div>58347 instances (no missing values)</div> <div>14 features (no missing values)</div> <div>Discrete class with 2 values (no missing values)</div> <div>No meta attributes</div> <div>Variables</div> <div><input checked="" type="checkbox"/> Show variable labels (if present)</div> <div><input checked="" type="checkbox"/> Visualize numeric values</div> <div><input checked="" type="checkbox"/> Color by instance classes</div> <div>Selection</div> <div><input checked="" type="checkbox"/> Select full rows</div> <div>Restore Original Order</div> <div><input checked="" type="checkbox"/> Send Automatically</div>							
	RainTomorrow	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	
1	No	17.9	35.2	0.0	12.0	12.3	
2	No	18.4	28.9	0.0	14.8	13.0	
3	No	15.5	34.1	0.0	12.6	13.3	
4	No	19.4	37.6	0.0	10.8	10.6	
5	No	21.9	38.4	0.0	11.4	12.2	
6	No	24.2	41.0	0.0	11.2	8.4	
7	No	27.1	36.1	0.0	13.0	0.0	
8	No	23.3	34.0	0.0	9.8	12.6	
9	No	16.1	34.2	0.0	14.6	13.2	
10	No	19.0	35.5	0.0	12.0	12.3	
11	No	19.7	35.5	0.0	11.0	12.7	
12	No	20.9	37.8	0.0	12.8	13.2	
13	No	23.9	39.1	0.0	13.8	12.1	
14	No	24.9	41.2	0.0	14.8	13.0	
15	No	25.2	40.5	0.0	16.4	10.3	
16	No	21.6	34.2	0.0	17.4	13.1	
17	No	18.4	31.8	0.0	16.0	12.9	

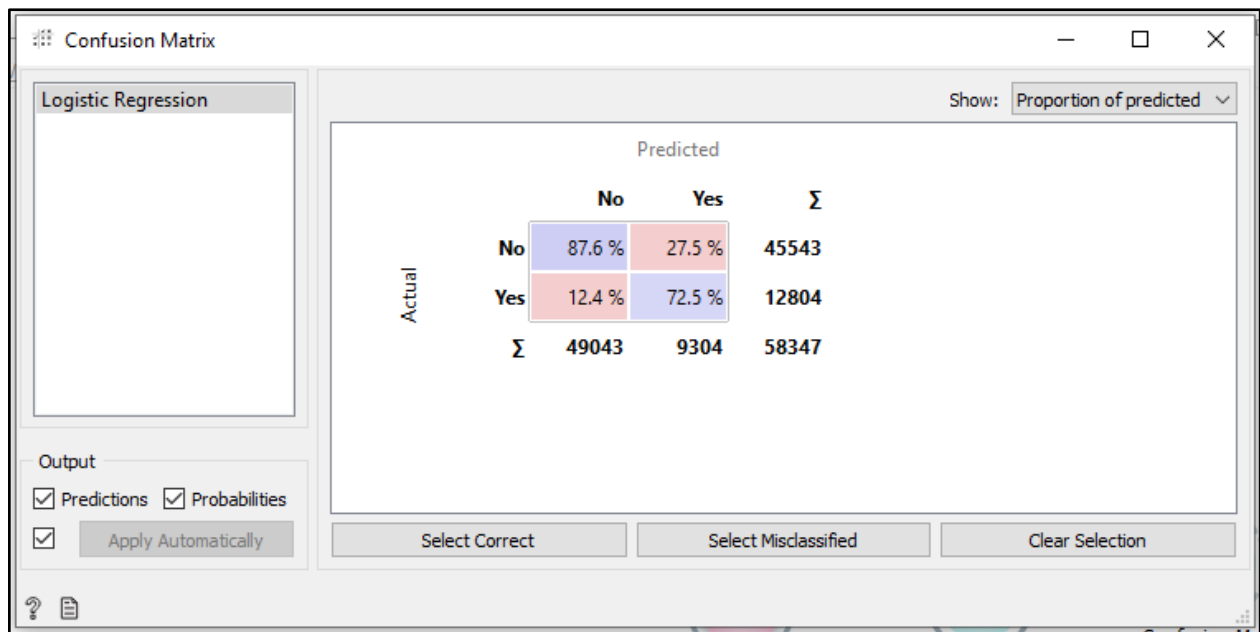


In the above scatter plot we see, as the humidity level increases and evaporation rates are low the chances of rainfall increases (red dots)

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:

```
[Running] cd "c:\Users\USER\Downloads\" && javac AMSA.java && java AMSA  
c=3 d=2 a=2  
Expected value for c is :: 75  
Expected value for d is :: 45  
Expected value for a is :: 45  
Second moment is:55  
  
[Done] exited with code=0 in 18.944 seconds
```

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:

```
> require("tm")
>
> kshingle<-function(){
+   k<- as.integer(readline("Enter a value for k - 1"))
+
+   u1<- readLines("BIBD1.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()
Enter a value for k - 12
character(0)
[1] "Lor"
[1] "ore"
[1] "rem"
[1] "em "
[1] "m i"
[1] " ip"
[1] "ips"
[1] "psu"
[1] "sum"
[1] "um "
[1] "m d"
[1] " do"
[1] "dol"
[1] "olo"
[1] "lor"
[1] "or "
[1] "r s"

[1] " si"
[1] "sit"
[1] "it "
[1] "t a"
[1] " am"
[1] "ame"
[1] "met"
[1] "et,"
[1] "t, "
[1] ", c"
[1] " co"
[1] "con"
[1] "ons"
[1] "nse"
[1] "sec"
[1] "ect"
[1] "cte"
[1] "tet"
[1] "etu"
[1] "tur"
[1] "ur "
[1] "r a"
[1] " ad"
[1] "adi"
[1] "dip"
[1] "ipi"
[1] "pis"
[1] "isc"
[1] "sci"
[1] "cin"
[1] "ing"
[1] "ng "
[1] "g e"
[1] " el"
[1] "eli"
[1] "lit"
[1] "it,"
[1] "t, "

[1] ", s"
[1] " se"
[1] "sed"
[1] "ed "
[1] "d d"
[1] " do"
[1] "do "
[1] "o e"
[1] " ei"
[1] "eiu"
[1] "ius"
[1] "usm"
[1] "smo"
[1] "mod"
[1] "od "
[1] "d t"
[1] " te"
[1] "tem"
[1] "emp"
[1] "mpo"
[1] "por"
[1] "or "
[1] "r i"
[1] " in"
[1] "inc"
[1] "nci"
[1] "cid"
[1] "idi"
[1] "did"
[1] "idu"
[1] "dun"
[1] "unt"
[1] "nt "
[1] "t u"
[1] " ut"
[1] "ut "
[1] "t l"
[1] " la"
```

[1] "lab"	[1] " ad"	[1] "ati"	[1] "x e"	[1] "e d"	[1] "vel"
[1] "abo"	[1] "ad "	[1] "tio"	[1] " ea"	[1] " do"	[1] "eli"
[1] "bor"	[1] "d m"	[1] "ion"	[1] "ea "	[1] "dol"	[1] "lit"
[1] "ore"	[1] " mi"	[1] "on "	[1] "a c"	[1] "olo"	[1] "it "
[1] "re "	[1] "min"	[1] "n u"	[1] " co"	[1] "lor"	[1] "t e"
[1] "e e"	[1] "ini"	[1] " ul"	[1] "com"	[1] "or "	[1] " es"
[1] " et"	[1] "nim"	[1] "ull"	[1] "omm"	[1] "r i"	[1] "ess"
[1] "et "	[1] "im "	[1] "lla"	[1] "mmo"	[1] " in"	[1] "sse"
[1] "t d"	[1] "m v"	[1] "lam"	[1] "mod"	[1] "in "	[1] "se "
[1] " do"	[1] " ve"	[1] "amc"	[1] "odo"	[1] "n r"	[1] "e c"
[1] "dol"	[1] "ven"	[1] "mco"	[1] "do "	[1] " re"	[1] " ci"
[1] "olo"	[1] "eni"	[1] "co "	[1] "o c"	[1] "rep"	[1] "cil"
[1] "lor"	[1] "nia"	[1] "o l"	[1] " co"	[1] "epr"	[1] "ill"
[1] "ore"	[1] "iam"	[1] " la"	[1] "con"	[1] "pre"	[1] "llu"
[1] "re "	[1] "am,"	[1] "lab"	[1] "ons"	[1] "reh"	[1] "lum"
[1] "e m"	[1] "m, "	[1] "abo"	[1] "nse"	[1] "ehe"	[1] "um "
[1] " ma"	[1] ", q"	[1] "bor"	[1] "seq"	[1] "hen"	[1] "m d"
[1] "mag"	[1] " qu"	[1] "ori"	[1] "equ"	[1] "end"	[1] " do"
[1] "agn"	[1] "qui"	[1] "ris"	[1] "qua"	[1] "nde"	[1] "dol"
[1] "gna"	[1] "uis"	[1] "is "	[1] "uat"	[1] "der"	[1] "olo"
[1] "na "	[1] "is "	[1] "s n"	[1] "at."	[1] "eri"	[1] "lor"
[1] "a a"	[1] "s n"	[1] " ni"	[1] "t. "	[1] "rit"	[1] "ore"
[1] " al"	[1] " no"	[1] "nis"	[1] ". D"	[1] "it "	[1] "re "
[1] "ali"	[1] "nos"	[1] "isi"	[1] " Du"	[1] "t i"	[1] "e e"
[1] "liq"	[1] "ost"	[1] "si "	[1] "Dui"	[1] " in"	[1] " eu"
[1] "iqu"	[1] "str"	[1] "i u"	[1] "uis"	[1] "in "	[1] "eu "
[1] "qua"	[1] "tru"	[1] " ut"	[1] "is "	[1] "n v"	[1] "u f"
[1] "ua."	[1] "rud"	[1] "ut "	[1] "s a"	[1] " vo"	[1] " fu"
[1] "a. "	[1] "ud "	[1] "t a"	[1] " au"	[1] "vol"	[1] "fug"
[1] ". U"	[1] "d e"	[1] " al"	[1] "aut"	[1] "olu"	[1] "ugi"
[1] " ut"	[1] " ex"	[1] "ali"	[1] "ute"	[1] "lup"	[1] "gia"
[1] "ut "	[1] "exe"	[1] "liq"	[1] "te "	[1] "upt"	[1] "iat"
[1] "t e"	[1] "xer"	[1] "iqu"	[1] "e i"	[1] "pta"	[1] "at "
[1] " en"	[1] "erc"	[1] "qui"	[1] " ir"	[1] "tat"	[1] "t n"
[1] "eni"	[1] "rci"	[1] "uip"	[1] "iru"	[1] "ate"	[1] " nu"
[1] "nim"	[1] "cit"	[1] "ip "	[1] "rur"	[1] "te "	[1] "nul"
[1] "im "	[1] "ita"	[1] "p e"	[1] "ure"	[1] "e v"	[1] "ull"
[1] "m a"	[1] "tat"	[1] " ex"	[1] "re "	[1] " ve"	[1] "lla"

[1]	"la "	[1]	"upi"	[1]	"ui "
[1]	"a p"	[1]	"pid"	[1]	"i o"
[1]	" pa"	[1]	"ida"	[1]	" of"
[1]	"par"	[1]	"dat"	[1]	"off"
[1]	"ari"	[1]	"ata"	[1]	"ffi"
[1]	"ria"	[1]	"tat"	[1]	"fic"
[1]	"iat"	[1]	"at "	[1]	"ici"
[1]	"atu"	[1]	"t n"	[1]	"cia"
[1]	"tur"	[1]	" no"	[1]	"ia "
[1]	"ur."	[1]	"non"	[1]	"a d"
[1]	"r. "	[1]	"on "	[1]	" de"
[1]	". E"	[1]	"n p"	[1]	"des"
[1]	" Ex"	[1]	" pr"	[1]	"ese"
[1]	"Exc"	[1]	"pro"	[1]	"ser"
[1]	"xce"	[1]	"roi"	[1]	"eru"
[1]	"cep"	[1]	"oid"	[1]	"run"
[1]	"ept"	[1]	"ide"	[1]	"unt"
[1]	"pte"	[1]	"den"	[1]	"nt "
[1]	"teu"	[1]	"ent"	[1]	"t m"
[1]	"eur"	[1]	"nt,"	[1]	" mo"
[1]	"ur "	[1]	"t, "	[1]	"mol"
[1]	"r s"	[1]	", s"	[1]	"oll"
[1]	" si"	[1]	" su"	[1]	"lli"
[1]	"sin"	[1]	"sun"	[1]	"lit"
[1]	"int"	[1]	"unt"	[1]	"it "
[1]	"nt "	[1]	"nt "	[1]	"t a"
[1]	"t o"	[1]	"t i"	[1]	" an"
[1]	" oc"	[1]	" in"	[1]	"ani"
[1]	"occ"	[1]	"in "	[1]	"nim"
[1]	"cca"	[1]	"n c"	[1]	"im "
[1]	"cae"	[1]	" cu"	[1]	"m i"
[1]	"aec"	[1]	"cul"	[1]	" id"
[1]	"eca"	[1]	"ulp"	[1]	"id "
[1]	"cat"	[1]	"lpa"	[1]	"d e"
[1]	"at "	[1]	"pa "	[1]	" es"
[1]	"t c"	[1]	"a q"	[1]	"est"
[1]	" cu"	[1]	" qu"	[1]	"st "
[1]	"cup"	[1]	"qui"	[1]	"t l"
[1]	" la"				
[1]	"lab"				
[1]	"abo"				
[1]	"bor"				
[1]	"oru"				
[1]	"rum"				
[1]	"um."				

Warning message:

In readLines("BIBD1.txt") : incomplete final line found on 'BIBD1.txt'

> |

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")
```

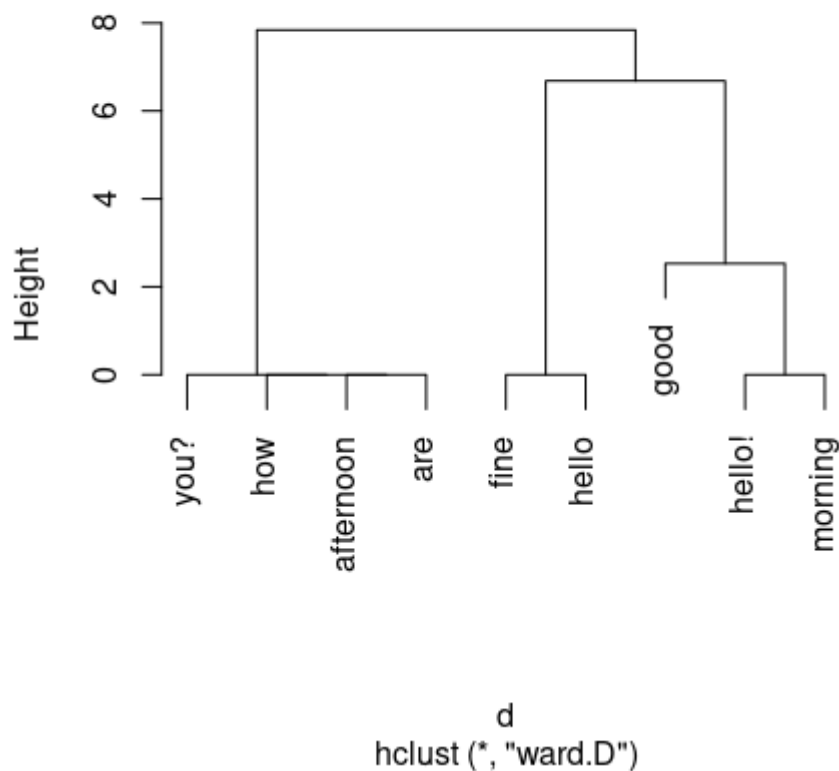
Source Code a:-

```
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=weightTfIdf,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)
```

Output:

```
<<TermDocumentMatrix (terms: 9, documents: 3)>>
Non-/sparse entries: 10/17
Sparsity           : 63%
Maximal term length: 9
Weighting          : term frequency (tf)
Sample            :
      Docs
Terms  file1.txt file2.txt file3.txt
afternoon      0         0         1
are            0         0         1
fine           0         1         0
good          1         0         1
hello         0         1         0
hello!        1         0         0
how           0         0         1
morning       1         0         0
you?          0         0         1
> my.df.scale<-scale(my.df)
> d<-dist(my.df.scale,method = "euclidean")
> fit<-hclust(d,method = "ward.D")
> plot(fit)
>
```

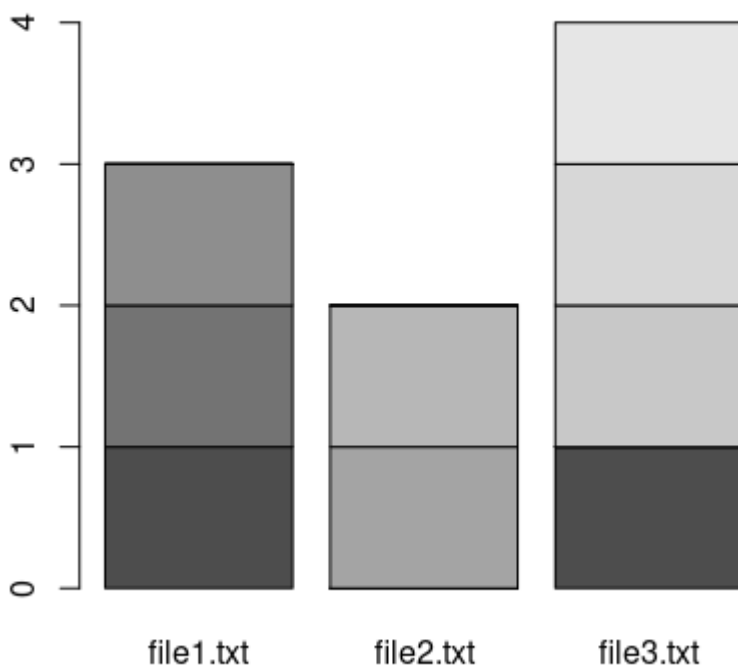
Cluster Dendrogram

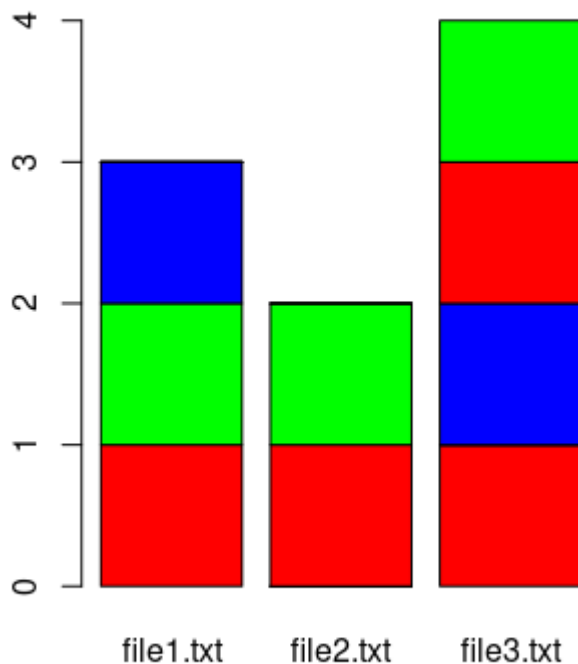


Source code b (using bar plot with and without color):-

```
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

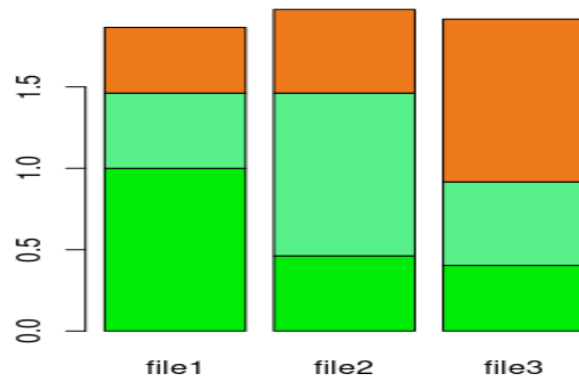
```
minhash <- minhash_generator(200, seed = 235)
ats <- TextReuseCorpus(dir = "files", tokenizer = tokenize_ngrams, n = 5,
minhash_func = minhash)
```

```
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:

```
# A tibble: 3 x 3
  a      b    score
<chr> <chr> <dbl>
1 file1 file2 0.462
2 file1 file3 0.403
3 file2 file3 0.513
```



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

```

        for(int i : arrayList)
        {
            firstMoment += i;
            secondMoment += i * i;
        }
        System.out.println("First Moment:\t\t\t" + firstMoment);
        System.out.println("Second Moment:\t\t\t" +
secondMoment);

    }
}

```

Output:

```

[Running] cd "c:\Users\USER\Downloads\" && javac KMoments.java && java KMoments
Note: KMoments.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
a      b      c      b      d      a      c      d      a      b      d      c      a      a      b
Zeroth moment:      0
First Moment:      15
Second Moment:      59

[Done] exited with code=0 in 13.622 seconds

```