THE ANALYTICS DOZEN

Tutorial (Orange Version)

Main Dataset used: HDB Housing Dataset

Contents

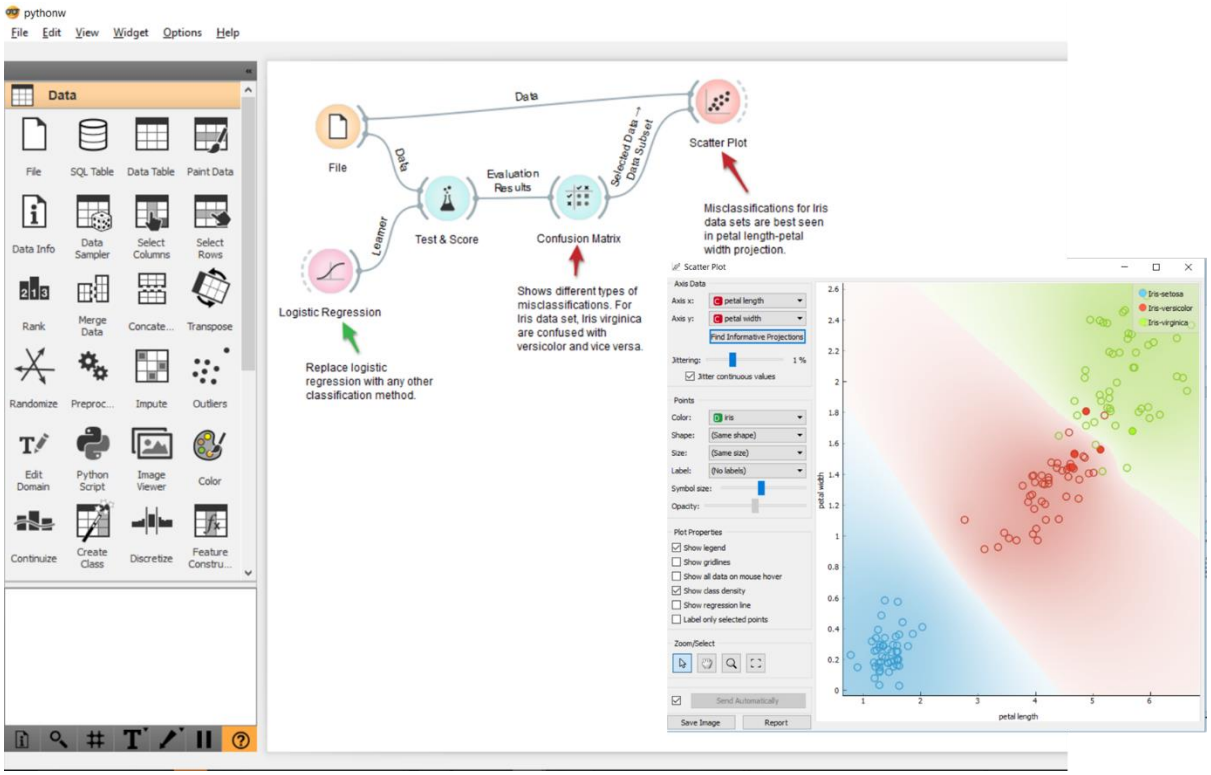
INTRODUCTION TO ORANGE DATA ANALYTICS TOOLKIT

Summary

This lab is to familiarize the student with the Orange Data Analytics Toolkit. The student will learn the various features of Orange and use it effectively.

Business Case

Learn how to use Orange – a free and open source data analytics toolkit.



Learning Objectives

Upon completing this lab, the student will be able to

1. Download and install Orange
2. Understand what is a workflow and its components
3. Identify the major features of Orange
4. Investigate an existing model
    a. Data Tables
    b. Scatterplot
    c. Confusion Matrix

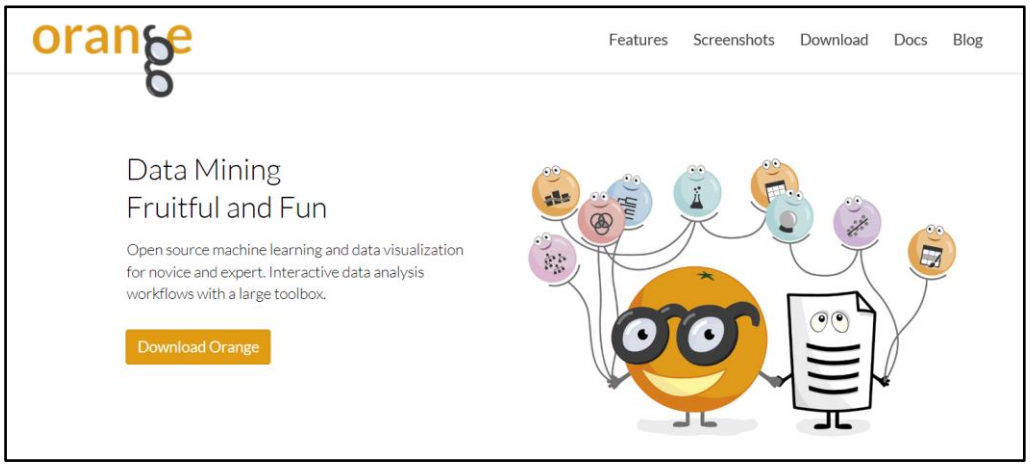
Lab Requirements

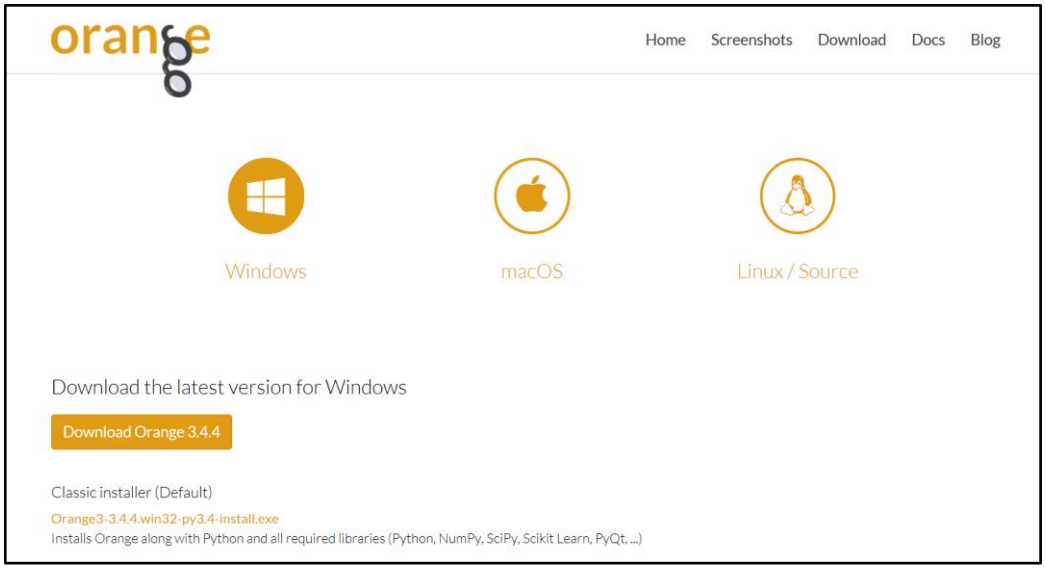
A Windows, Mac or Linux Laptop can be used for all the labs with Orange.

INSTALLING ORANGE

Downloading Orange

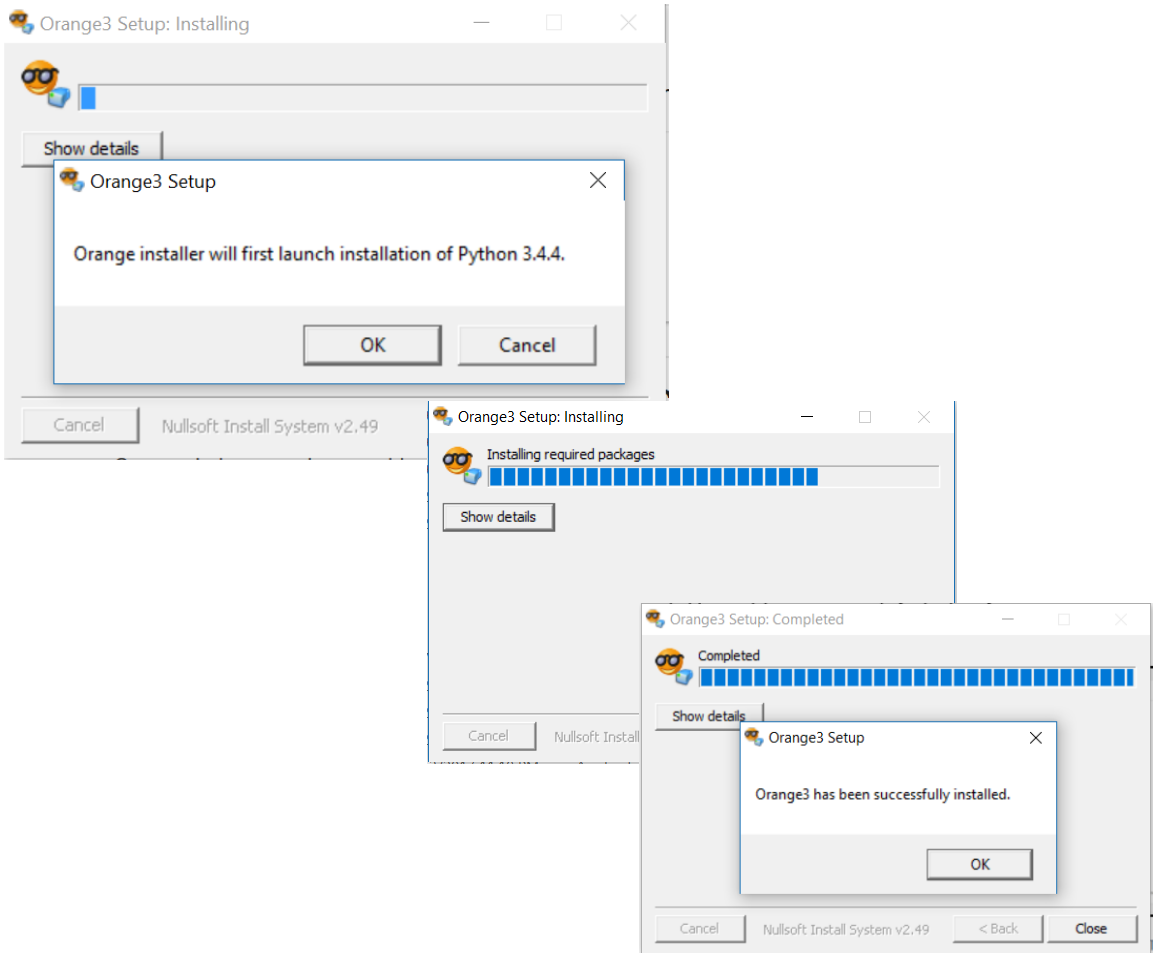
1. Orange can be downloaded from the following site: https://orange.biolab.si
2. If you are using your PC, please go ahead to visit the above URL and download the version of Orange for your operating system.
3. If you are using our lab PC/Laptops, Orange is already installed for you.





Installing Orange

1. Orange is simple. Just click through the installation walk-through and accept the defaults.

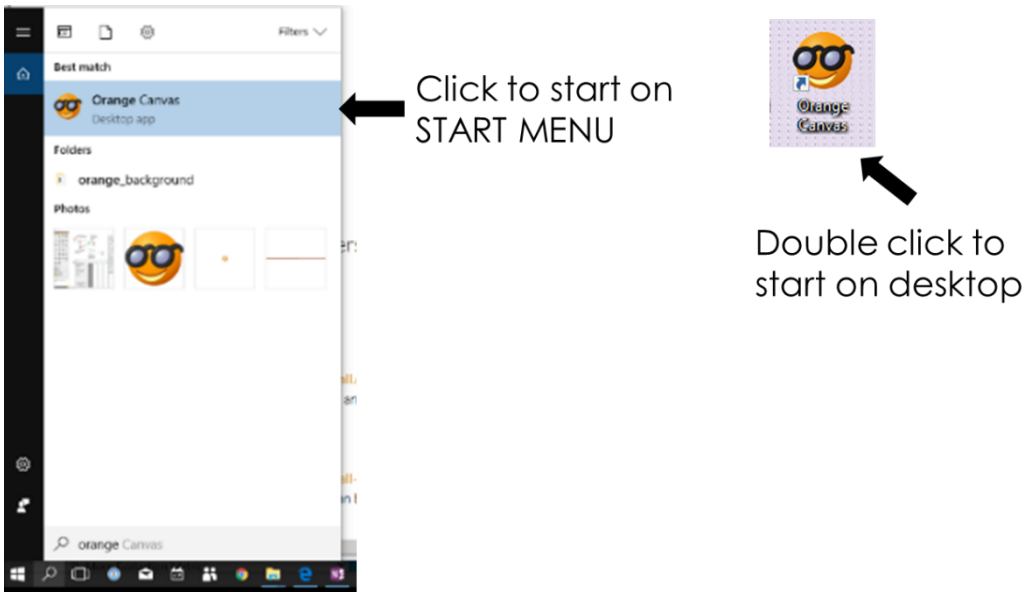


Note that this tutorial will be based on Microsoft Windows version of Orange. The features of Orange across all supported platforms (Windows, MacOS, Linux) is the same.

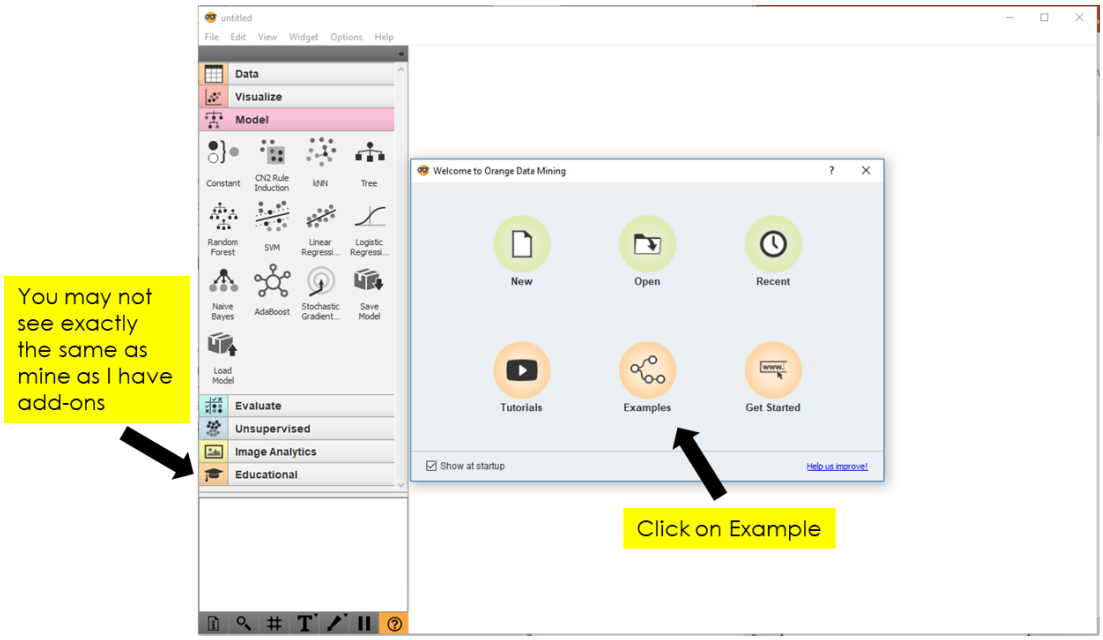
TOUR OF ORANGE

Starting Orange

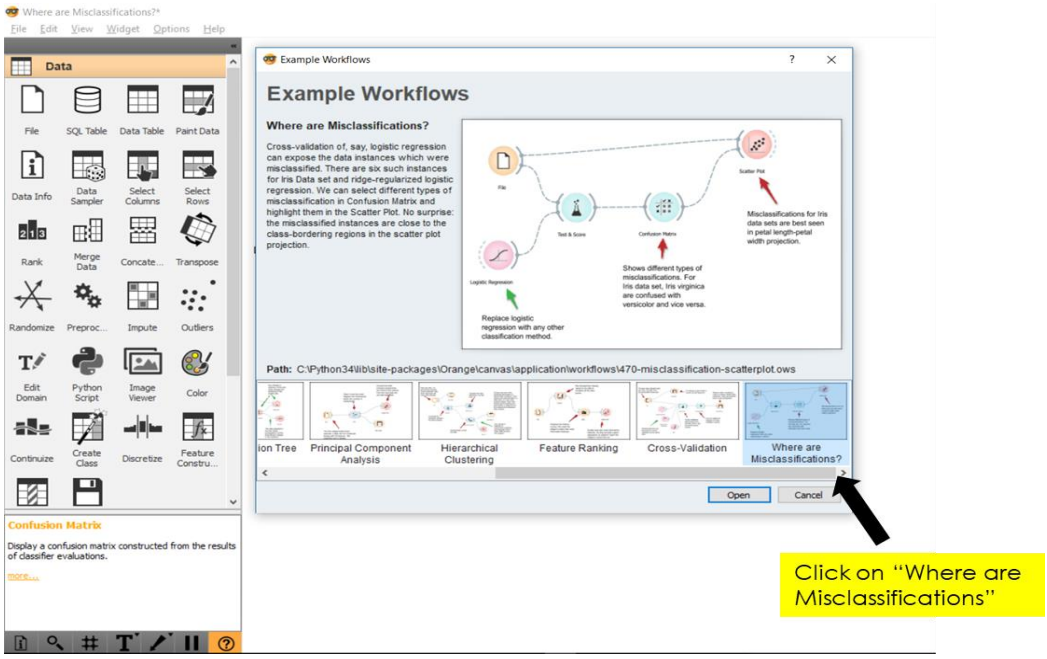
1. Look for Orange Icon on your Start Menu or Desktop. Click to start Orange.



2. Orange typically starts with a Welcome Screen. You can turn this off if you wish. Click on Examples to open an existing workflow.



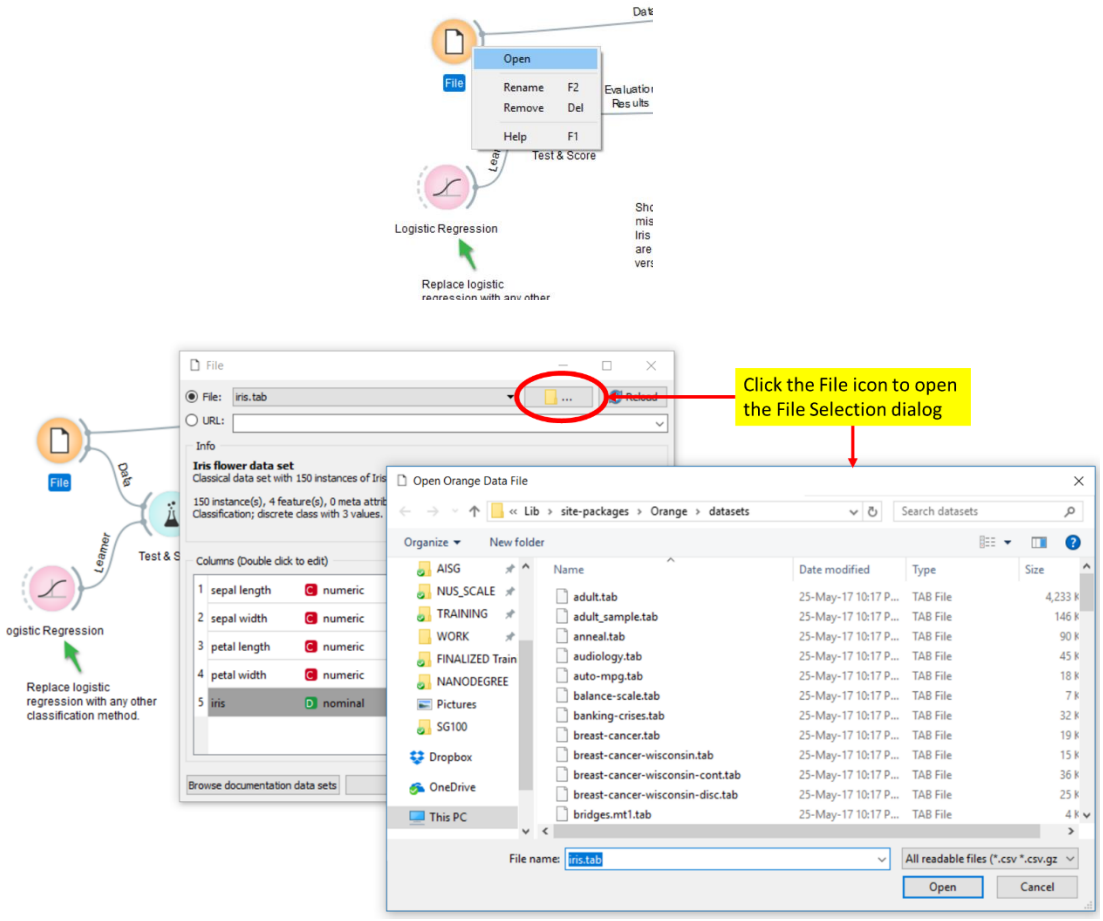
3. Select the Misclassifications workflow.



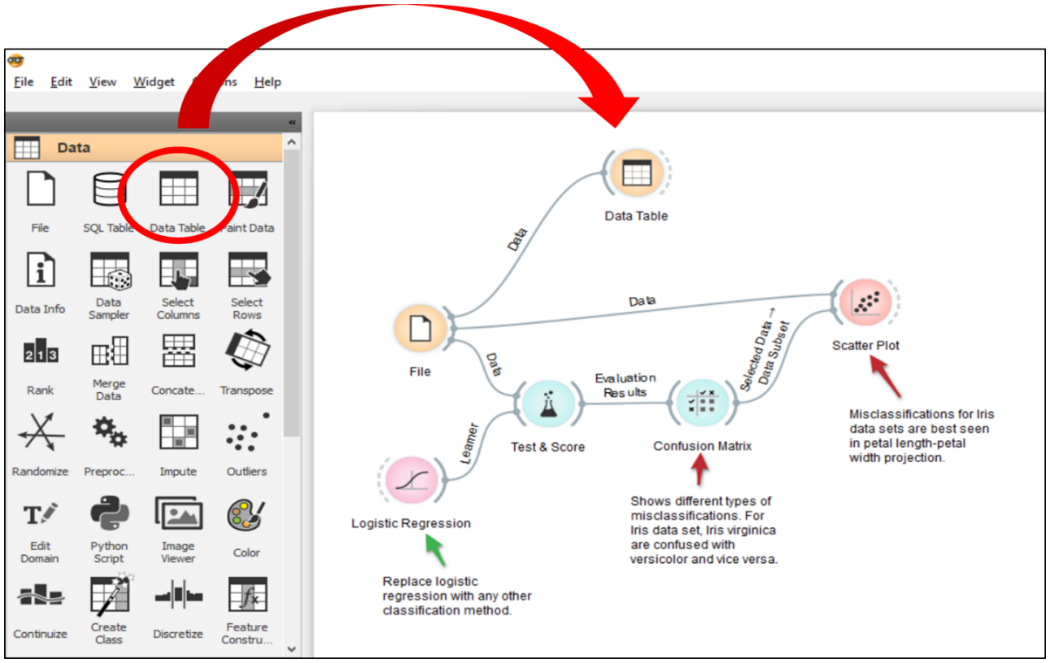
4. Explore the workflow. Orange calls the little bubbles as Widgets. You can connect widgets together to create workflows.

5. You can add text and arrow to annotate the workflows. This is especially useful if you plan to share your workflows with your colleagues.

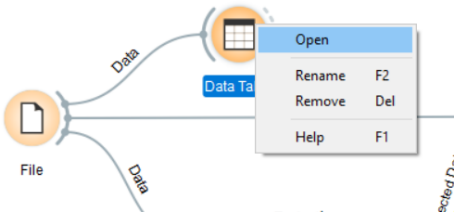
6. Select the IRIS dataset to explore!
    a. Right-click on the **File** widget and select **Open**.
    b. The **File** dialog will open. Select the **File** icon to open the **File Selection** dialog. Select *iris.tab* dataset.



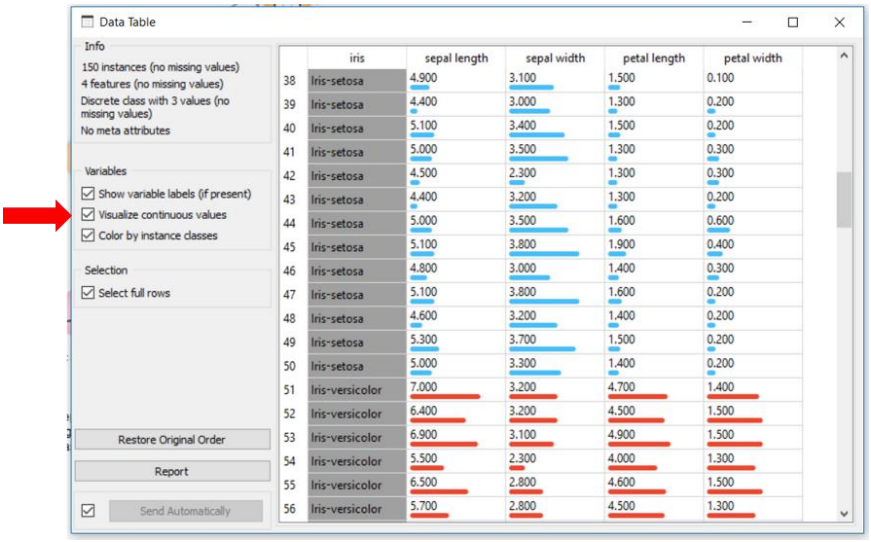
7. Search for the **Data Table** widget in the **Data** panel. Click and drag the **Data Table** widget to the Orange canvas.



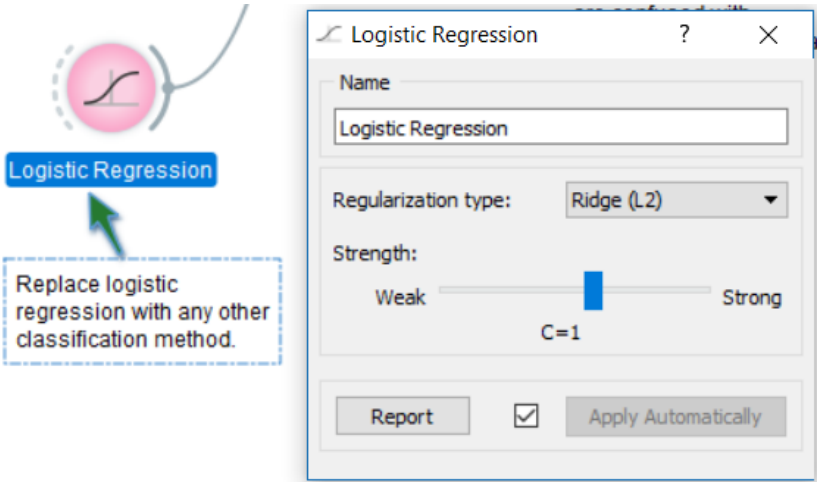
8. Right-click the Data Table widget and explore the data.



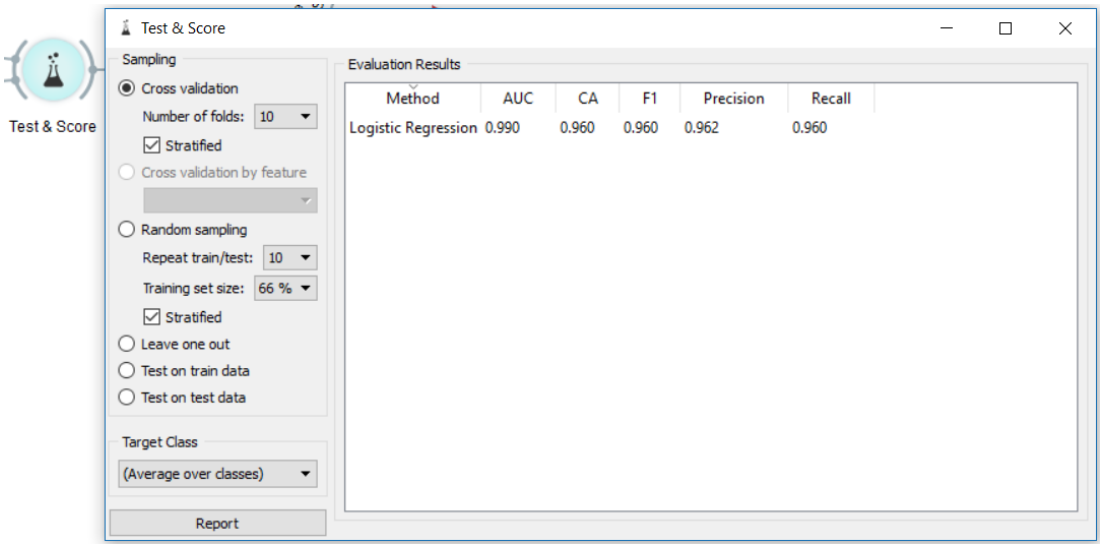
9. You may want to turn on Visualize continuous values.



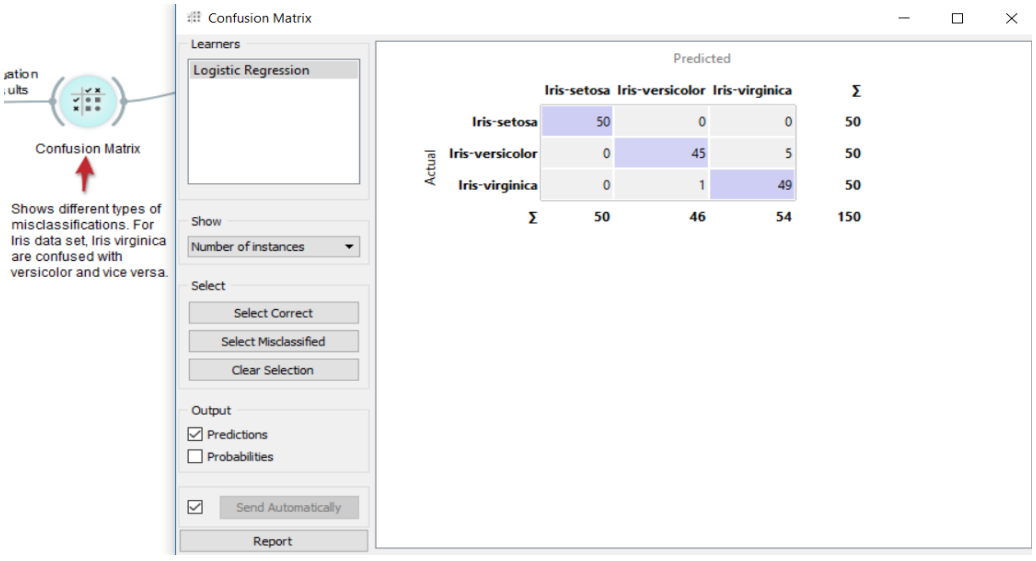
10. The model used in this example is the Logistic Regression model. Right-click on the **Logistic Regression widget** to see the options available. You can typically keep the default options as Orange selects sane defaults.



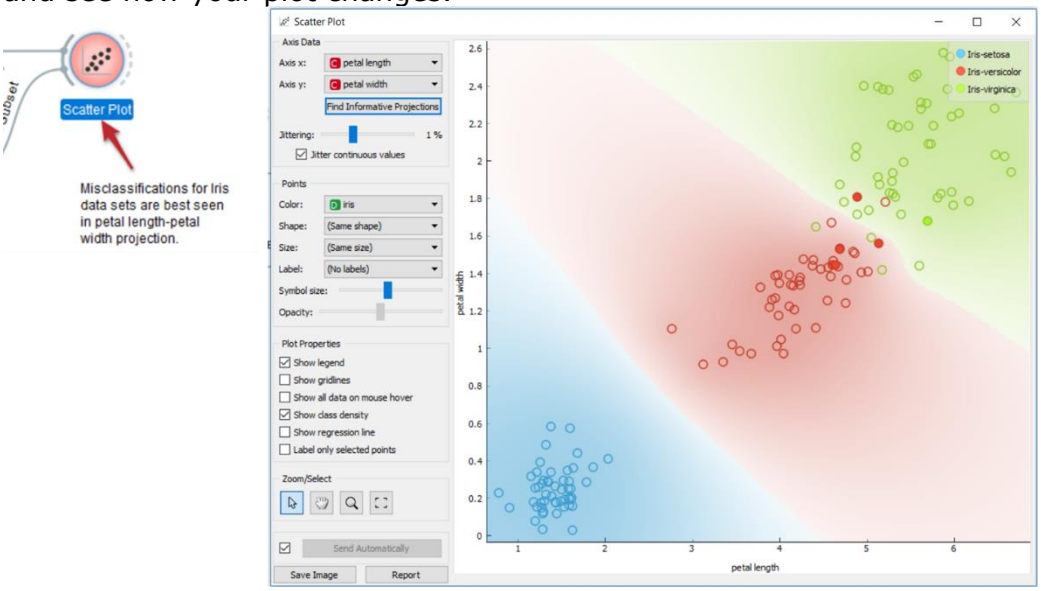
11. Right-click on the **Test & Score widget** to see the results of the model. Do not worry too much about AUC, CA, F1 etc. We will discuss these later. Just know that the scores look pretty good (near 1.0) which means the Logistic Regression algorithm worked well on this particular dataset and was able to classify most of the iris flowers correctly!



12. Right-click on the **Confusion Matrix widget** to see the results of the model. You will see that the model predicted all iris-setosa correctly, 45 of the iris-versicolor correctly (and misclassified 5 as iris-virginica), and 49 iris-virginica correctly (and misclassified 1 as iris-versicolor). Again do not worry, we will discuss the Confusion Matrix in more detail later.

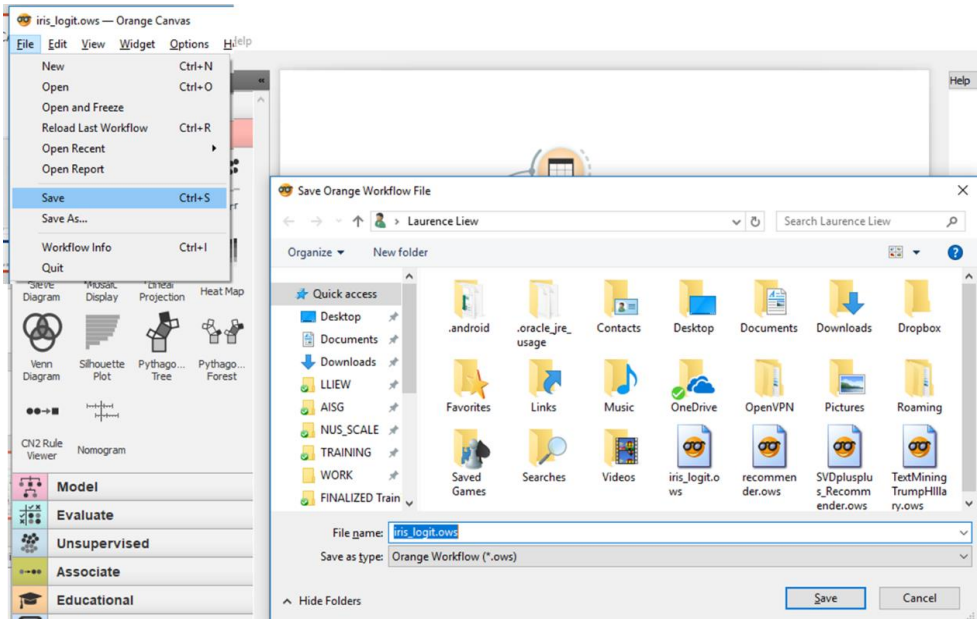


13. Go ahead to explore the other widgets. Check out the **Scatter Plot widget** and the nice visualization it provides. Explore the various options it provides and see how your plot changes.

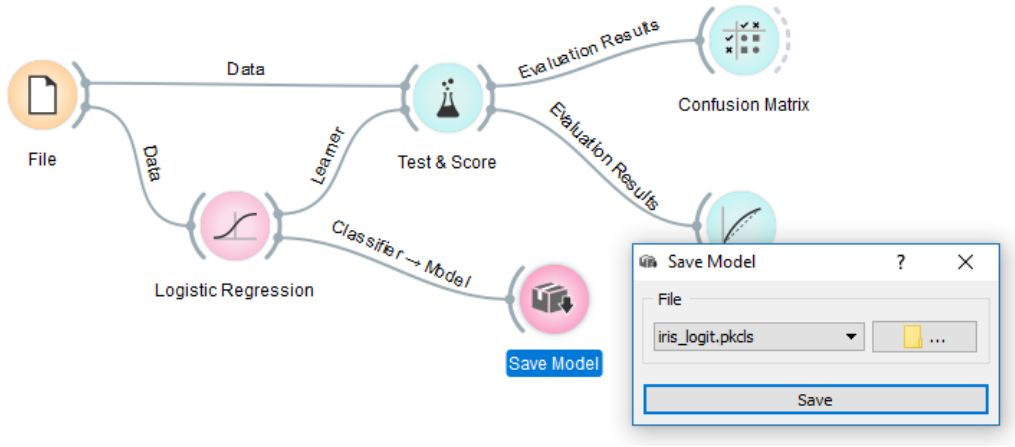


Save your best model

1. Okay, so you are done building your model. You can save the workflow from the file menu and come back and improve on the model later. You can even send the whole workflow to your colleagues. He/she only needs your workflow file and the dataset.

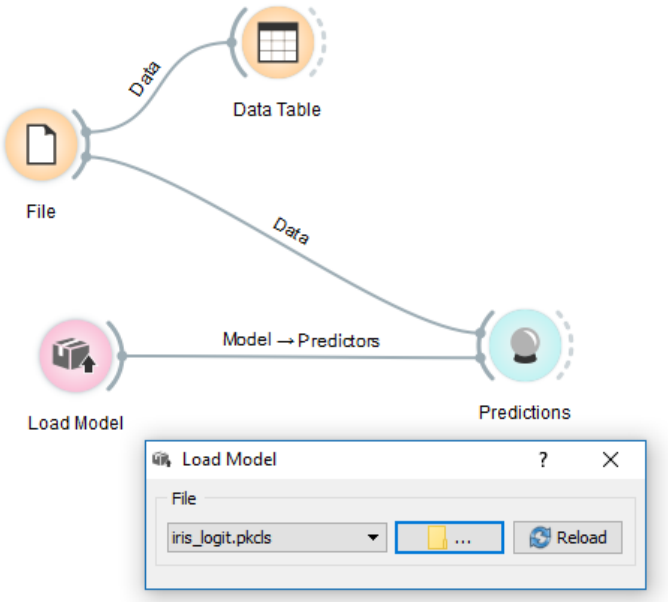


2. If your model is sensitive and you rather not share your whole workflow, Orange allows you to just save the model itself and share it with others. Add a **Save Model** widget as shown to your best model (Logistic Regression). The model will be saved as a Python pickled file. Note that you need to link the **File** widget to the Logistic Regression model so that the **Save Model** widget will know the schema of the dataset.



Load your model

1. You or your colleague can use the **Load Model** widget to load the model and run the model on a similar set of iris dataset and prediction the categories of the iris flowers.
2. Build the workflow as shown below.



Conclusion

This concludes the Introduction to Orange Data Analytics toolkit. What you have learnt:

1. Download and install Orange
2. Use Orange to load and explore a dataset
3. Explored and ran a simple supervised learning model (Logistic Regression) on the iris.tab dataset, and saw how it performed with the Test & Score tables and the Confusion Matrix.
4. You reviewed the data in both table format and in a scatter plot.
5. You learnt how to save your model so you can improve on it later and share it with colleagues.

You have worked on your FIRST supervised learning model!

DATA PREPARATION, VISUALIZATION AND FEATURE ENGINEERING

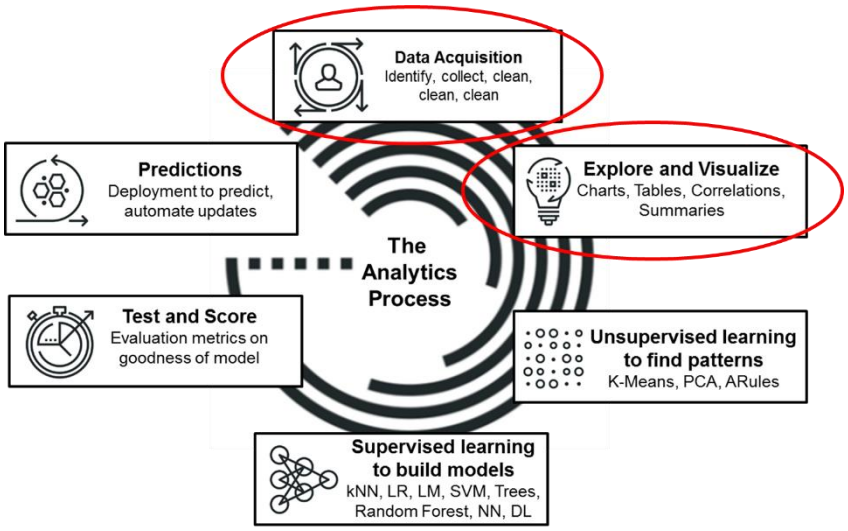
Summary

This lab is to familiarize the student with the data preparation, visualization and feature engineering. Feature engineering here refers to how you can combine, reduce or add new variables into your dataset to make your dataset better for analysis.

Business Case

This part is often the most important process and most "art" of the data science practice. Good data generates accurate models, while bad data – no matter how good the algorithm, will generate poor models.

Often you also need to be proficient in SQL, ie, your ***SQL Kung-fu*** needs to be good.



Learning Objectives

Upon completing this lab, the student will be able to

a. Build a workflow to load data
b. Prep-process or clean the data
c. Perform feature engineering
d. Visualize and explore the datasets

Lab Requirements

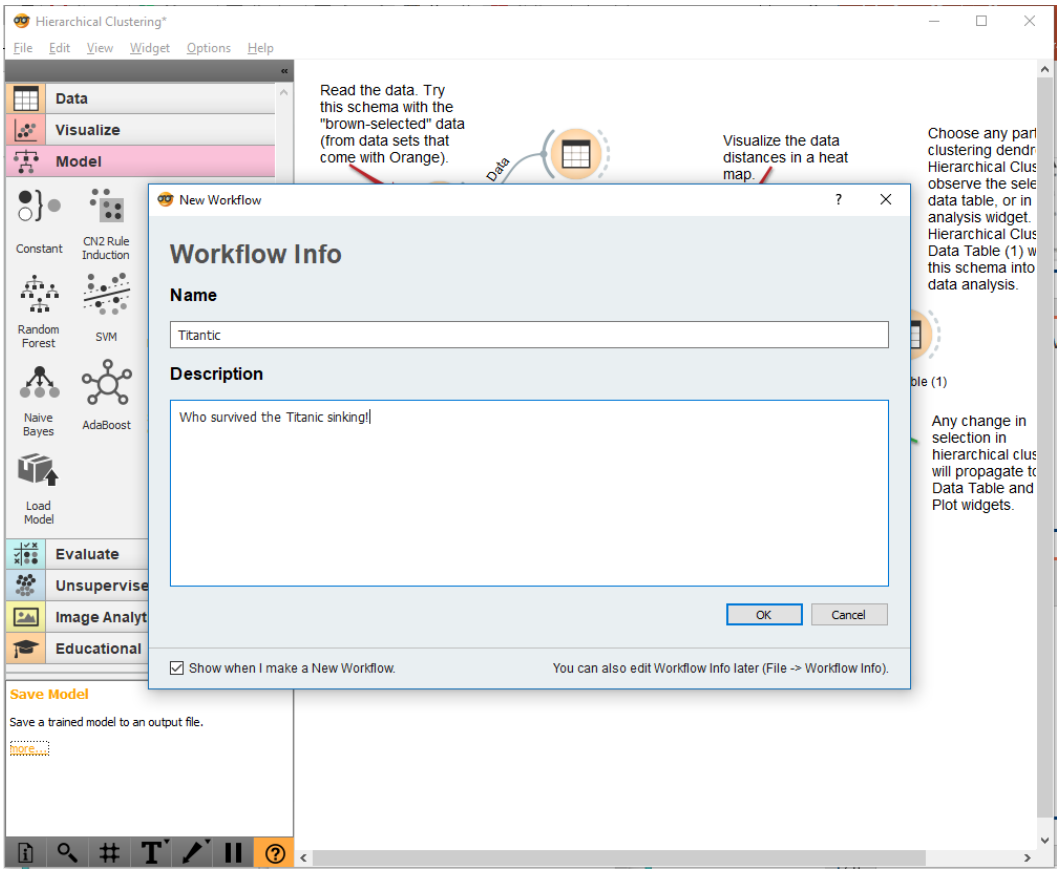
A Windows, Mac or Linux Laptop can be used for all the labs with Orange.

Build your first data exploration workflow

Create a new workflow to analyze your data

1. Start Orange and create a new workflow. You may need to close any current open workflow.

    a. File -> New
    b. Give your workflow a Name and maybe some description
    c. Click OK

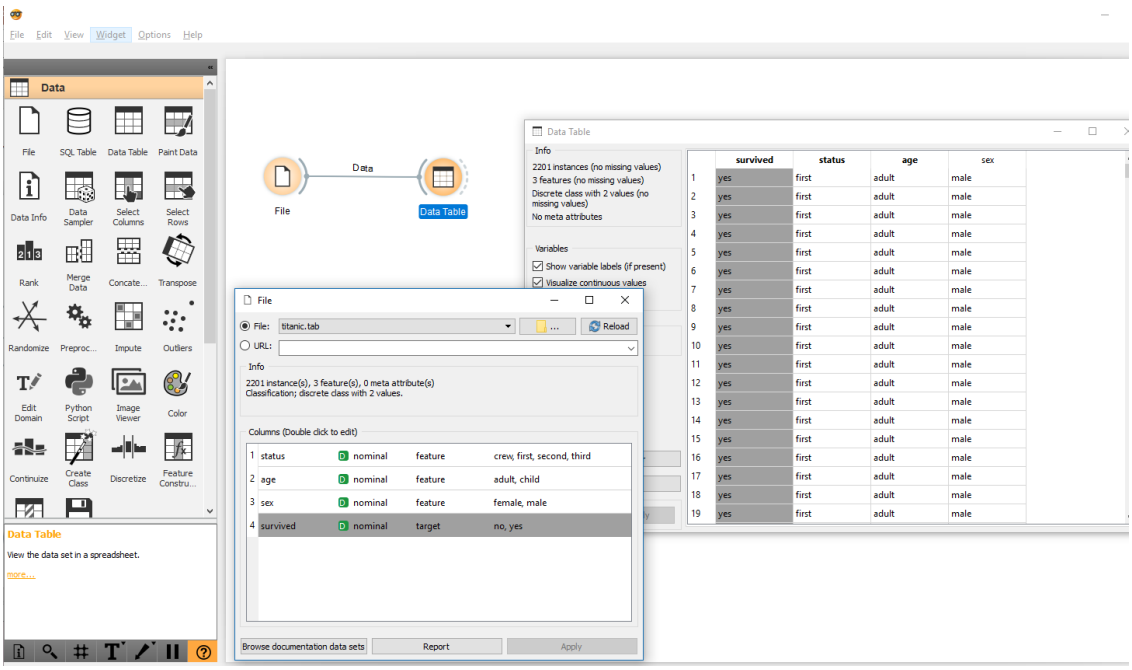


2. Place a **File** widget and **Data Table** widget on the canvas. Click and drop the File Widget onto the canvas. There are 3 ways to put Widget onto canvas:

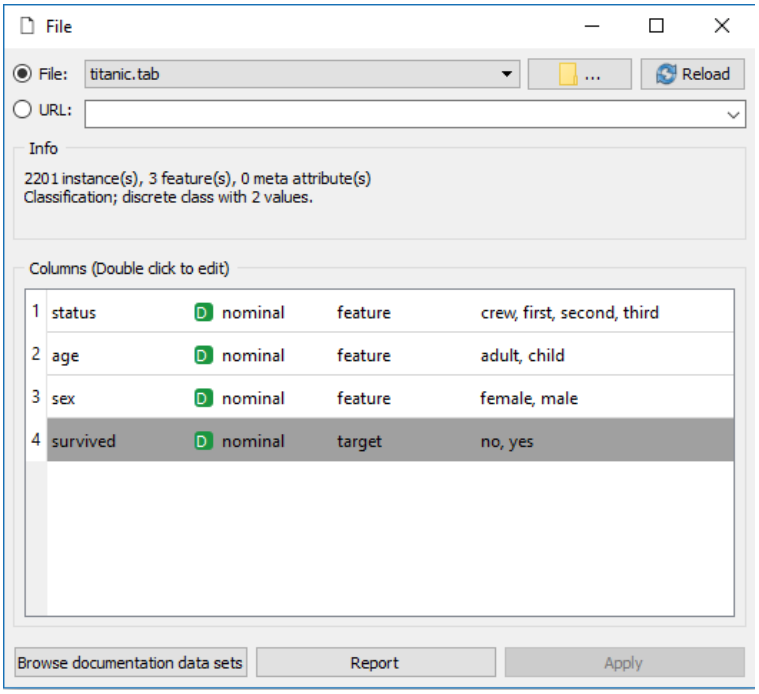
    a. Select Widget and drag to canvas
    b. Right click on canvas, and select or search
    c. Click on existing Widget's Connection interface, drag and release to place and select COMPATIBLE Widget

3. Link the **File** widget to the **Data Table** widget.

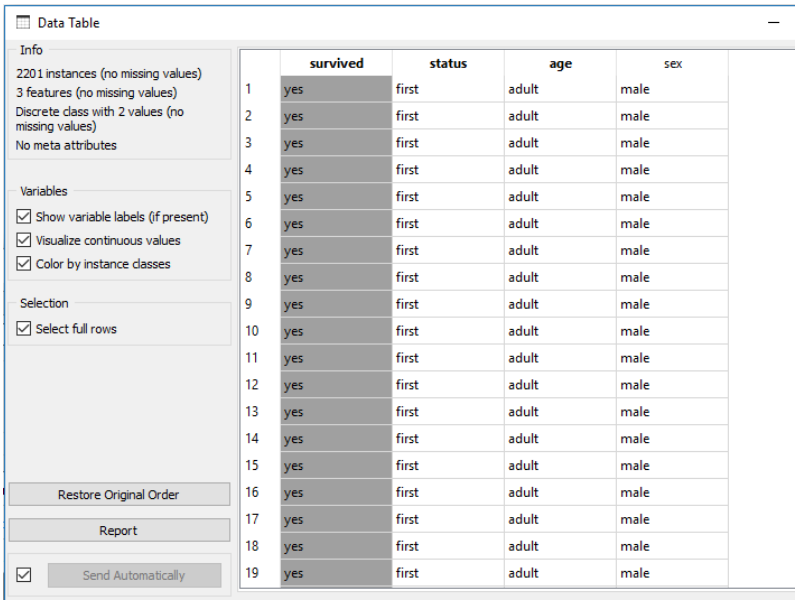
4. Loading the *titanic.tab* dataset data into the **File** widget.



5. Review the **File** dialog. See that this dataset has 2201 instances or rows of data. There are 4 columns in the dataset. 3 features and 1 target variable.



6. Review the Data Table dialog. It displays the data in a familiar Excel row-column format.



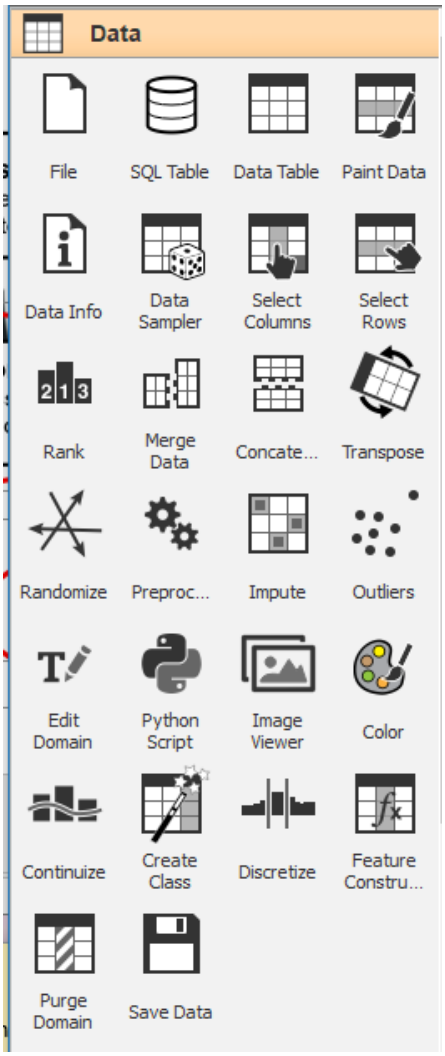
7. The **File** Widget reads the data from a file on your computer. The data can also be from a URL from say a Google doc file on the internet. When you connect a Data Table Widget to it, you can view the data.

   Orange stores the examples dataset in the following directory on my laptop. You may wish to note where your datasets are as it may differ from my installation, it case you need to look for it later.

   C:\Python34\Lib\site-packages\Orange\datasets

Data Processing in Orange

1. Orange has an extensive set of widgets for data process.



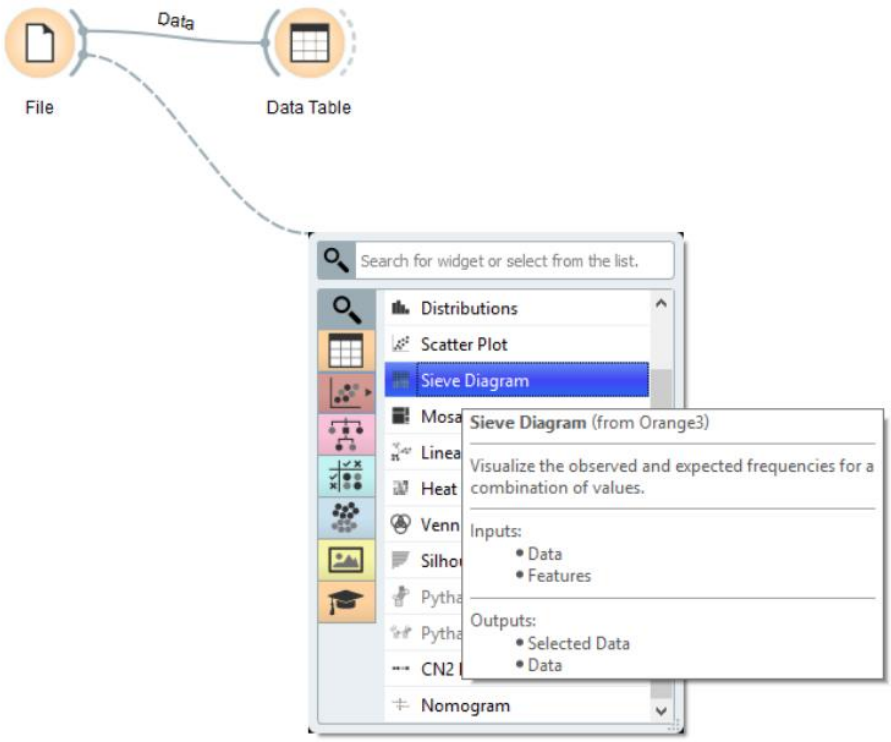
2. Some commonly used widgets in this class include:

Widget	Function
File	Read data from an input file or network and send the data table to the output
Data Table	View the dataset in a spreadsheet like format
Data Sampler	Draw a random subset of data points from the input dataset
Select Columns	Select columns from the data table and assign them to data features or classes
Preprocess	Construct a data pre-processing pipeline

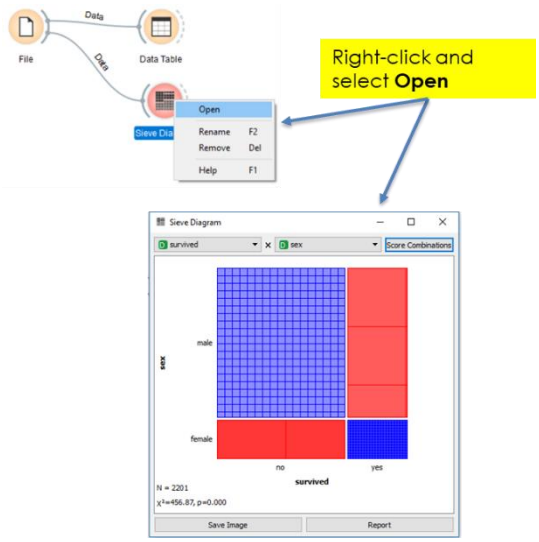
Visualize your data to understand it!

1. Put a **Sieve Diagram** widget on the canvas and connect to the **File** widget.

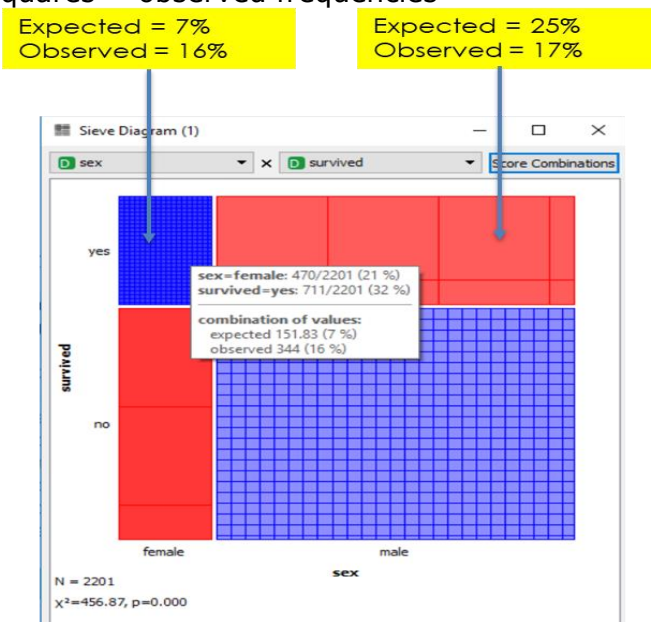
   Here we click on the **File** widget and then let Orange tell us what the compatible widget it can connect to. Select the **Sieve Diagram** widget.



2. Right click and open the Sieve Diagram widget.

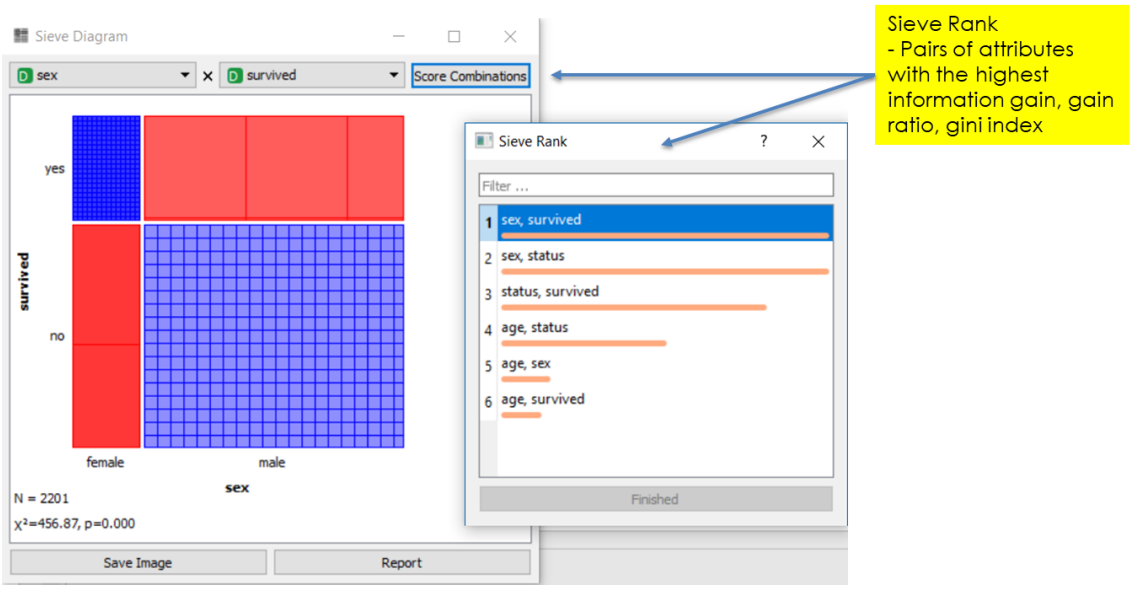


3. The Sieve Diagram is a 2D plot for visualizing frequencies (of an event/occurrence) and comparing them to **expected frequencies** under assumption of independence
    a. Colour
        i. Blue: positive deviation from independence
        ii. Red: negative deviation
    b. # of squares = observed frequencies

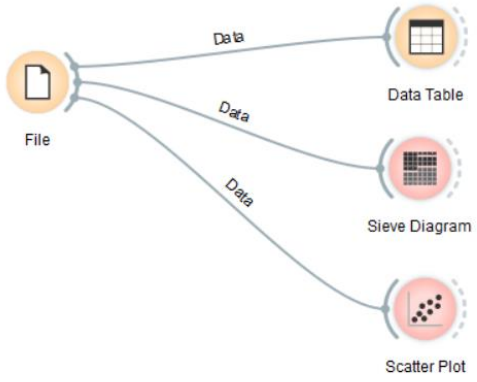


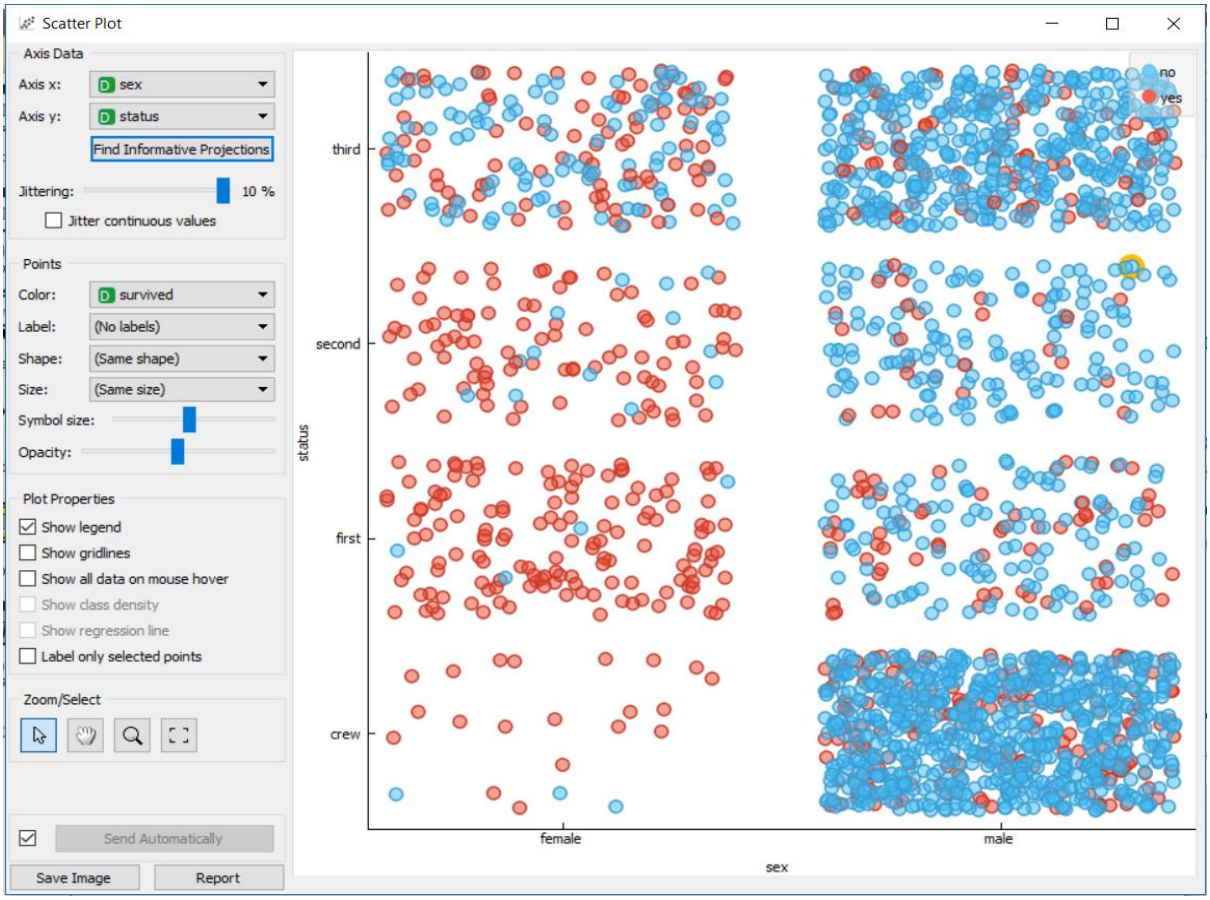
4. Observations:
    a. Sex-female = 470/2201 (21%)
    b. Survived-yes = 711/2201 (32%)
    c. Therefore, EXPECTED sex-female-survived = 32% x 470 / 2201 = 7% (150)
    d. But actual observed = 344 -> 16%

5. But wait! I am not a data scientist. How would I know what variables to compare? Orange have you covered here. Click on **Score Combinations**. The Sieve Rank shows the most interesting and relevant combinations to study based on internal statistical calculations Orange have done (Information theory).



6. Connect the **Scatter Plot** widget and explore.

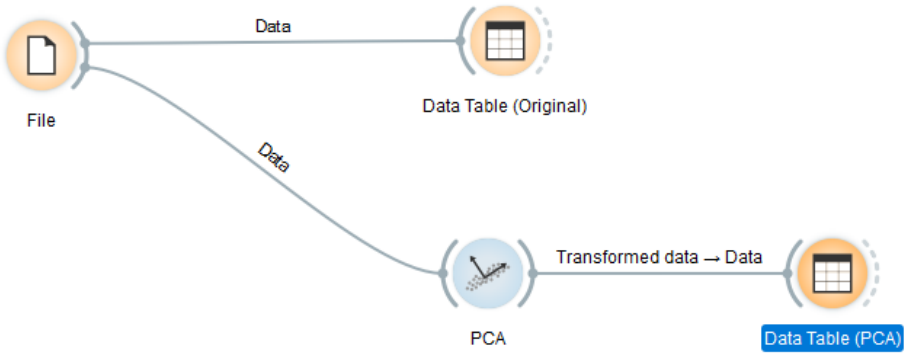




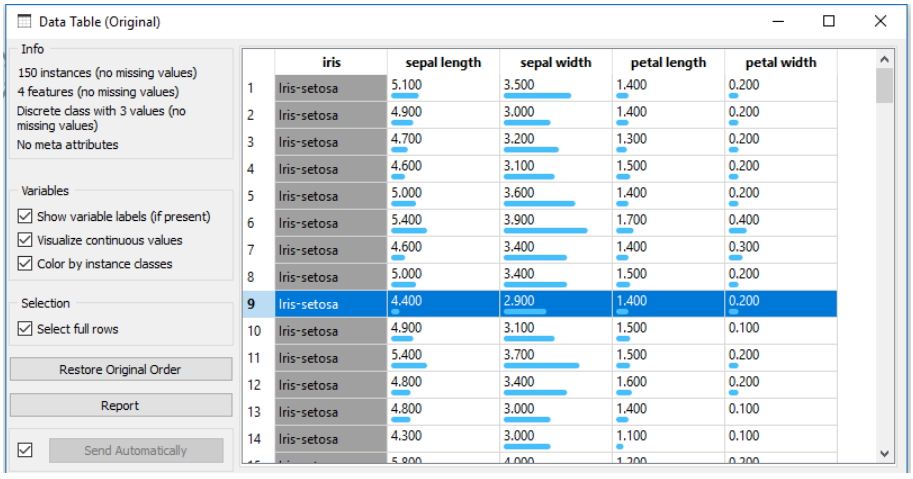
Mouse over to see additional details
Check out "Find Informative Projections"

Feature Engineering with PCA

1. Create a new workflow as shown. Load up the famous *iris.tab* dataset.

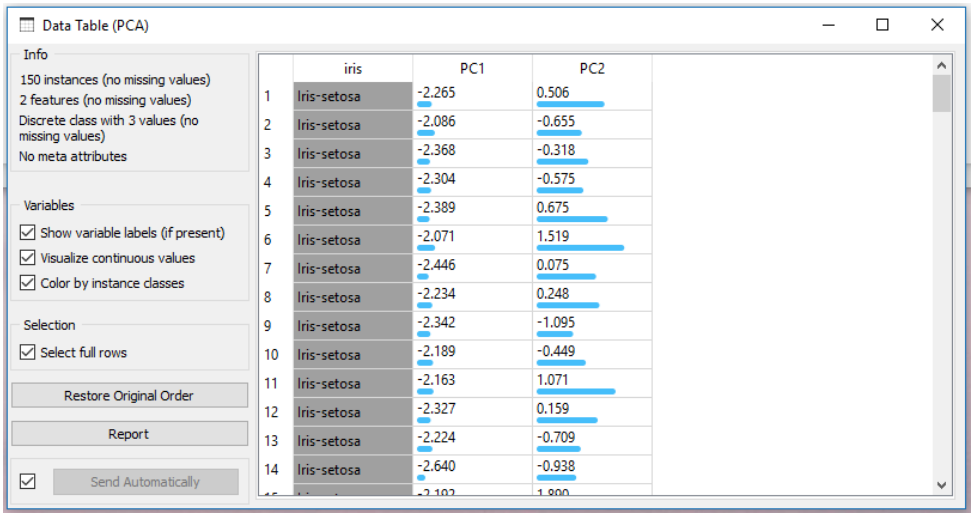


2. The **Data Table (Original)** dataset is shown below. It has 4 features (or variables) and 1 target column. Question: Do we need all 4 features to classify the iris flowers? Imagine storing and only using 2 columns of data compared to 4 columns (half the storage requirements – think big data).

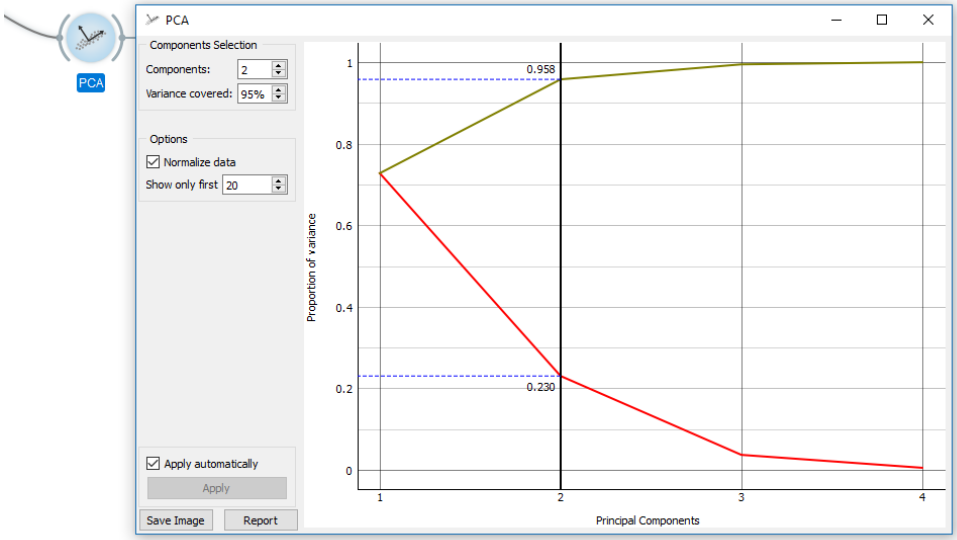


	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100

3. Here is the **Data Table (PCA)** dataset. Here the **Principal Component Analysis (PCA)** widget computes the PCA linear transformation of the input data. It outputs either a transformed data set with weights of individual instances or weights of principal components.



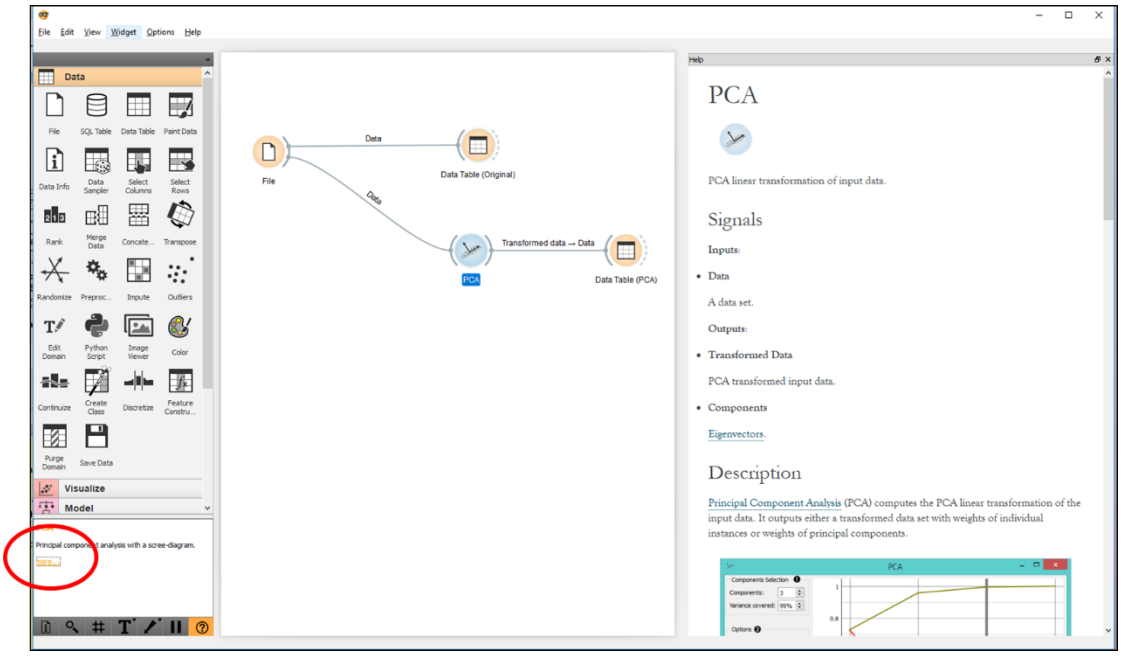
4. But how good is two components? Open the **PCA** widget. See that with 2 components, 95% of the variance can be explained. Using all 4 features only gain us an additional 5%. So, is this good enough?



Conclusion

You have learnt how to work with Orange to build a data analysis pipeline and prepare the data for analysis. You have learnt how to visualize the data.

Orange includes a wide range of visualization widgets. You are encouraged to explore them. Orange help system is decent. Select the widget of interest on your canvas and a short description appears on the lower left-corner. Click on more to bring up the detail help panel.



THE ANALYTICS DOZEN

Summary

This lab is to familiarize the student with TWELVE common algorithms – The Analytics Dozen - used in most data science projects.

Note that this tutorial is to EXPOSE you to these common algorithms, so that you can have a working level conversation with your data science teams and vendors. It is UNLIKELY you will use Orange to deploy a production system yourself.

Business Case

The Analytics Dozen is a selection of most commonly used analytics algorithms in most data science projects.

While there are many state of the art algorithms used by large internet companies like Netflix and Amazon, often the TWELVE algorithms (what I call The Analytics Dozen) presented here suffice for most projects (at least in most initial projects) and will help you achieve significant accuracy with good data sets and feature engineering.

We will cover the use of the algorithms with some basic theory (little or no math) of the algorithm, so that as technical managers managing data science projects, you will have a working level knowledge of these algorithms. This will allow you to discuss in the same lingo with your data science and external vendors.

We will cover both unsupervised and supervised learning algorithms show below.

Unsupervised Learning

1. K-Means
2. Principal Component Analysis
3. Association Rules

Supervised Learning

4. Naïve Bayes
5. K Nearest Neighbors
6. Linear Regression
7. Logistic Regression
8. Tree, Random Forest
9. Support Vector Machine
10. CN2 Rule Induction
11. Recommendation
12. Text Analytics

Learning Objectives

Upon completing this lab, the student will be able to

1. Build a basic classification and regression models
2. Determine the goodness of the models build
3. Understand TWELVE most commonly used algorithms in the industry
4. Have a common lingo to enable him/her to engage with the data science team and/or data scientist.

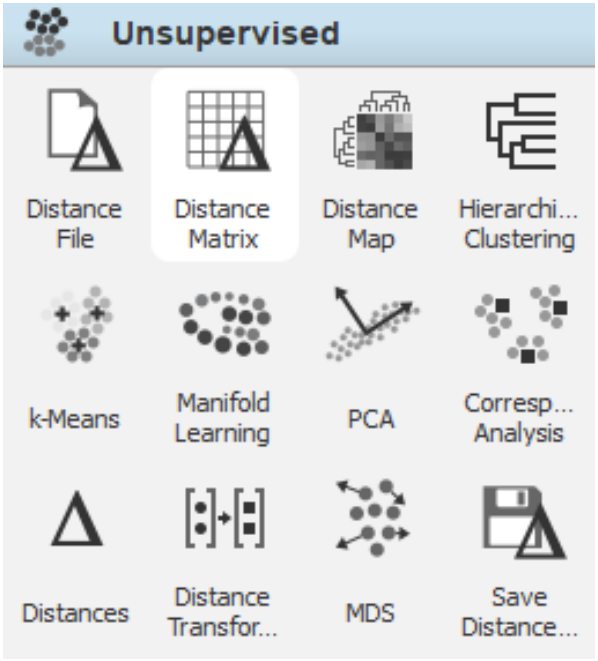
Lab Requirements

A Windows, Mac or Linux Laptop can be used for all the labs with Orange.

UNSUPERVISED LEARNING

Unsupervised Learning algorithms in Orange

1. Orange has a range of unsupervised learning algorithms. This is shown below. We will however focus on the following three:
    a. K-means clustering
    b. Principal Component Analysis (PCA)
    c. Association Rules



2. Often these unsupervised algorithms are used at the beginning of a project to
    a. help understand the characteristics of the data
    b. to understand the domain by allowing the data scientist to ask questions about the data to the domain experts
    c. feature engineering

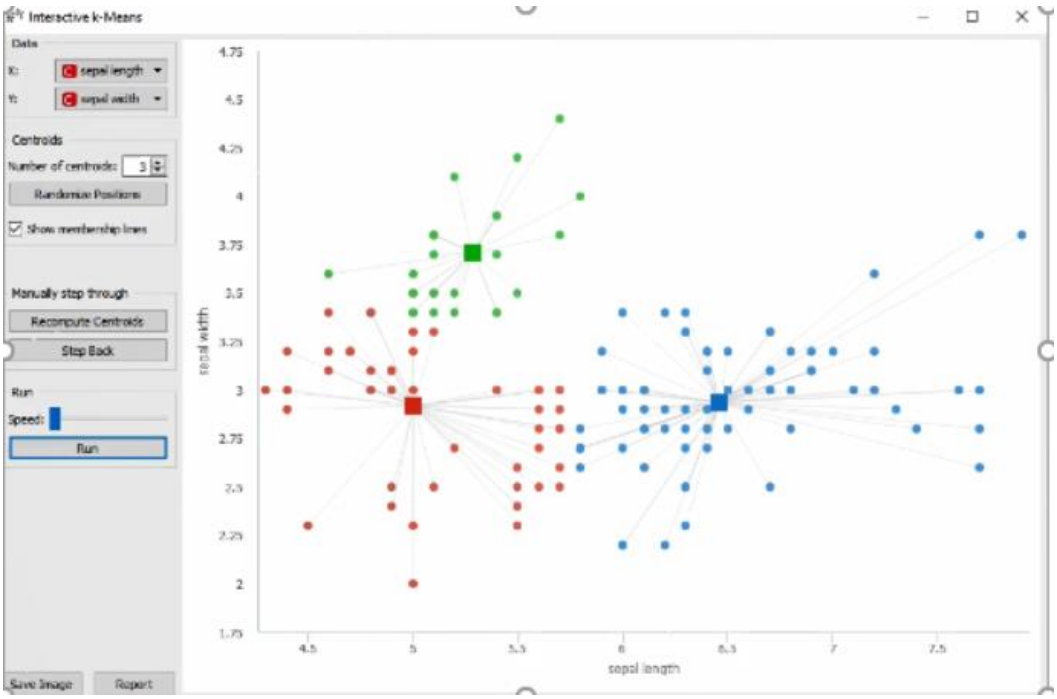
K-MEANS CLUSTERING

Motivation

Often the user is presented with data where there is no known label (or target). That is, we do not know how many categories there are in the data, nor what the data looks like. In such scenario, the data scientist will often apply techniques such as k-means clustering to determine and find patterns in the data. This will help him/her in further understanding the dataset and then ask the right questions (often to get better data).

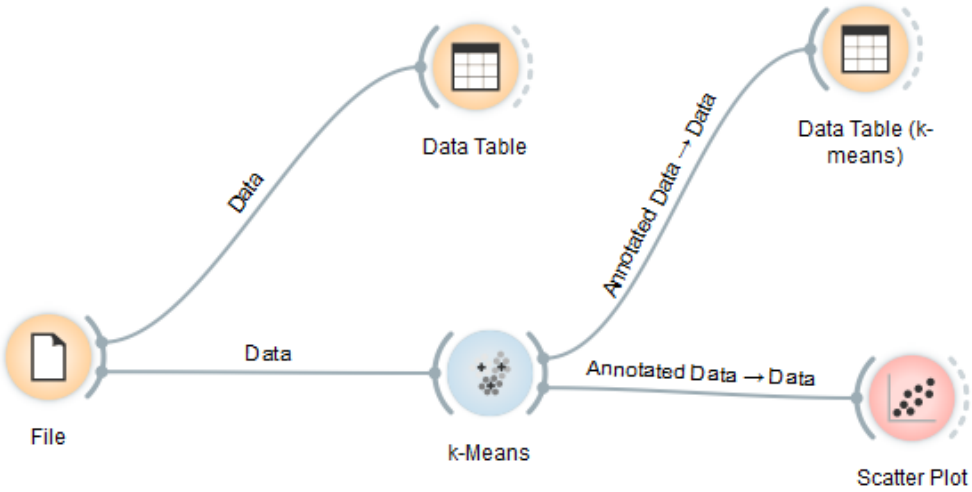
Theory

k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

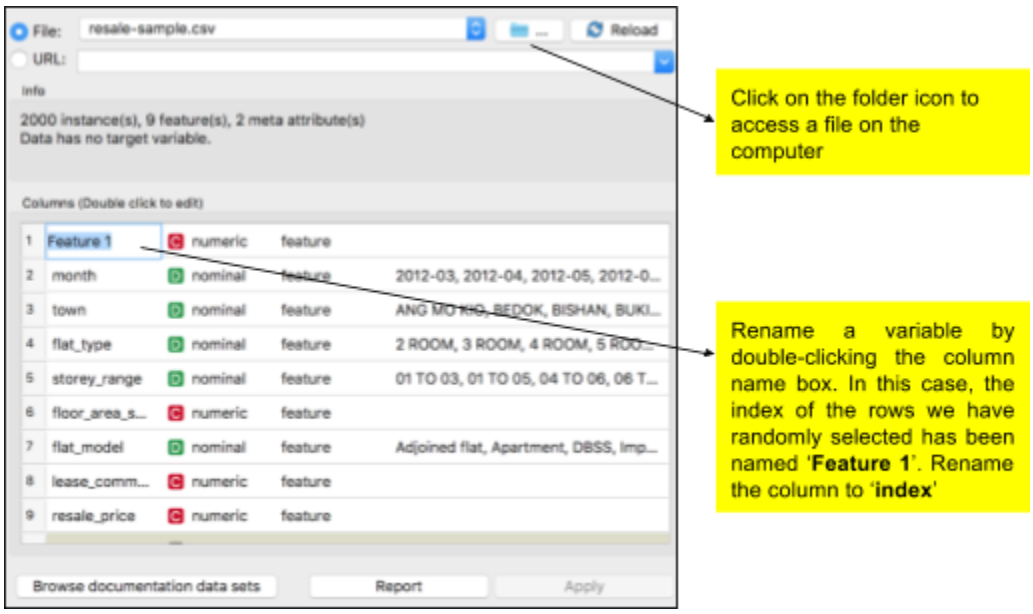


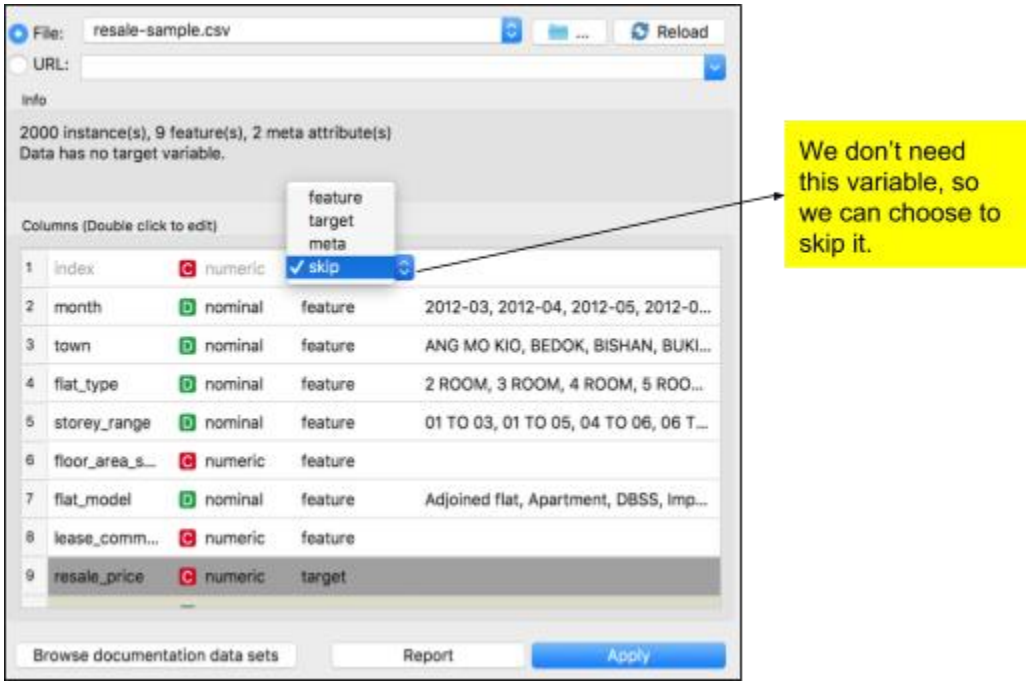
Workflow

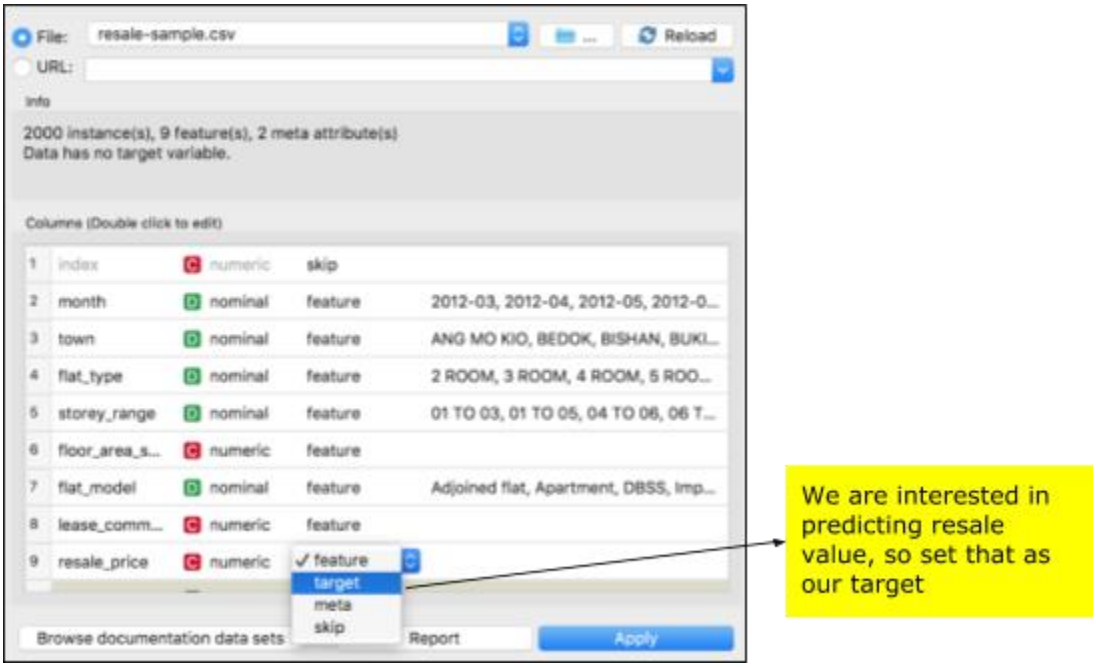
1. Create the following workflow.



2. We will use a scaled-down version of the *resale flat prices* dataset we looked at previously. From the original dataset of 10,000 rows, we randomly sample 2000 rows. This is mainly to speed up compute time in Orange.



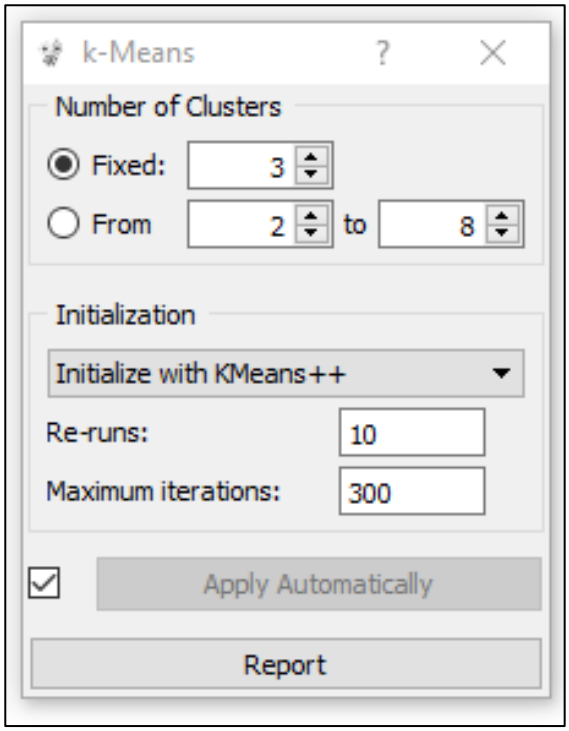




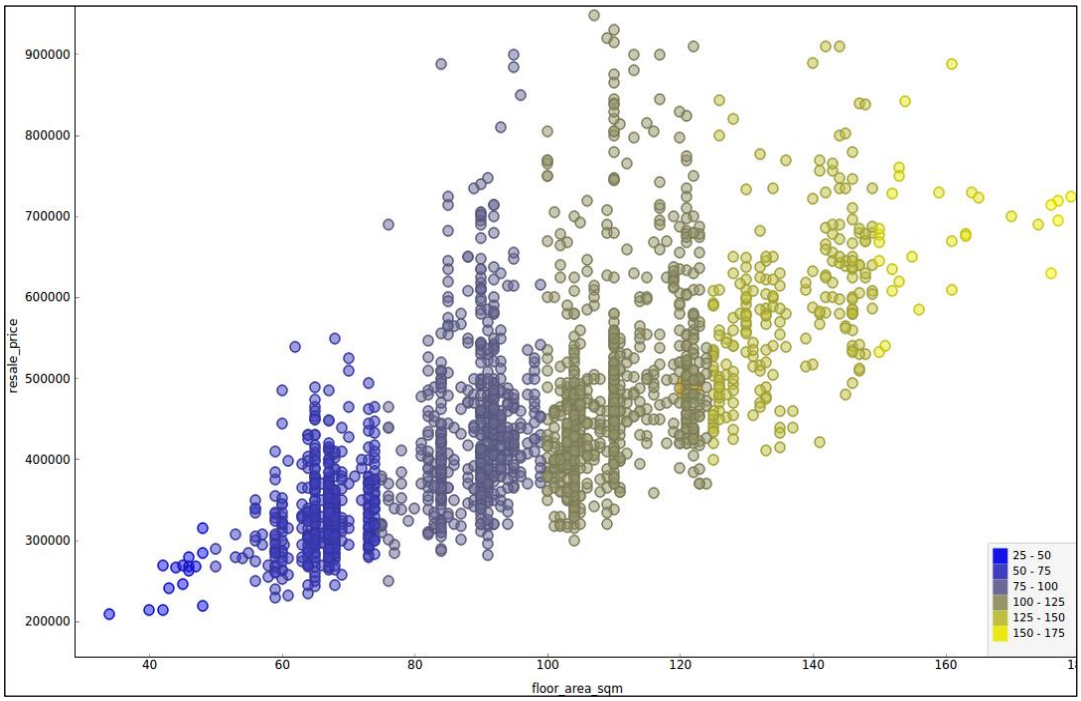
3. This is what is known as some simple ***data cleaning***

Explore

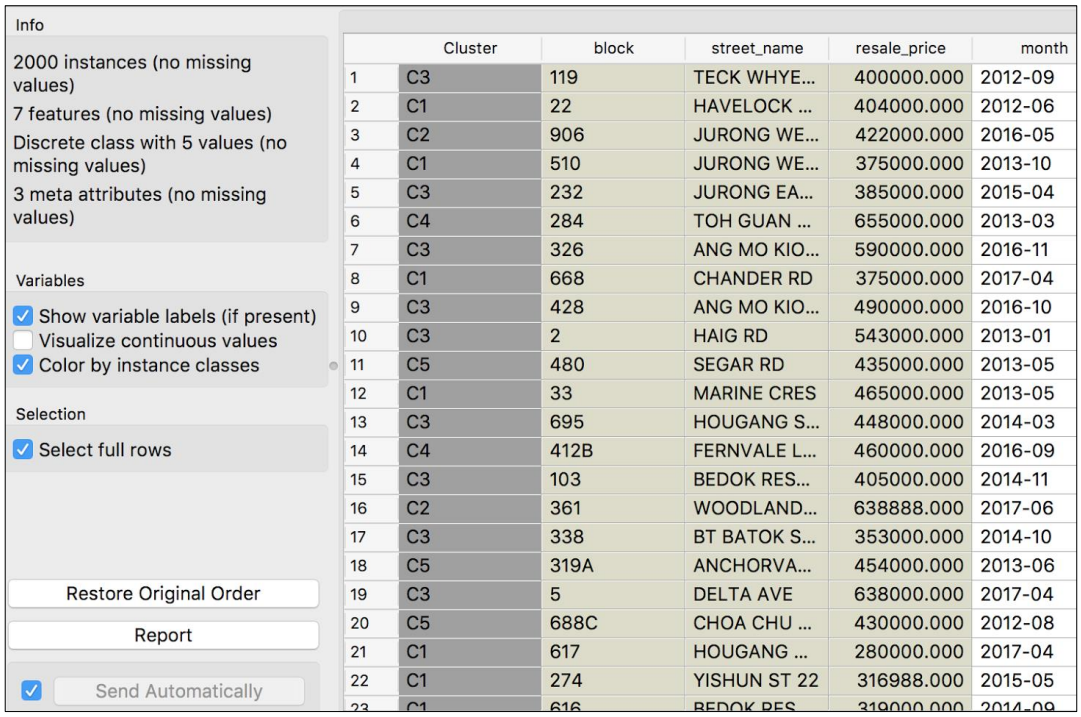
1. Open the k-means widget. These are some of the parameters the k-means algorithm work with. The most important is the number of clusters, that is, how many categories there are in the data. You can leave the rest (Initialization, Re-runs, Maximum iterations as their defaults).



2. Our visualization seems to suggest 5 clusters, as seen by the 5 colours in the graph (this can be subjective). We can test our intuition by setting number of clusters to be five and checking the k-means algorithm results.

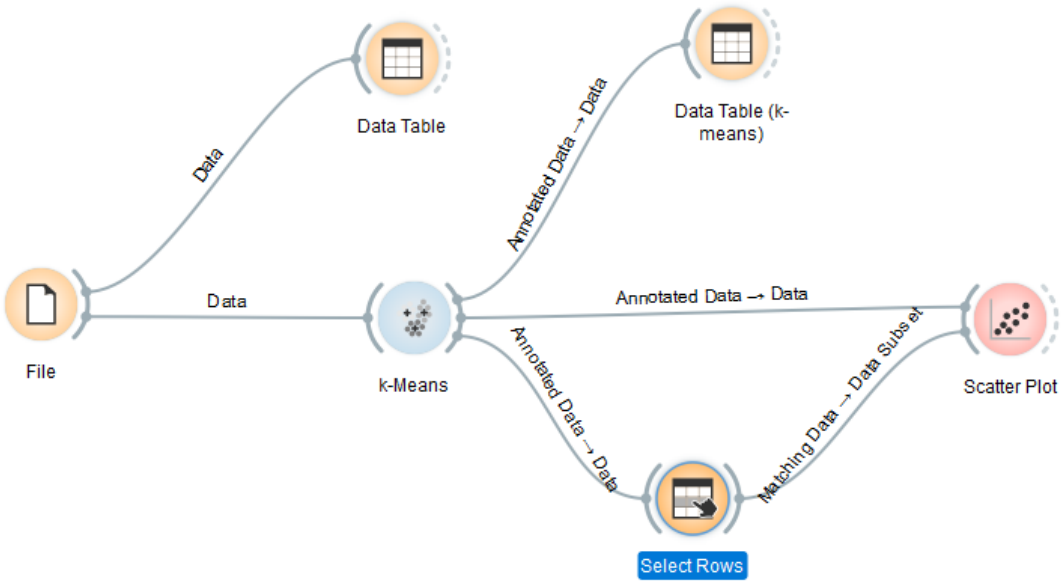


3. Review the **Data Table (k-means)** widget. This shows the scored results. The Cluster column shows the assigned cluster after learning.

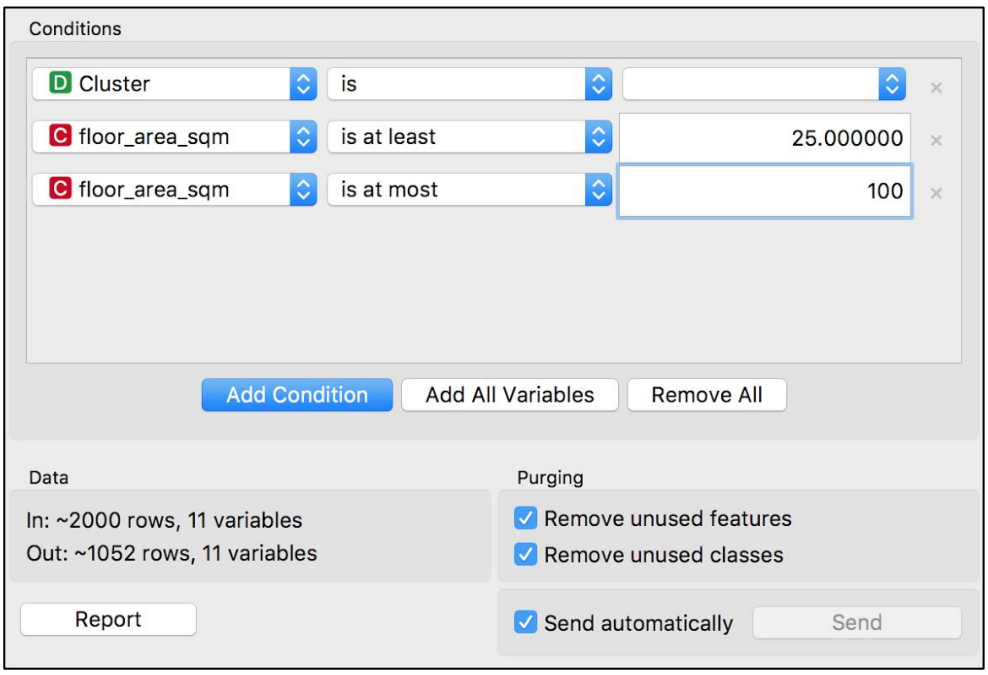


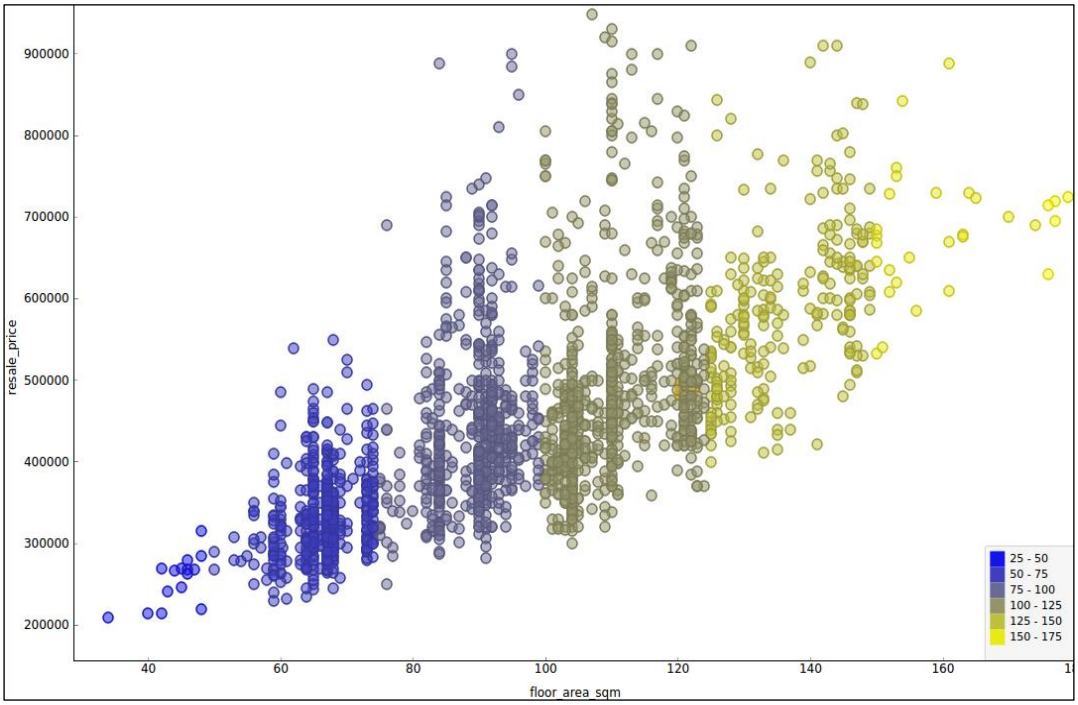
	Cluster	block	street_name	resale_price	month
1	C3	119	TECK WHYE...	400000.000	2012-09
2	C1	22	HAVELOCK ...	404000.000	2012-06
3	C2	906	JURONG WE...	422000.000	2016-05
4	C1	510	JURONG WE...	375000.000	2013-10
5	C3	232	JURONG EA...	385000.000	2015-04
6	C4	284	TOH GUAN ...	655000.000	2013-03
7	C3	326	ANG MO KIO...	590000.000	2016-11
8	C1	668	CHANDER RD	375000.000	2017-04
9	C3	428	ANG MO KIO...	490000.000	2016-10
10	C3	2	HAIG RD	543000.000	2013-01
11	C5	480	SEGAR RD	435000.000	2013-05
12	C1	33	MARINE CRES	465000.000	2013-05
13	C3	695	HOUGANG S...	448000.000	2014-03
14	C4	412B	FERNVALE L...	460000.000	2016-09
15	C3	103	BEDOK RES...	405000.000	2014-11
16	C2	361	WOODLAND...	638888.000	2017-06
17	C3	338	BT BATOK S...	353000.000	2014-10
18	C5	319A	ANCHORVA...	454000.000	2013-06
19	C3	5	DELTA AVE	638000.000	2017-04
20	C5	688C	CHOA CHU ...	430000.000	2012-08
21	C1	617	HOUGANG ...	280000.000	2017-04
22	C1	274	YISHUN ST 22	316988.000	2015-05
23	C1	616	BEDOK RES...	319000.000	2014-09

4. Add the Select Rows widget to the workflow as shown

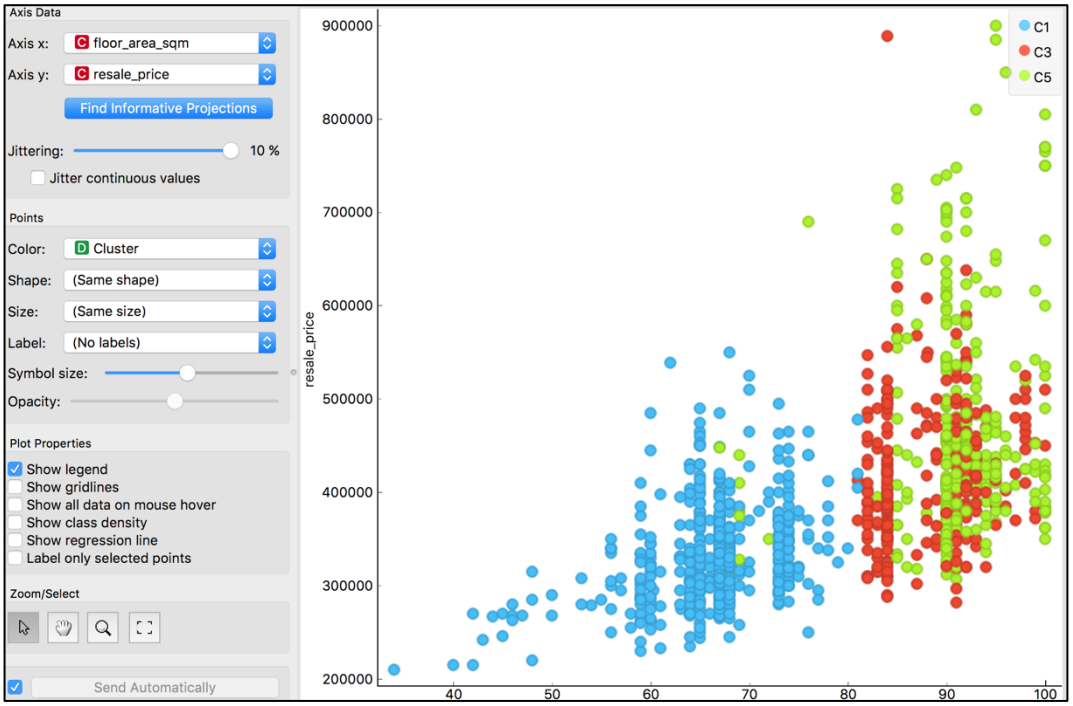


5. The Select Rows widget provides an easy way for you to explore the dataset further (to help you understand the data better!). Have both the Select Rows dialog and Scatter Plot open side by side. Try to change or add conditions and see what happens.





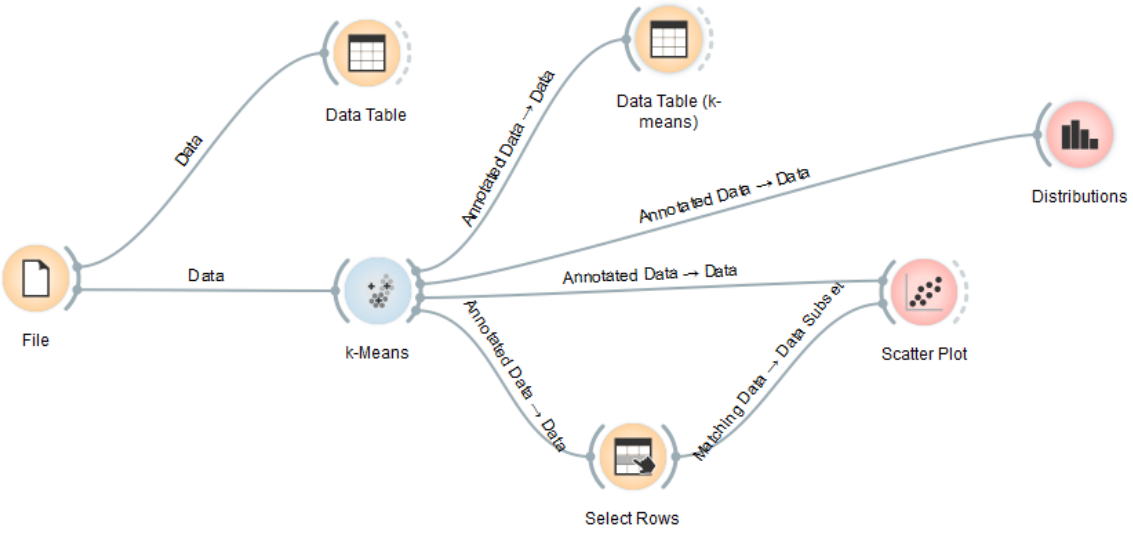
6. From the graph above, we can see that apartments with *floor area sqm* between 25 – 100 sqm seem to fall into two clusters. Looking at the scatter plot generated, however, and we find that there seem to be 3 clusters instead of 2. This might be worth investigating further if we have time.



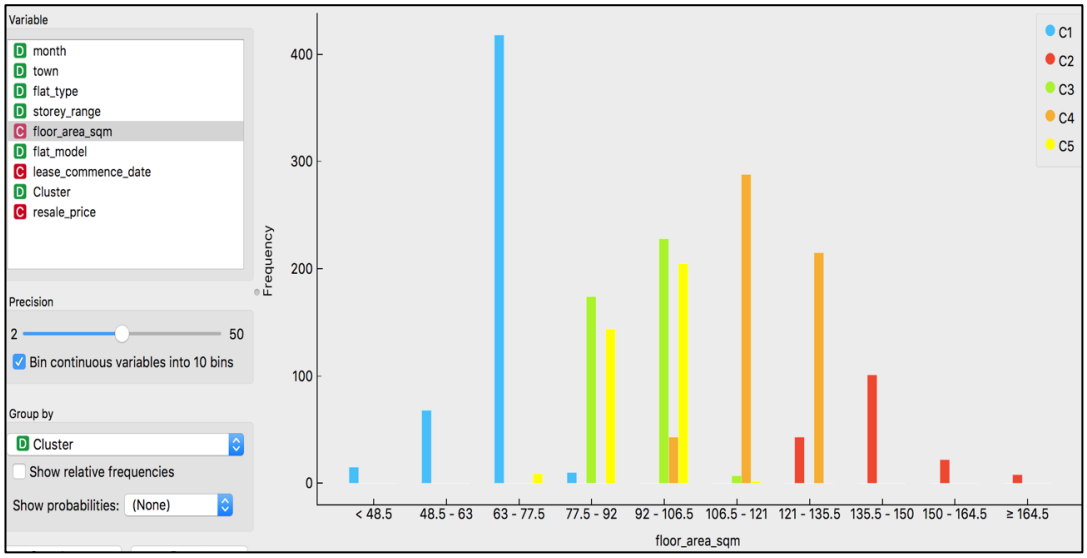
See that Orange highlights only those points which meets the conditions you have set. This interactivity allows you to explore and asks questions about the data and see the results immediately.

Obviously, this works for non-big data set. But in most common business situations, the dataset you have on your desktop can be analyzed without much problem.

7. Explore the data further. Add in a Distribution widget as shown,

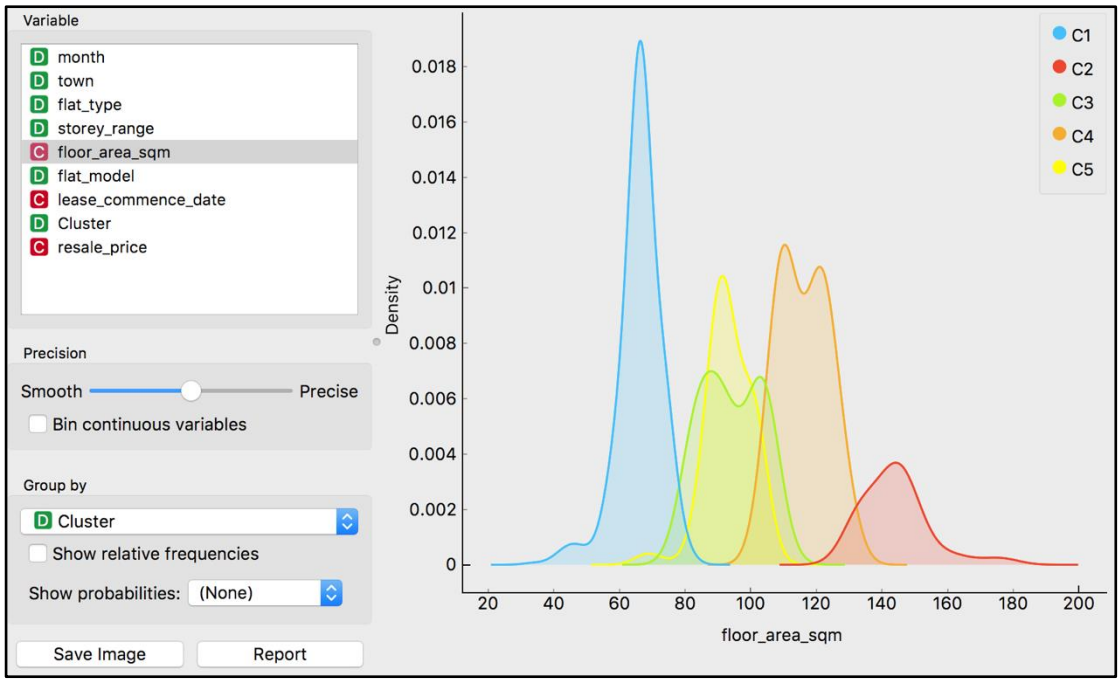


8. Open and explore the Distributions widget.



If you have binning checked, the Distribution widget displays the data points into the number of bins you have selected. Binning is a nice way to group range of continuous variables together to give you a sense of how the data is distributed.

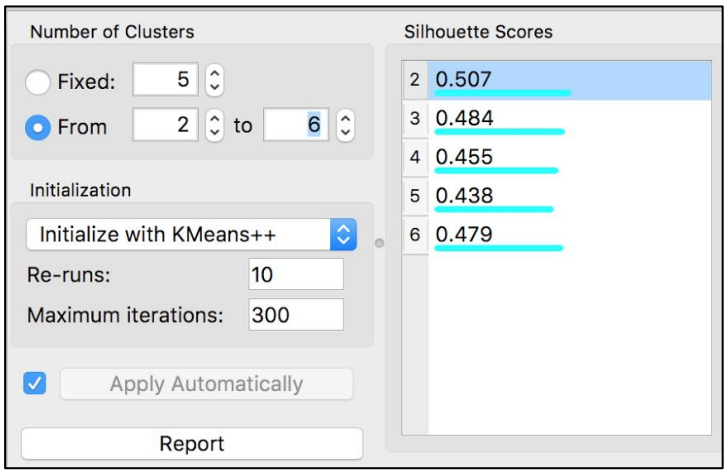
Uncheck the Bin continuous variable option. See that you get the usual normal distribution bell curve. It's not clear whether there are four or five clusters.



Data science is very much data exploration as it is algorithms and models. The ability to manipulate data and explore the underlying nature of the data is key to building a good model.

Orange provides a nice set of tools for such data exploration.

9. Now, click on the K-means dialog and select the radio button to indicate the range of number of clusters. Recall our earlier uncertainty about whether four or five (or less) clusters best describe the data. So we explore all by setting number of clusters from 2 to 6.

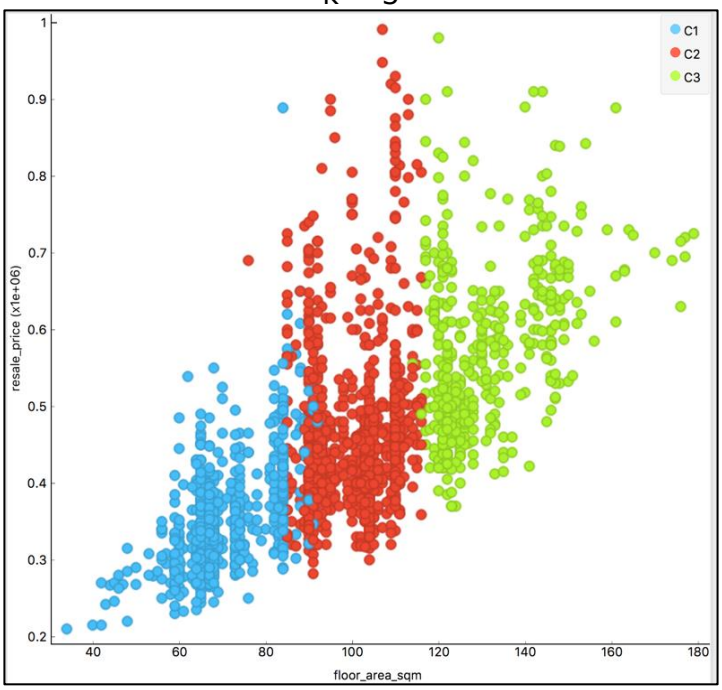


There does not seem to be much difference between silhouette scores. This might mean the data is not easily separable. In such cases, going with an easily understandable number eg. 4, that corresponds to our earlier visualization works.

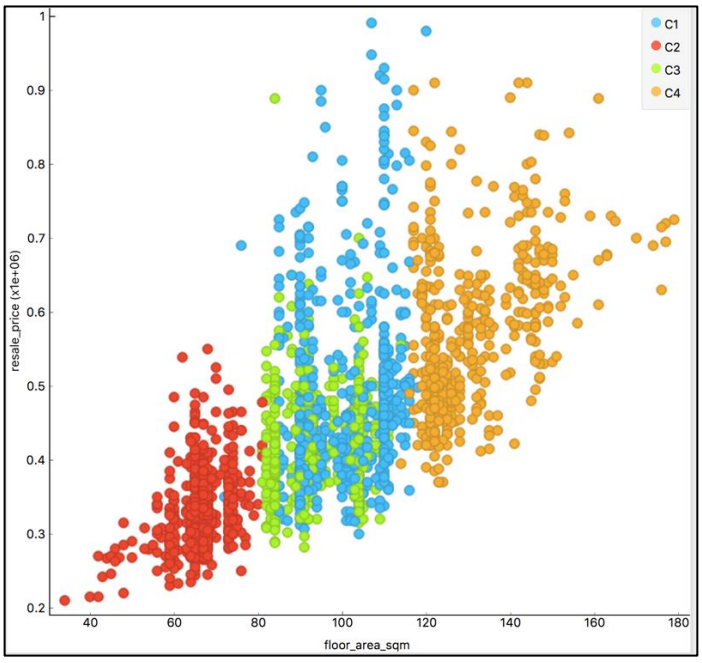
10. Recall, the k in k-means refers to the number of clusters in the dataset. Often you do not know how many clusters there are, and typically a good guess would be in the range 2 to 8. Anything more may suggest the use of further preprocessing of the data (PCA to reduce the dimensions) or to use a different algorithm altogether.

11. View the cluster with different values of k.

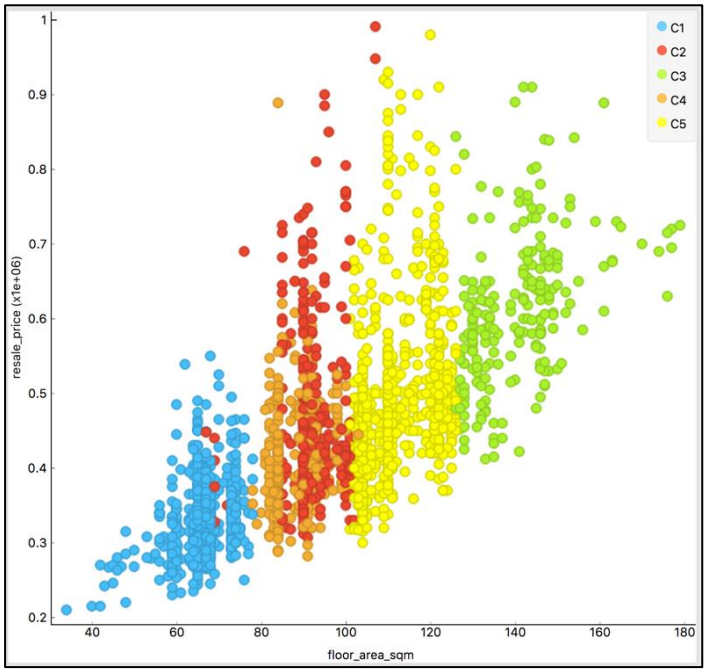
k = 3



k= 4



k = 5



Conclusion

k-Means is a popular unsupervised learning algorithm. Use of k-Means comes with caveats.

K-Means algorithm works best for nicely packed clusters where the center of the cluster is equal-distance from the cluster points.

For the iris dataset, the C2 and C3 data points were elongated, so k-Means could not easily distinguish there were 2 classes in that segment.

So, for k-Means to work well (automatically)

a. Data distribution needs to be in round/spherical clusters

b. The input data needs to be in the same order of magnitude (or range of values). That is, you should normalize the value to the same order of magnitude to allow k-Means to work better, e.g.:

    i. Raw: Height = 1.60 – 1.80m and weight = 45 – 100kg

    ii. Normalized:

        1. Height = 0.89 – 1.0 (height/max-height)
        2. weight = 0.45 – 1.0 (weight /max-weight)

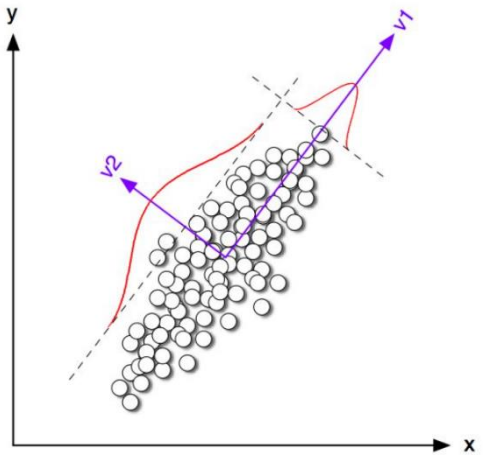
This normalization to get values into the same order of magnitude is important for k-means to work properly.

PRINCIPAL COMPONENT ANALYSIS

Motivation

Real world datasets can be of high dimension and/or correlated and hence hard to visually see the patterns. Principal Component Analysis (PCA) can be used to reduce the number of dimensions or variables by transforming the original variables to the **linear combination of these variables which are independent**!

Theory



Source: Lazy Programmer + Wikipedia

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of **linearly uncorrelated and independent** variables called principal components. Independent here means the data are orthogonal (90degrees) to each other in the graph above.

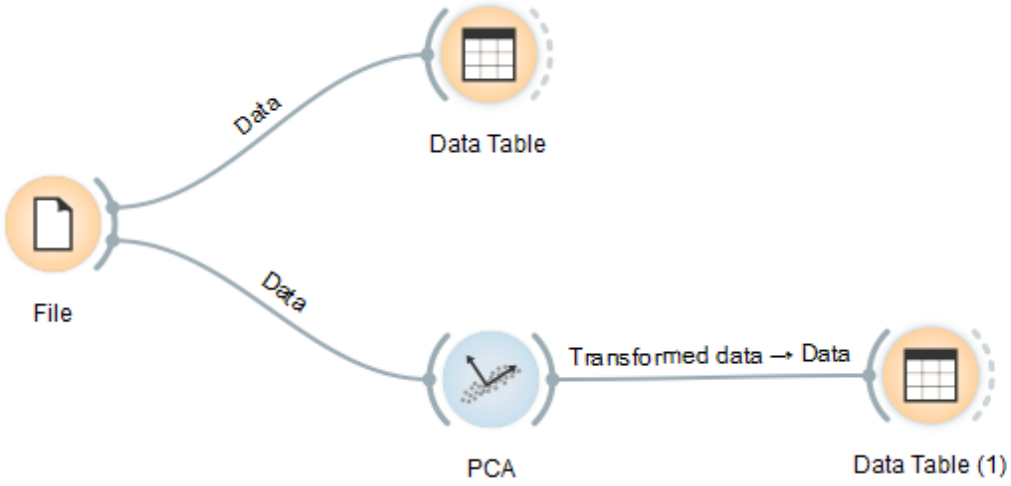
PCA reduces dimensionality by moving as much "information" as possible into as few dimensions as possible. PCA is sensitive to the relative scaling of the original variables (similar to k-Means).

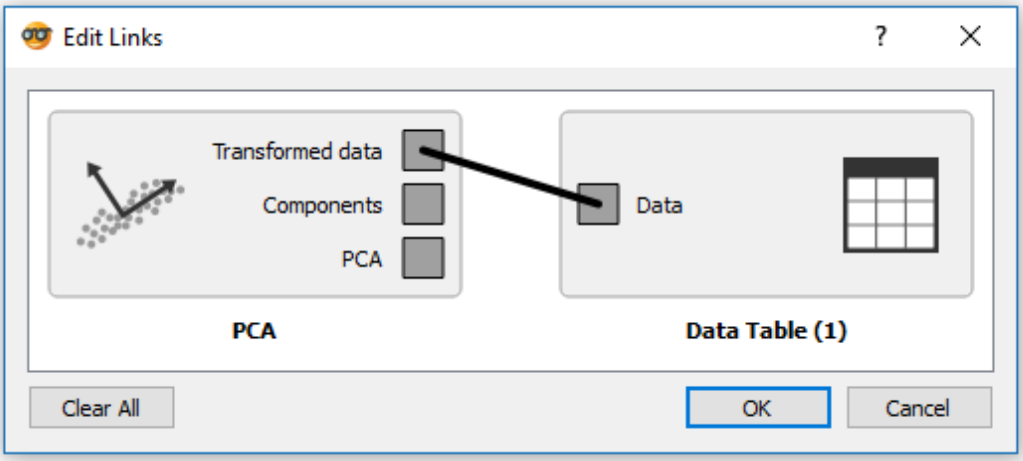
PCA is mostly used as a tool in exploratory data analysis and for making predictive

models.

Workflow

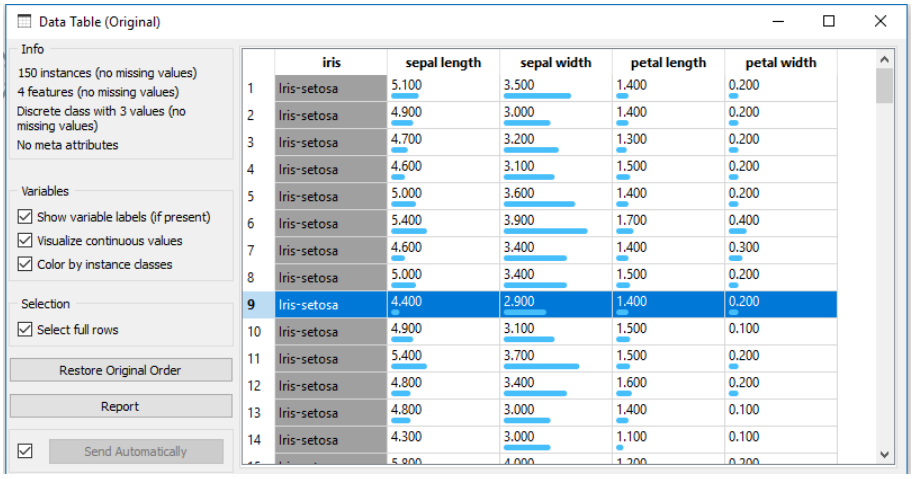
1. Create a new workflow as shown. Load up the famous *iris.tab* dataset.





The **PCA** widget to **Data Table (PCA)** widget link allows you to specify the type of information you want to propagate. In the above, we want the **Transformed data** to be send to the **Data Table (PCA)** widget.

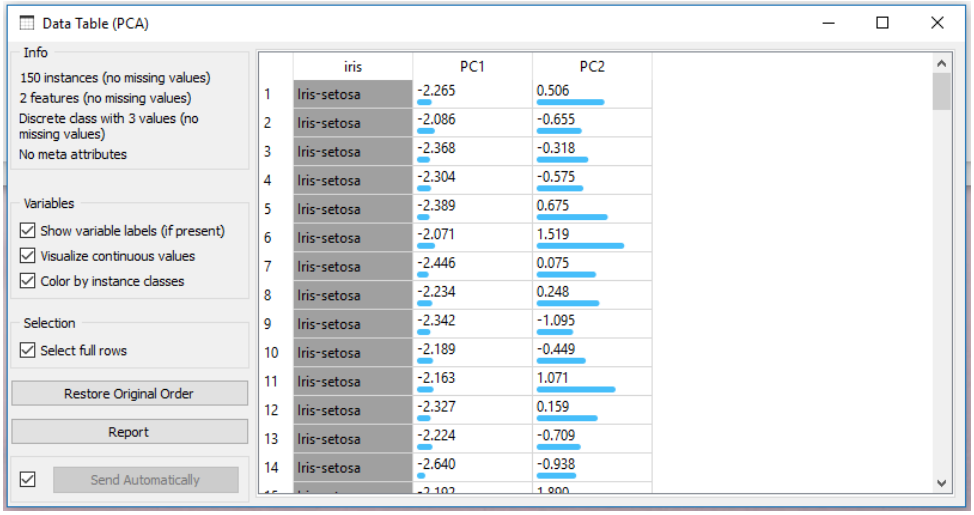
2. The **Data Table (Original)** dataset is shown below. It has 4 features (or variables) and 1 target column. Question: Do we need all 4 features to classify the iris flowers? Imagine storing and only using 2 columns of data compared to 4 columns (half the storage requirements – think big data).



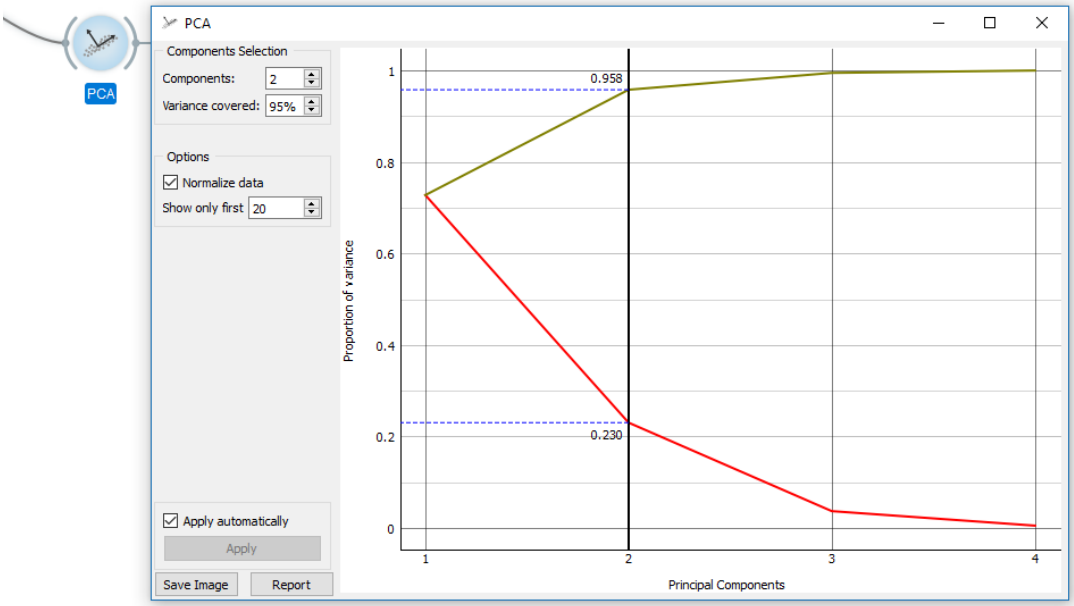
	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100

3. See the **Data Table (PCA)** dataset. Here the **Principal Component Analysis (PCA)** widget computes the PCA linear transformation of the input data. It outputs either a transformed data set with weights of individual instances or weights of principal components.

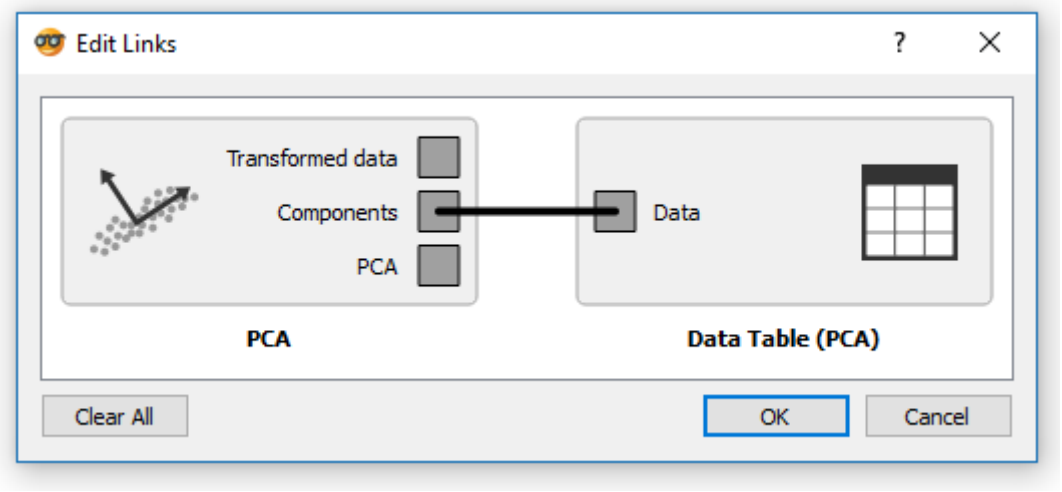
   The widget provides two outputs: transformed data and principal components. Transformed data are weights for individual instances in the new coordinate system, while components are the system descriptors (weights for principal components). When fed into the Data Table, we can see both outputs in numerical form.

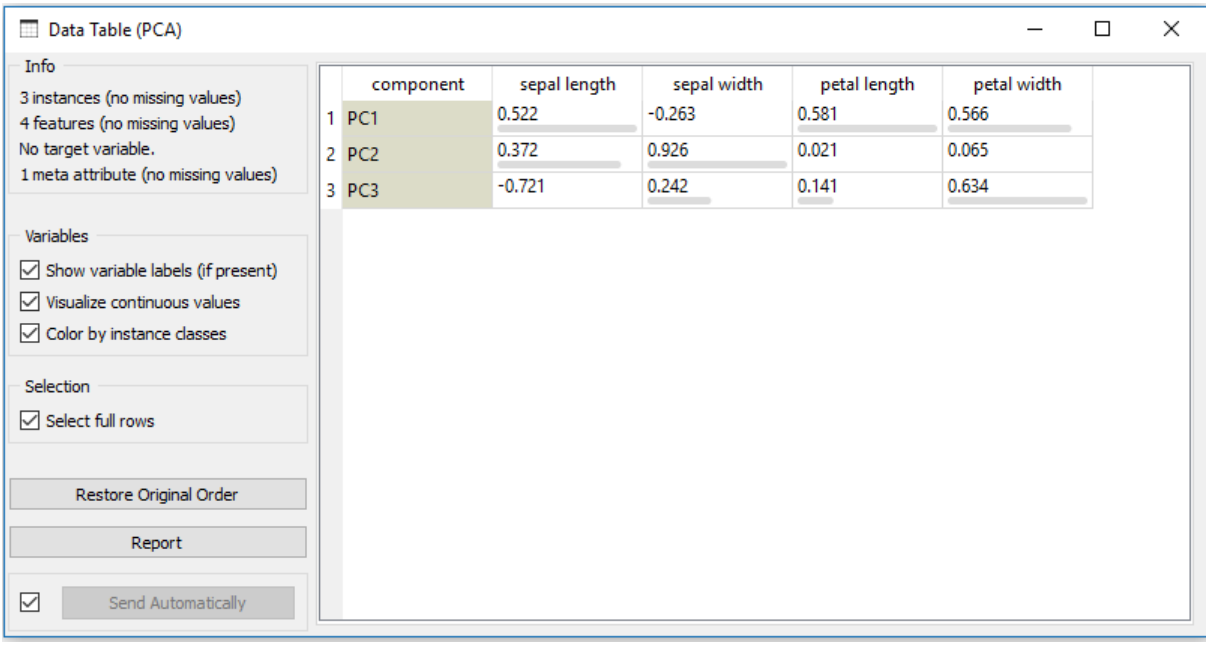


4. But how good is two components? Open the **PCA** widget. See that with 2 components, 95% of the variance can be explained and with 3 components we get 99.5% of the variance are explained. Using all 4 features only gain us an additional 5% and 0.5% respectively. So, is this good enough?



5. Let's explore the PCA data further. Click on the link between the **PCA** widget and **Data Table (PCA)** widget and change the data as follows:

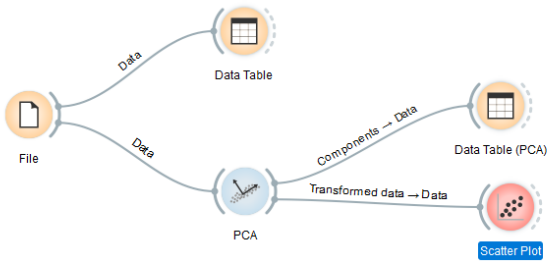


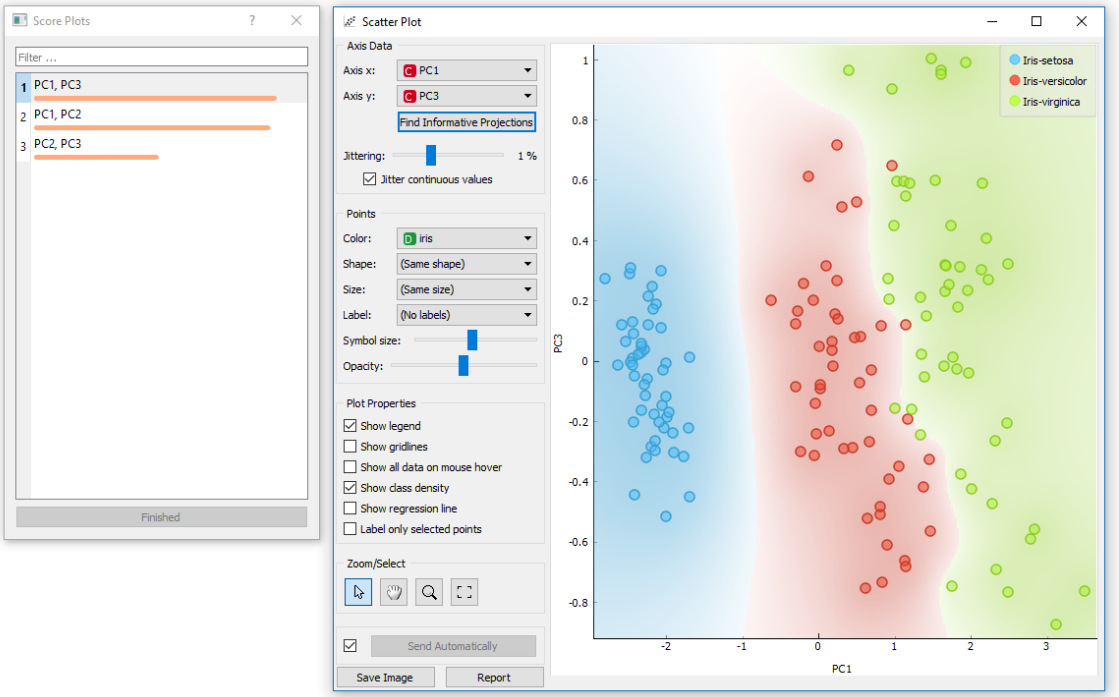


So, PCA has found the following linear model which make PC1 linearly independent of its features (variables)

PC1 = 0.522.sepal_lenght -0.263.sepal_width + 0.581.petal_length + 0.566.petal_width

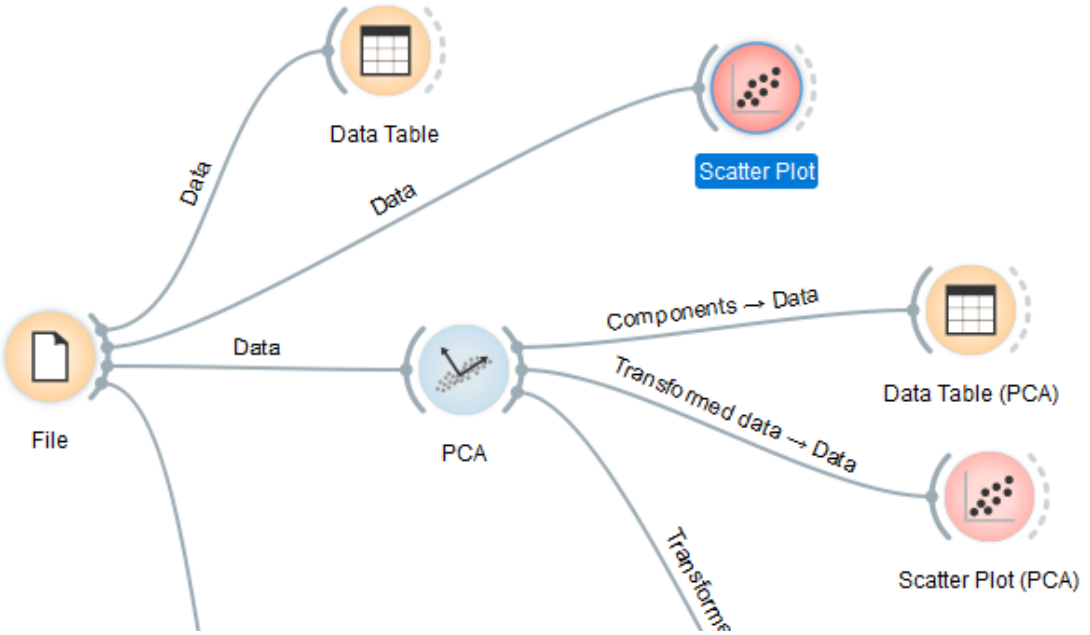
6. Add Scatter Plot widget and explore the data further. Use the **Find Informative Projections** function to find the most useful plots to view.

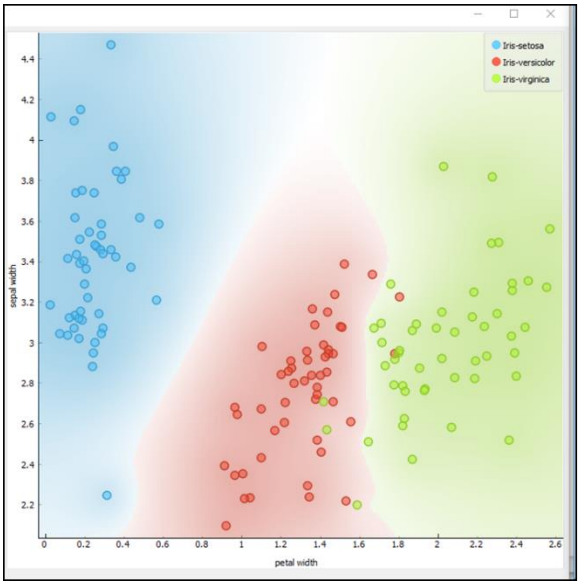


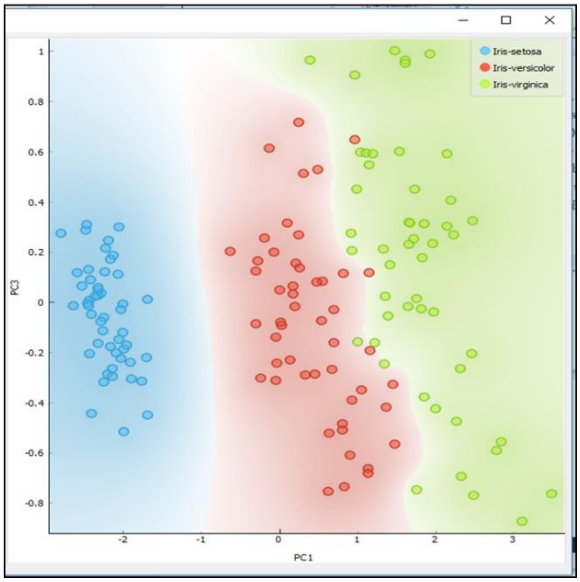


See that in the above plots there is now less misclassifications and the three iris flowers are cleanly separated.

7. Compare the outputs of both the original and PCA data with the two **Scatter Plot** widgets.

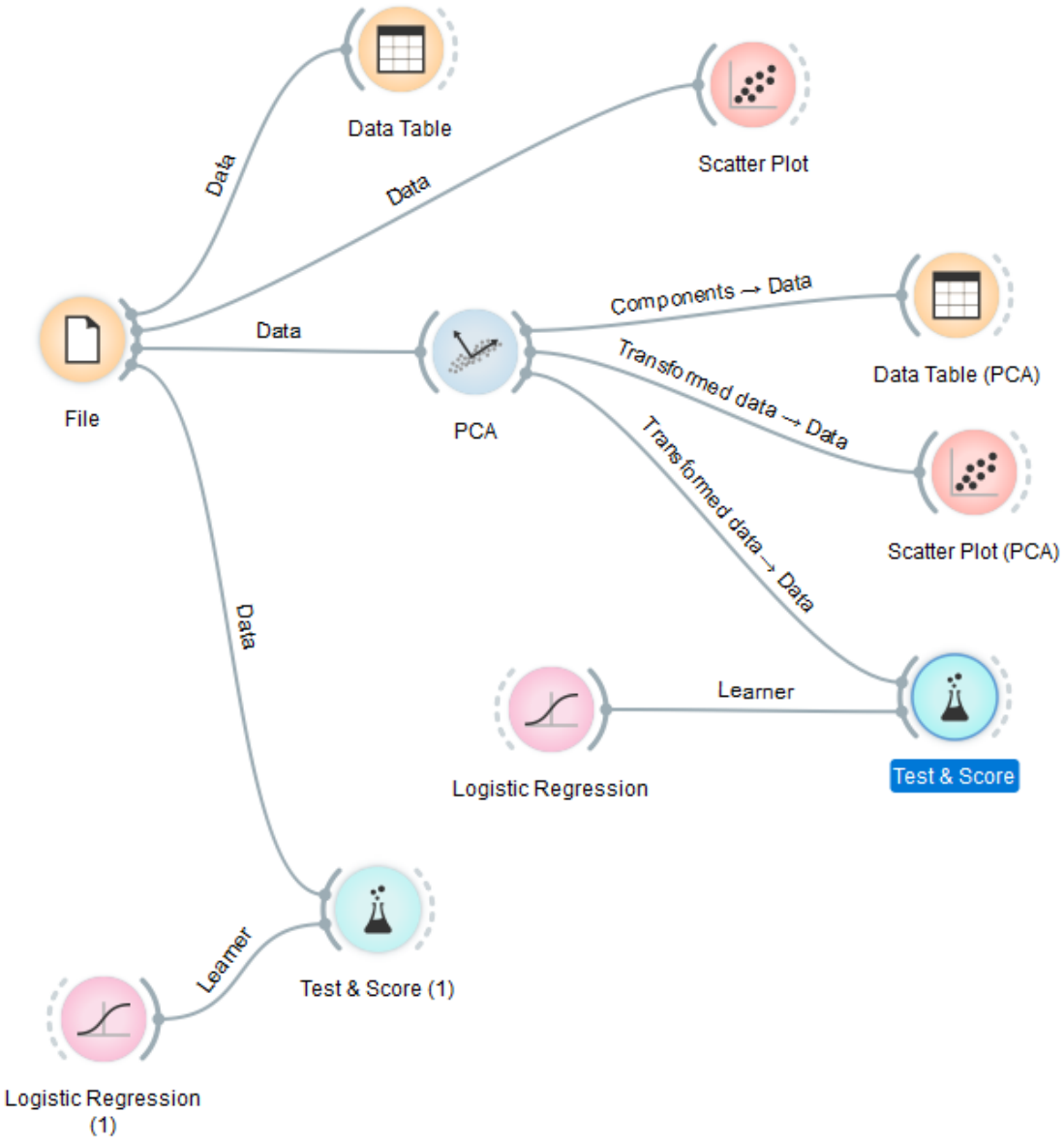




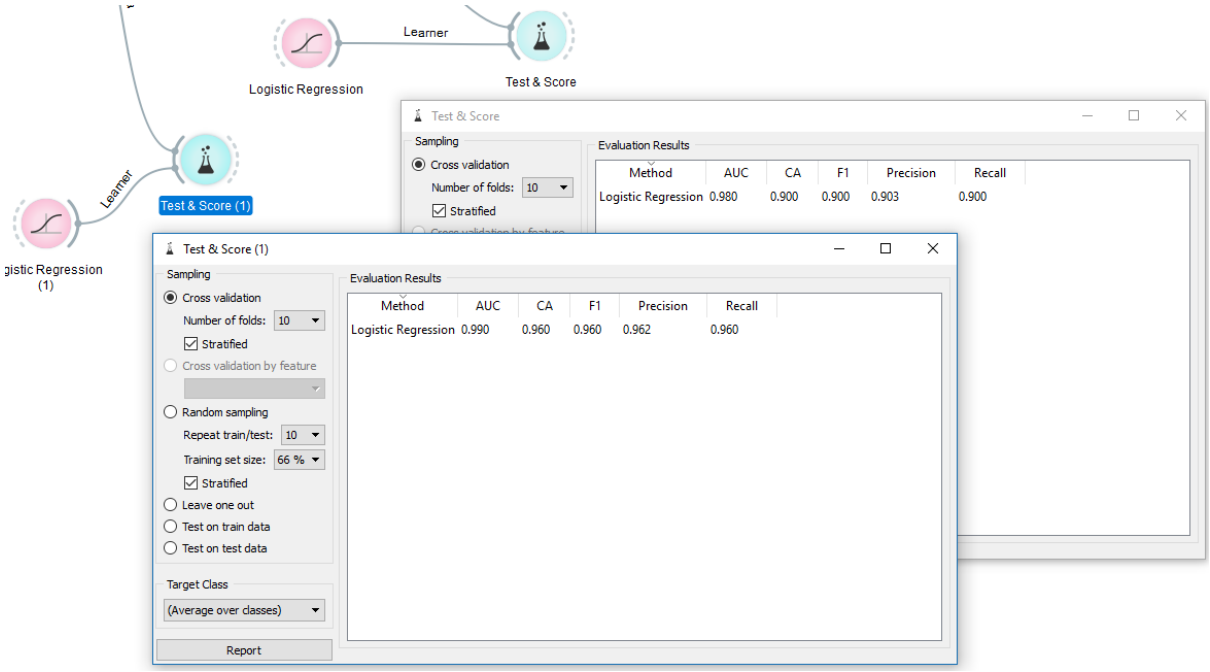


See the BEFORE and AFTER transformation plots.

8. What about the performance of the models with and without PCA transformation? Add two sets of **Test & Score** and **Logistics Regression** widgets as shown below.



9. Compare and discuss the accuracies.



Conclusion

You have used the **PCA** widget to explore the data and reduced the dimensions of the dataset (feature engineering).

In this case, the transformation by PCA affected the accuracy of the models. Simplicity comes at a cost and often in data science you must weigh the costs of better accuracy versus computational costs.

ASSOCIATION RULES (ARULES)

Motivation

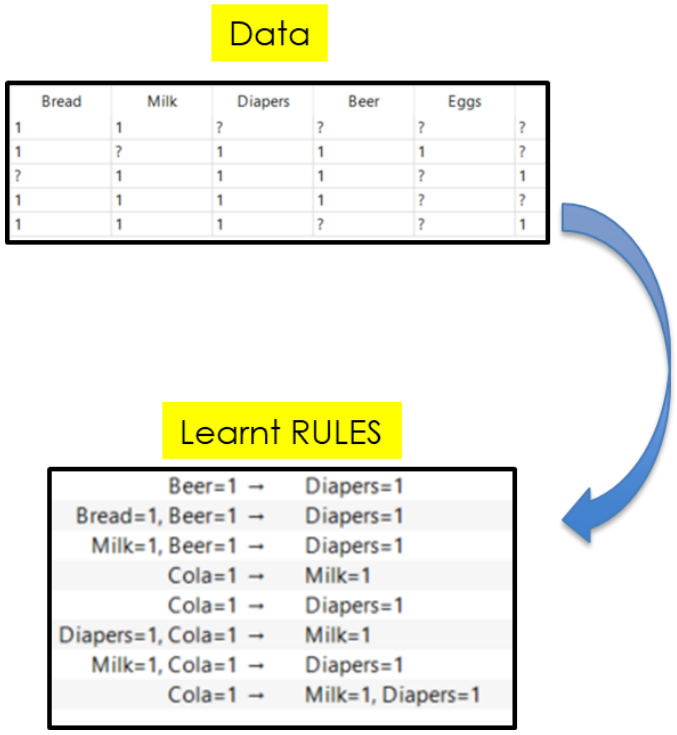
Human readable explanation of how the model behaves brings significant benefit as it would allow the data science team to make recommendations in layman terms which can be easily understood.

Associate Rules is a common use in many e-commerce and retail analysis. It answers the question of "What are the associated items the customer buys when he buys milk."

Theory

Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases.

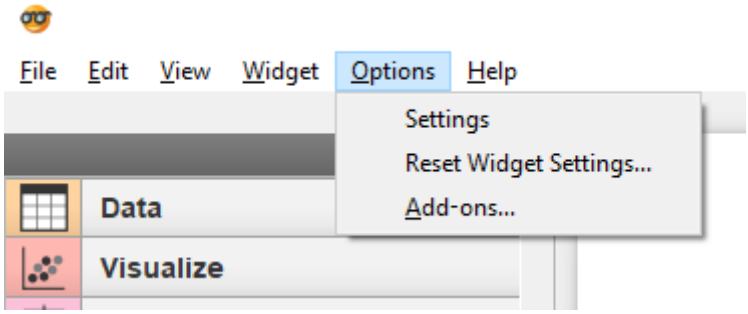
It is intended to identify strong rules discovered in databases using some measures of interestingness. There is no "target" in the dataset!

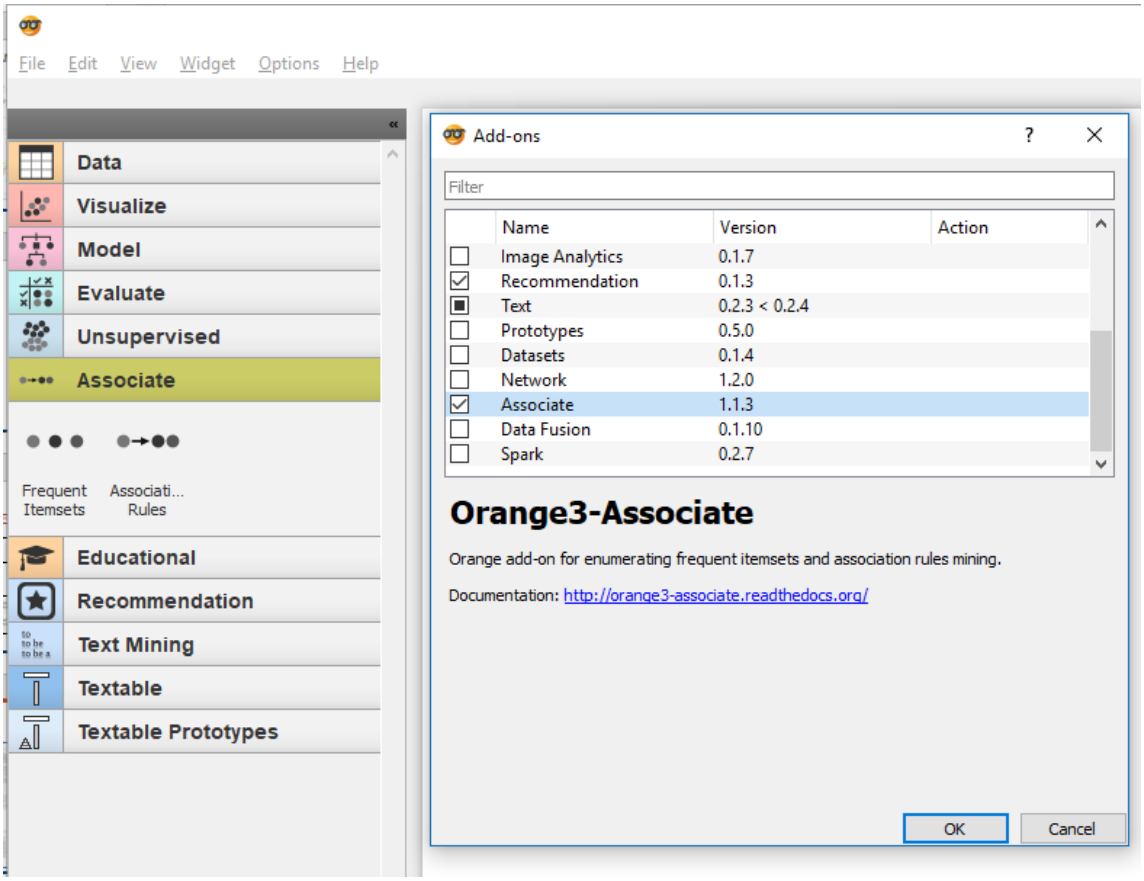


Workflow

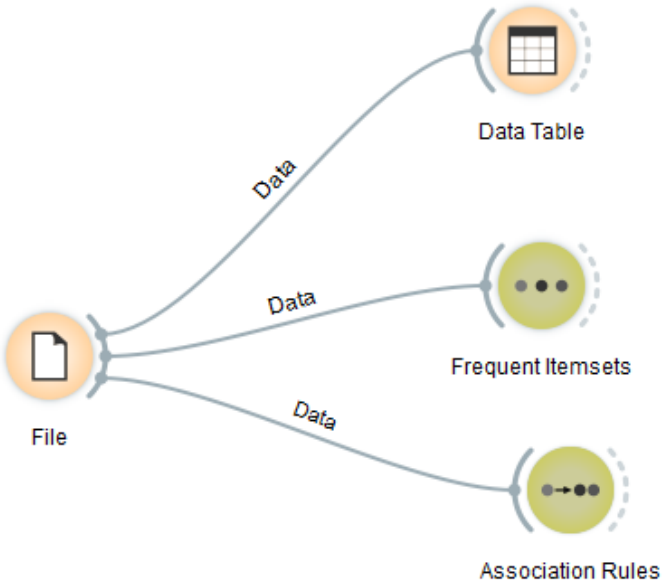
1. The Association Rules widget is not part of the Orange default widgets. It is an Add-on which you can download and install easily. Click on the **Options** menu and select **Add-ons**..

   Look for **Associate** add-on, select it and click **OK**.

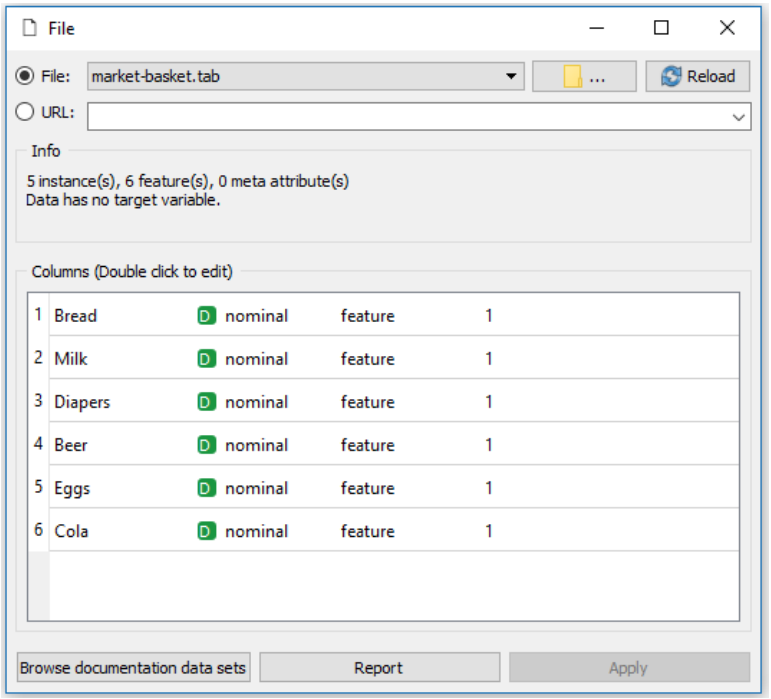




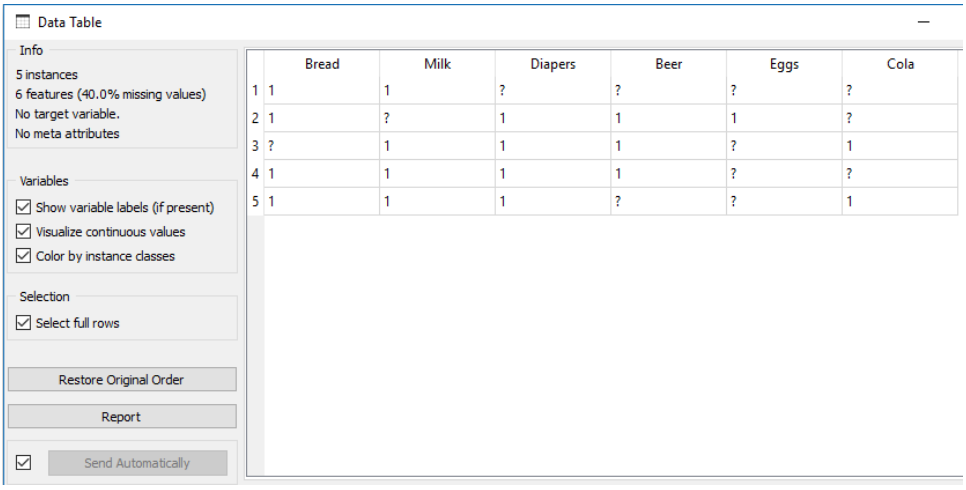
2. Build the following workflow.



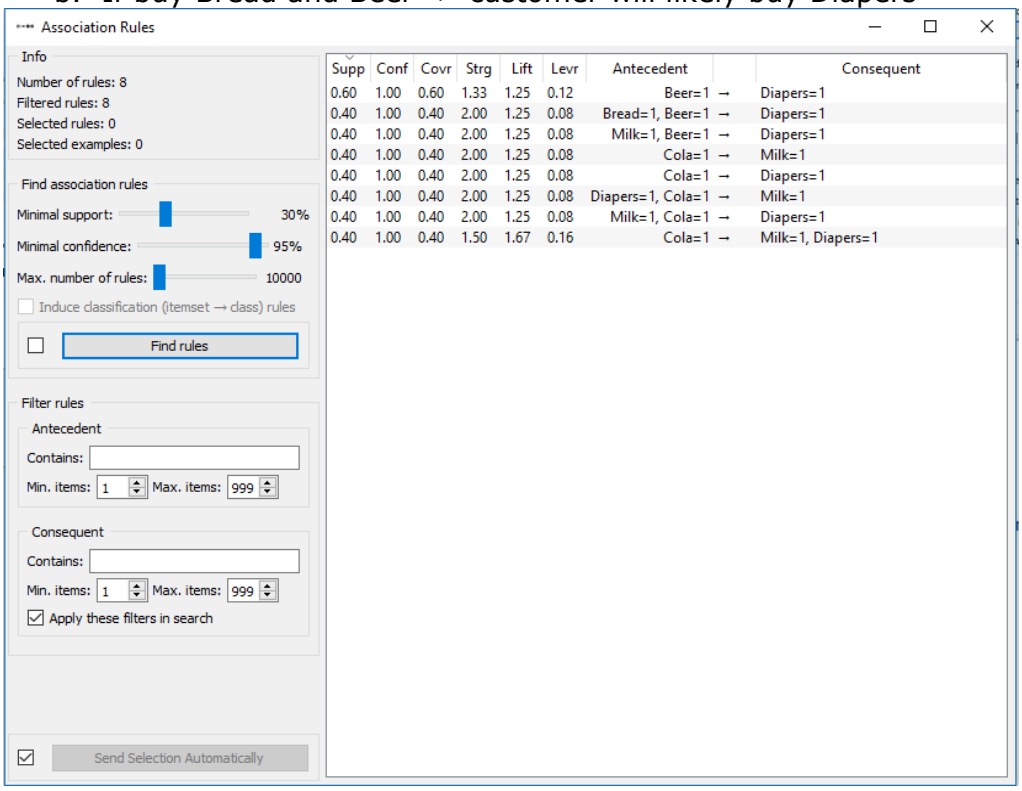
3. Use the following dataset: market-basket.tab



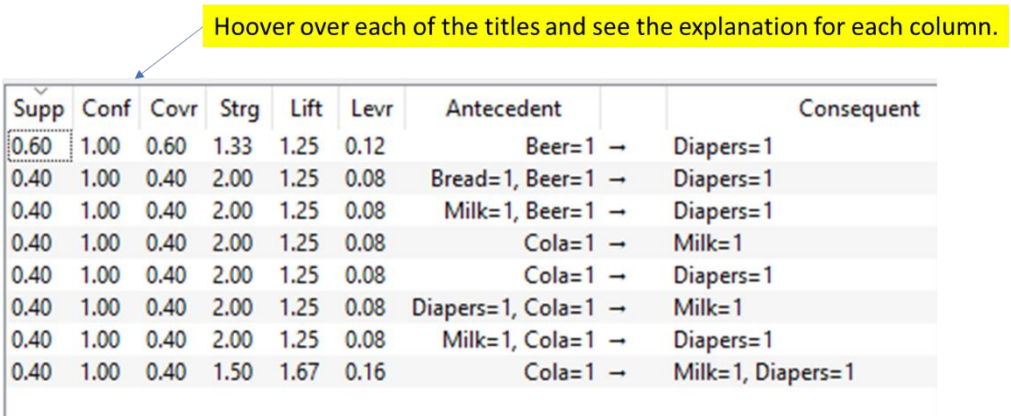
View the dataset.



4. Open the **Association Rules** widget. The rules generated include:
    a. If buy Beer -> customer will likely buy Diapers
    b. If buy Bread and Beer -> customer will likely buy Diapers

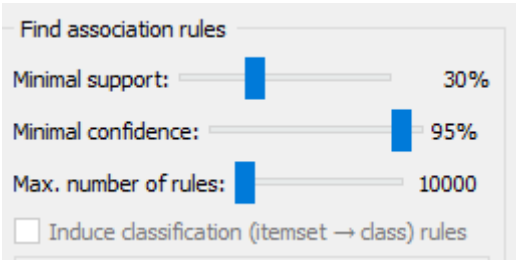


5. What is the support, confidence or coverage levels of the rules?



Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.60	1.00	0.60	1.33	1.25	0.12	Beer=1	→	Diapers=1
0.40	1.00	0.40	2.00	1.25	0.08	Bread=1, Beer=1	→	Diapers=1
0.40	1.00	0.40	2.00	1.25	0.08	Milk=1, Beer=1	→	Diapers=1
0.40	1.00	0.40	2.00	1.25	0.08	Cola=1	→	Milk=1
0.40	1.00	0.40	2.00	1.25	0.08	Cola=1	→	Diapers=1
0.40	1.00	0.40	2.00	1.25	0.08	Diapers=1, Cola=1	→	Milk=1
0.40	1.00	0.40	2.00	1.25	0.08	Milk=1, Cola=1	→	Diapers=1
0.40	1.00	0.40	1.50	1.67	0.16	Cola=1	→	Milk=1, Diapers=1

6. The **Find association rules** option panel provide you the ability to fine tune the number of rules and coverage of the dataset. Try changing the **Minimal support** to 80%. See what rules are generated. Can you explain why the output is as such?

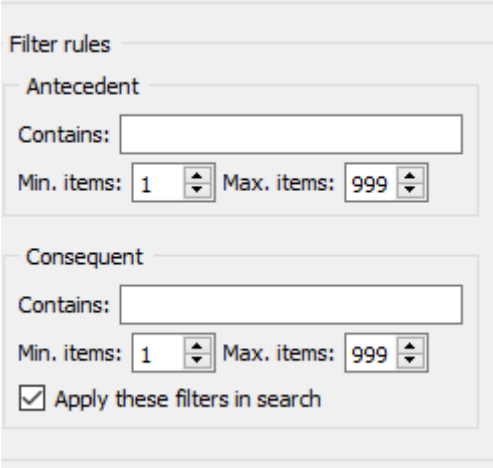


Minimal support: percentage of the entire data set covered by the entire rule (antecedent and consequent).

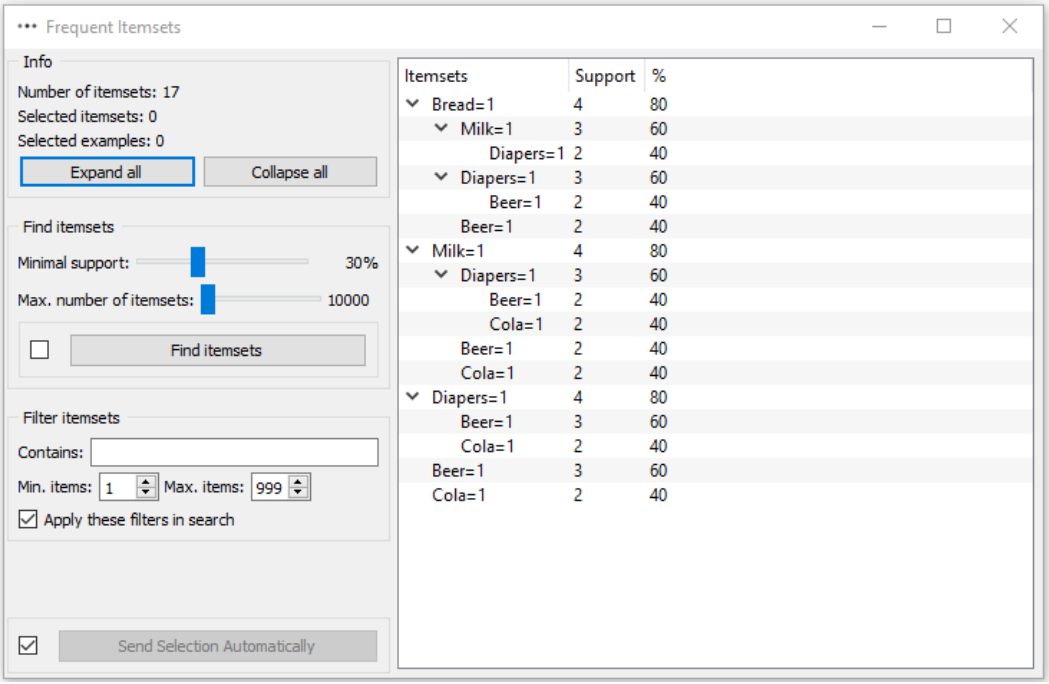
Minimal confidence: proportion of the number of examples which fit the right side (consequent) among those that fit the left side (antecedent).

Max. number of rules: limit the number of rules the algorithm generates. Too many rules can slow down the widget considerably.

7. When the dataset is large, it is useful to filter and see only the rules you are interested in. Try!



8. Open the **Frequent Itemsets** widget. It shows the Support level (percentage of the entire data set covered by the entire rule (antecedent and consequent).



Conclusion

The Associate Rules is a very popular algorithm to find matching itemsets and generate human understandable rules. It is commonly used by retail folks, e-commerce and loyalty card players.

In this section, you have learnt how to:

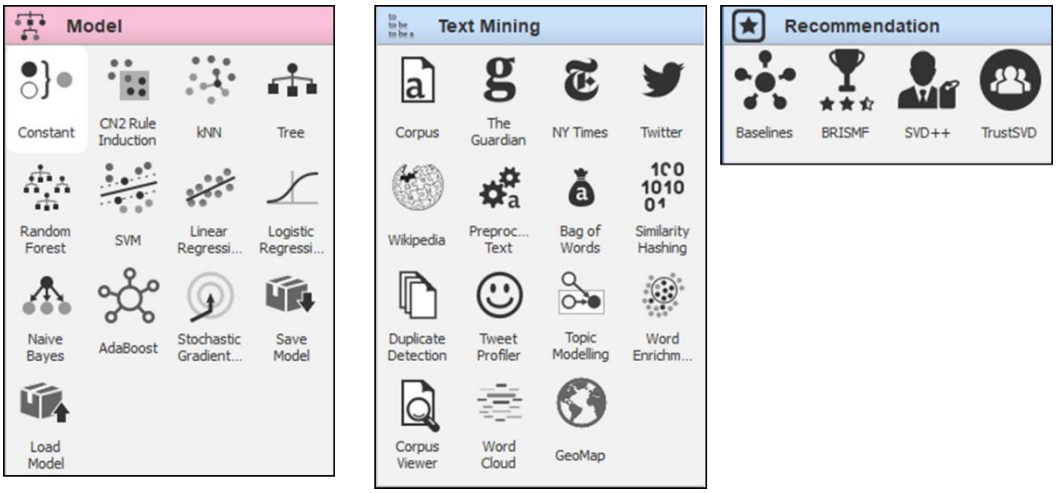
1. Download and install an add-on Orange package
2. Use the **Associate Rules** widget to generate human-friendly rules.

SUPERVISED LEARNING

Supervised Learning algorithms in Orange

1. Orange has a range of supervised learning algorithms. This is shown below. We will however focus on the following 9 algorithms:

    a. Naïve Bayes
    b. K Nearest Neighbors
    c. Linear Regression
    d. Logistic Regression
    e. Tree, Random Forest
    f. Support Vector Machine
    g. CN2 Rule Induction
    h. Recommendation (Add-on package)
    i. Text Analytics (Add-on package)



2. The selected algorithms are some of the most commonly used algorithms in data science projects. These would often be the algorithms you would use for your projects in the beginning. Often, they are good enough, but if you need more performance or accuracy, some of the newer algorithms may be necessary.

NAIVE BAYES

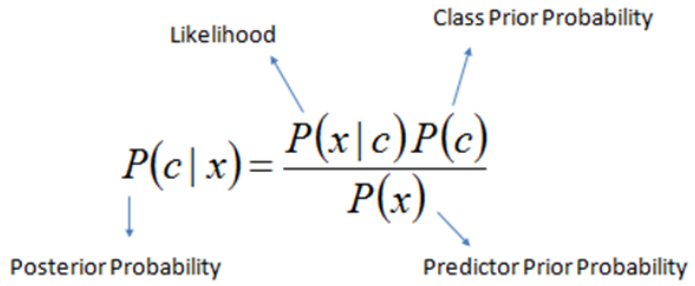
Motivation

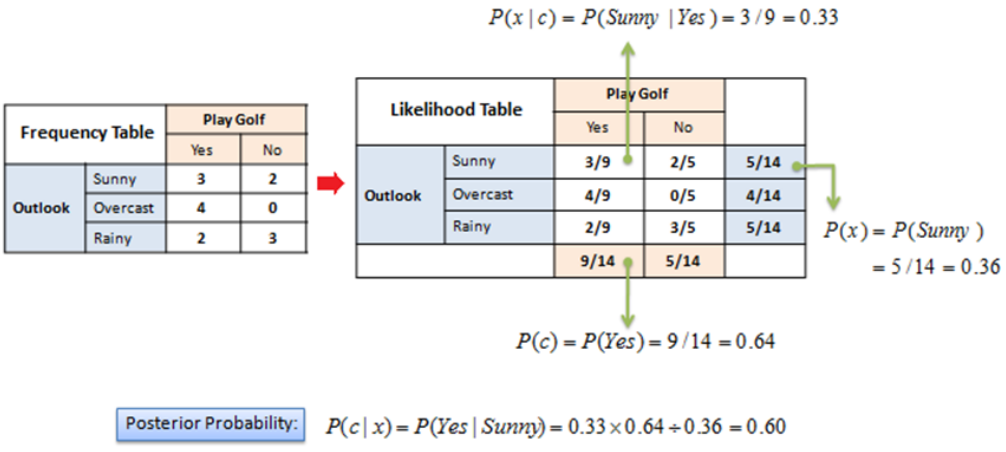
Naïve Bayes would be an algorithm you would have used without knowing. Most popular anti-spam software uses Naïve Bayes as their algorithm or one of several.

It is fast and light weight and gives reasonable results.

Theory

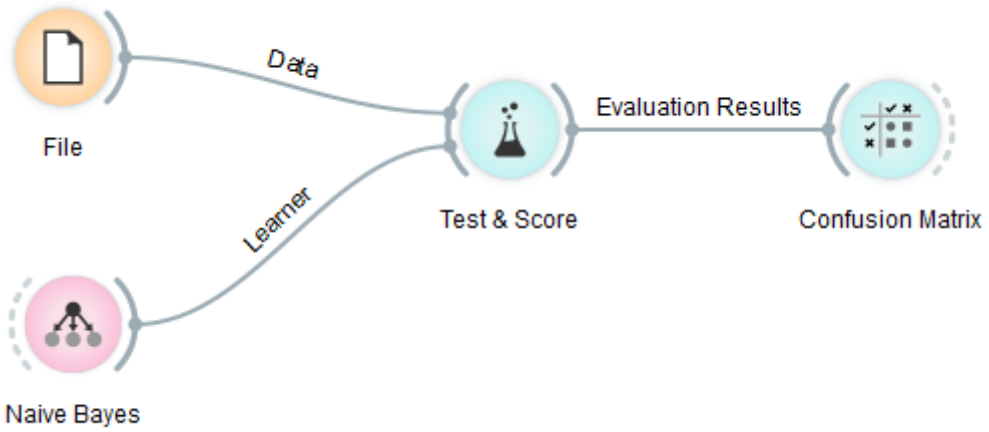
A fast and simple probabilistic classifier based on Bayes' theorem with the assumption of feature independence.



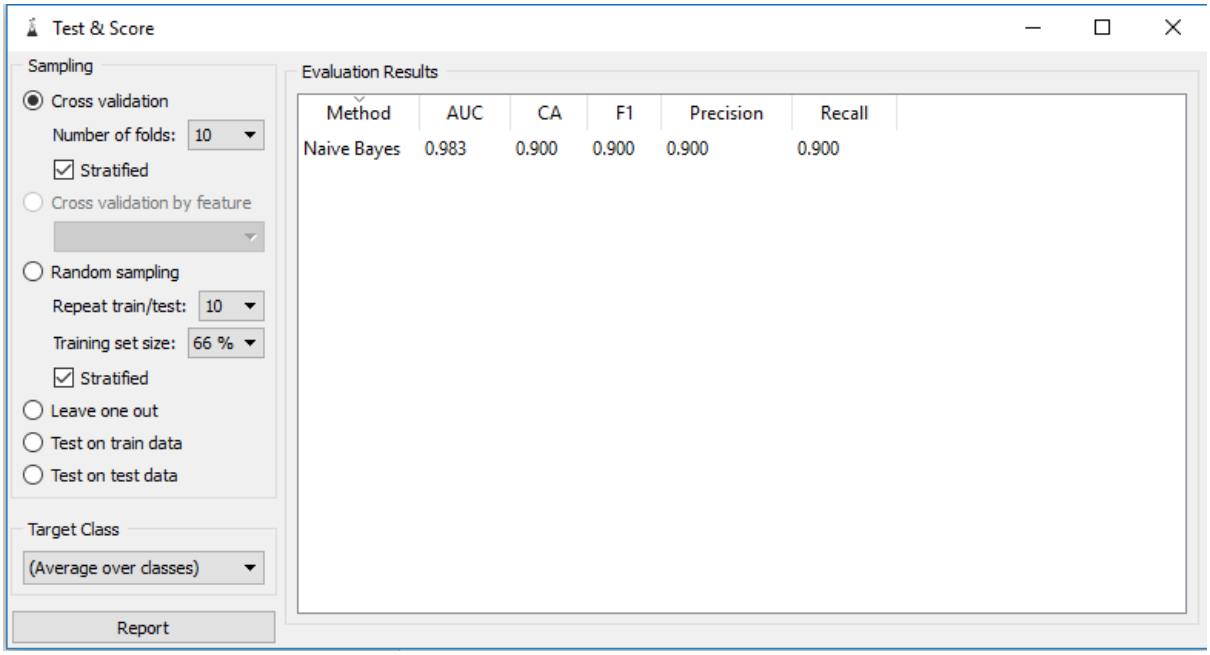


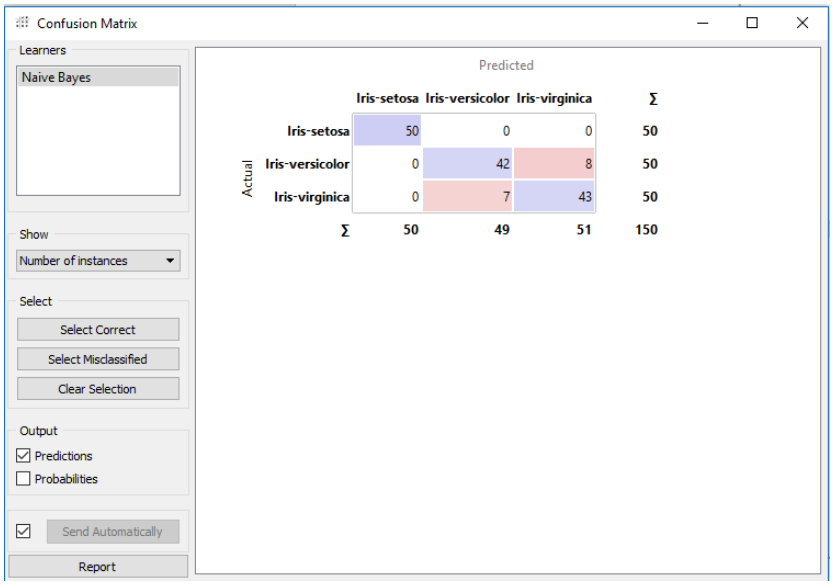
Workflow

1. Build the following workflow.



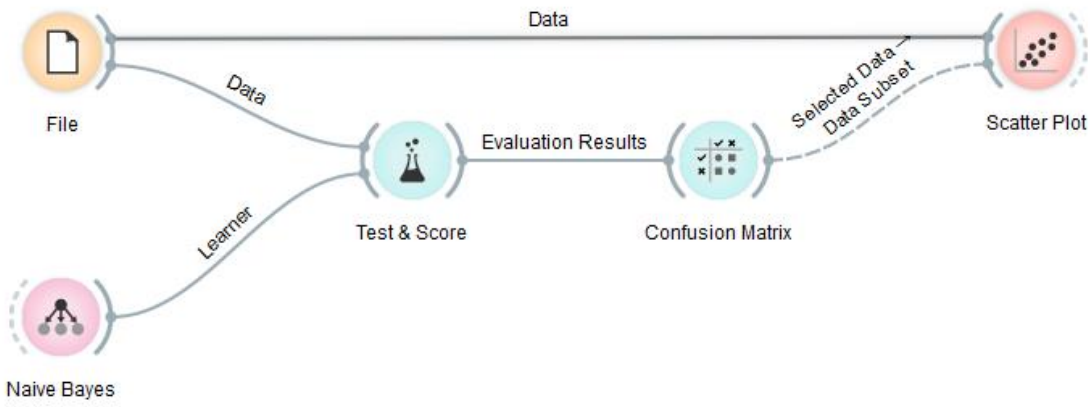
2. Review the performance of the model in **Test & Score** and **Confusion Matrix** widgets.



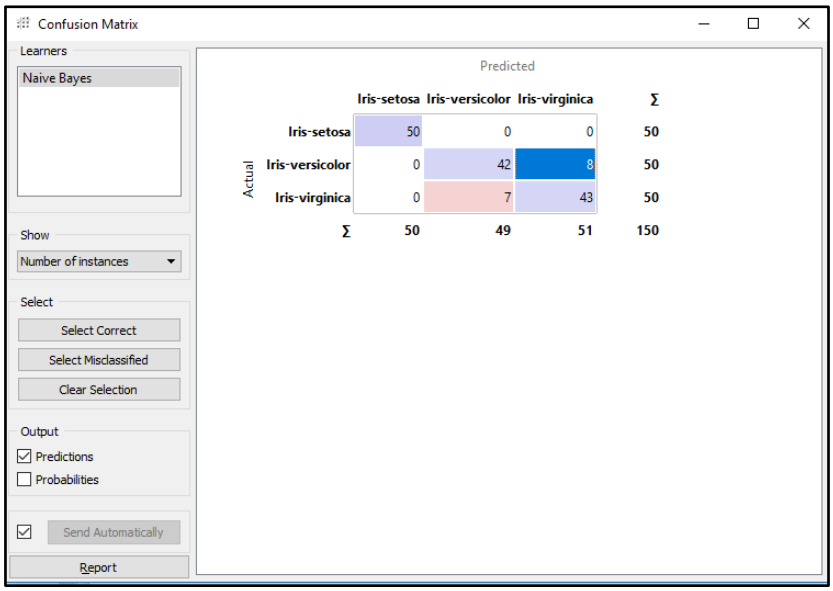


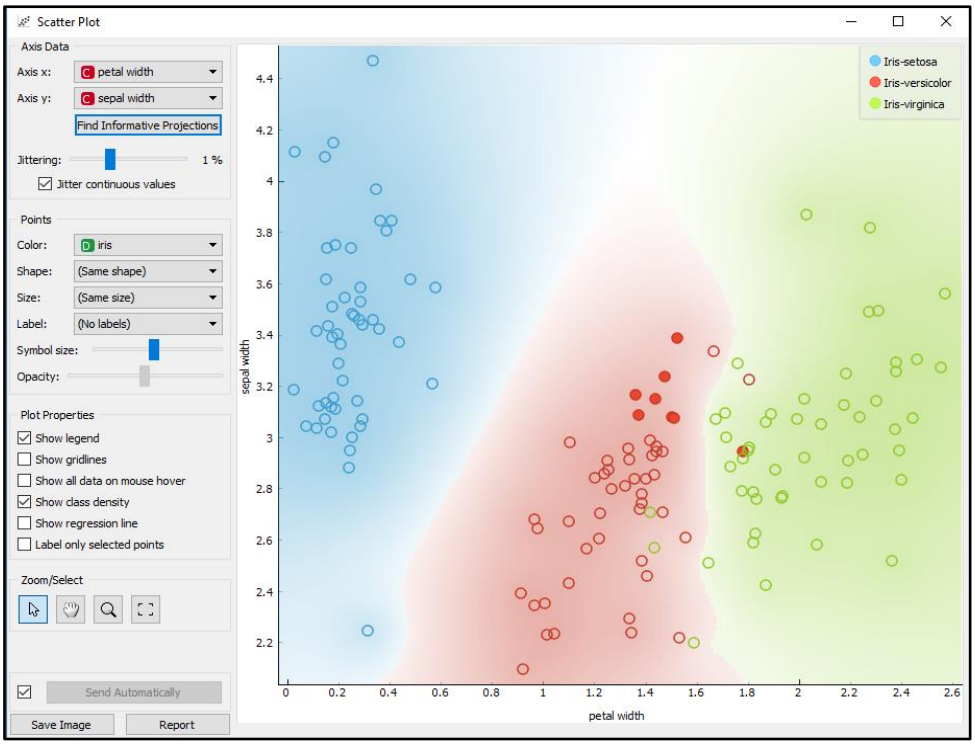
3. Compare with the earlier models you built for the iris dataset. How does Naïve Bayes compare?

4. Let's investigate the model further. Add a Scatter Plot widget. Make sure you have the links setup as shown below. This allows you to interactively view the misclassifications.

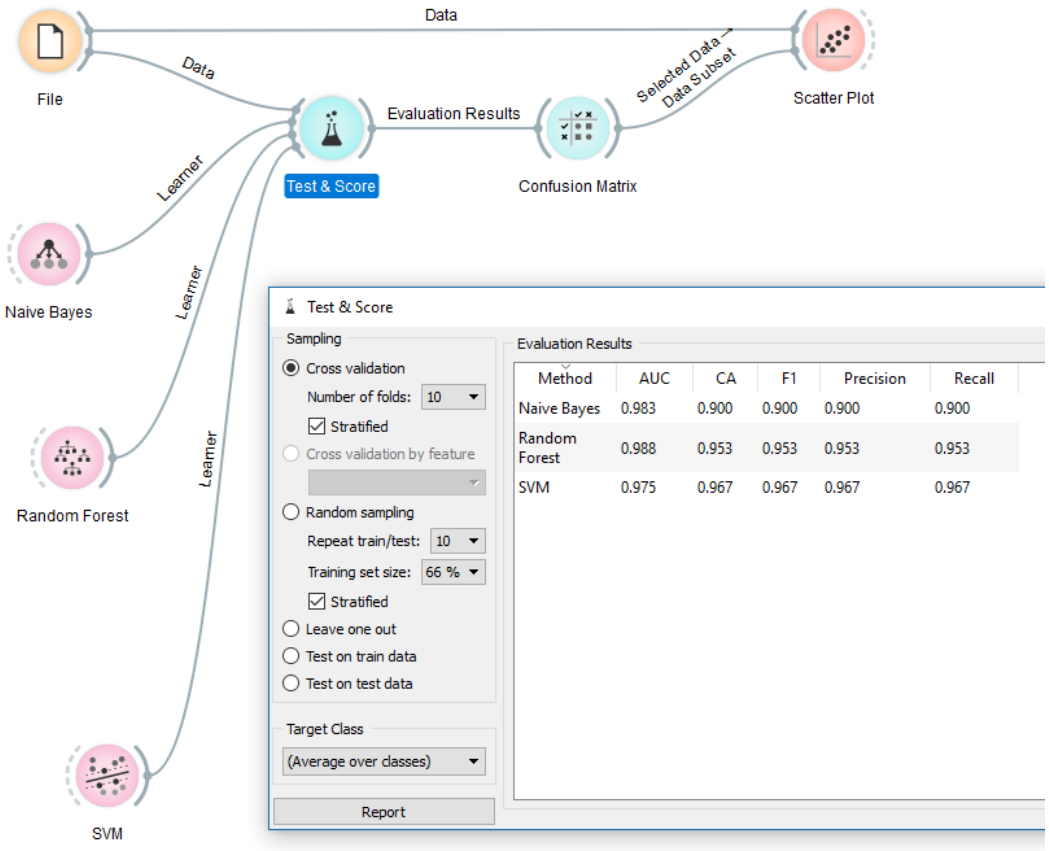


5. Now, click on the misclassifications, you will see the Scatter Plot highlights them. By chaining up the Orange widgets, it allows you to interactively see where the misclassifications are and is a good way to explore and understand your data.





6. You can easily compare the performance of other algorithms or learners (in Orange speak). Here we connect the **Random Forest** and **SVM** widgets to the **Test & Score** widget. Which learner works best?



Conclusion

You have built a Naïve Bayes classifier and compared the performance against other algorithms like Random Forest and SVM. We will cover both later. You have also learnt how to chain widgets together to enable interactive data visualization and exploration.

K-NEAREST NEIGHBOUR (KNN)

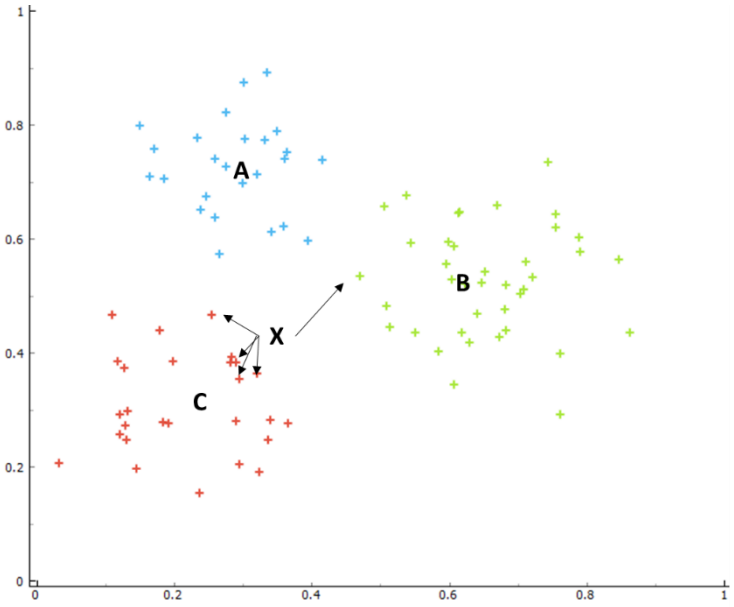
Motivation

kNN is a simple and fast algorithm SUPERVISED learning algorithm and often is used as the base upon which other algorithms are compared.

kNN can be used for classification and regression problems.

Theory

The kNN algorithm searches for k closest training examples in feature space and uses their average as prediction.



A new data point **X**, needs to be classified. In kNN we find the k nearest neighbor to **X**.

In this example, if we set k = 5.

The 5 nearest neighbors are shown. 4 neighbors in set C, and 1 in set B. Set A points are too far.

So we assign **X** to set C.

In k-NN classification, the output is a class membership based on majority of closest

In k-NN regression, the output the average of the values of its k nearest neighbors.

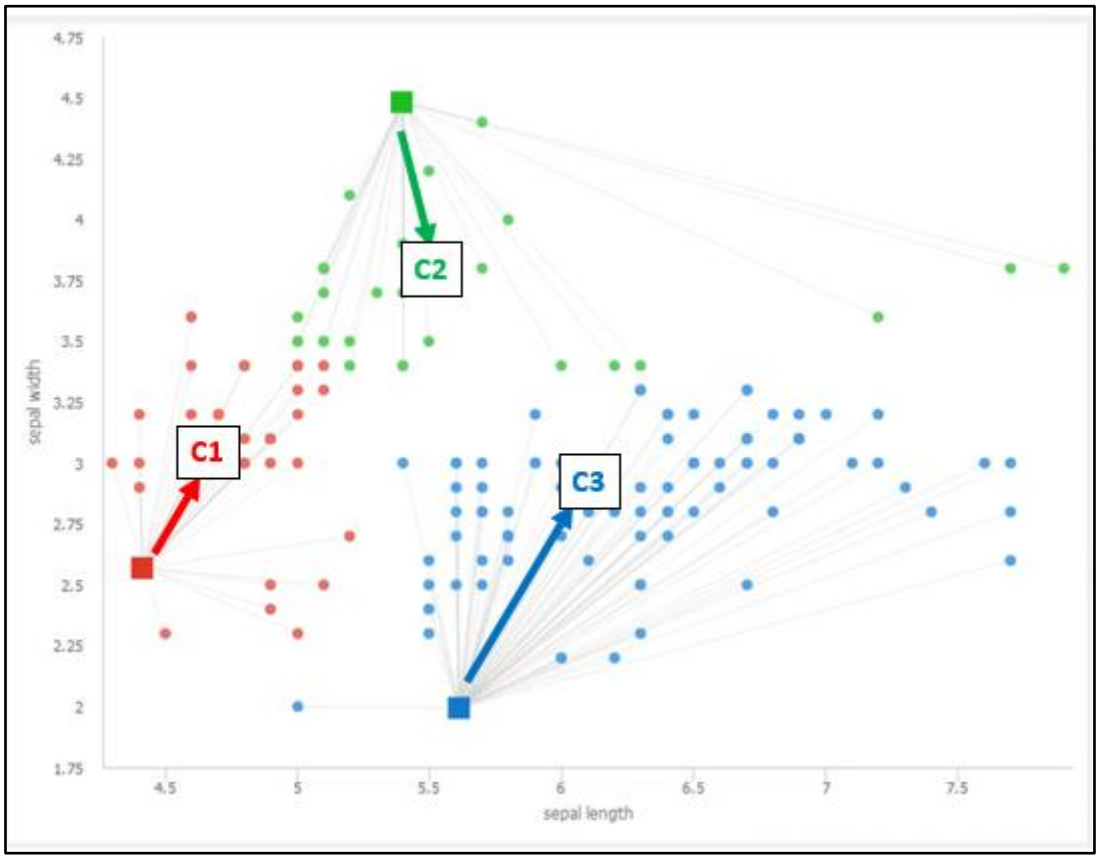
What is difference with k-Means?

k-Means is an unsupervised learning algorithm and is used for CLUSTERING, Note the difference between clustering (an unsupervised learning algorithm) and classification (a supervised learning problem).

k-Means belongs to the family of moving centroid algorithms.

Recall, in k-Means, the k refers to the number of clusters or groups. We normally do not know what the value of k is and we will normally try a range of 2 to 8.

K-Means works by moving the center (or centroid) of the cluster at every iteration to minimize the objective function.

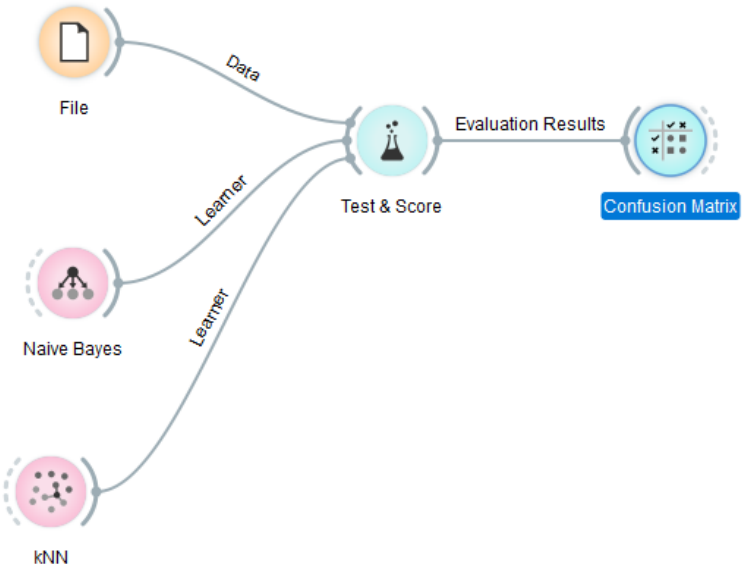


The k in kNN refers to the number of nearest neighbors.

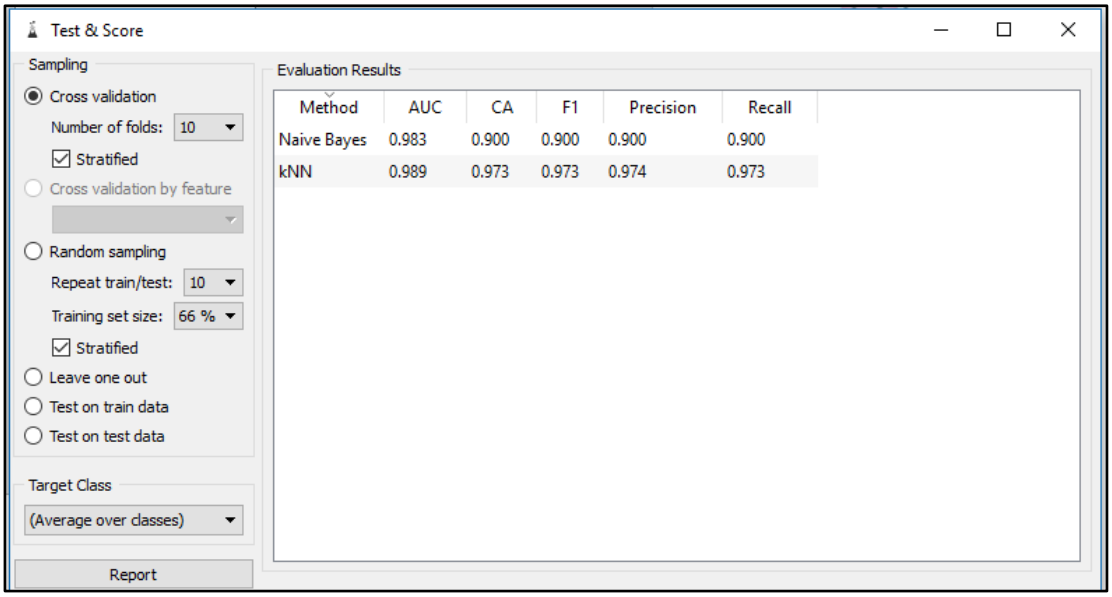
The k in k-Means refers to the number of clusters or groups.

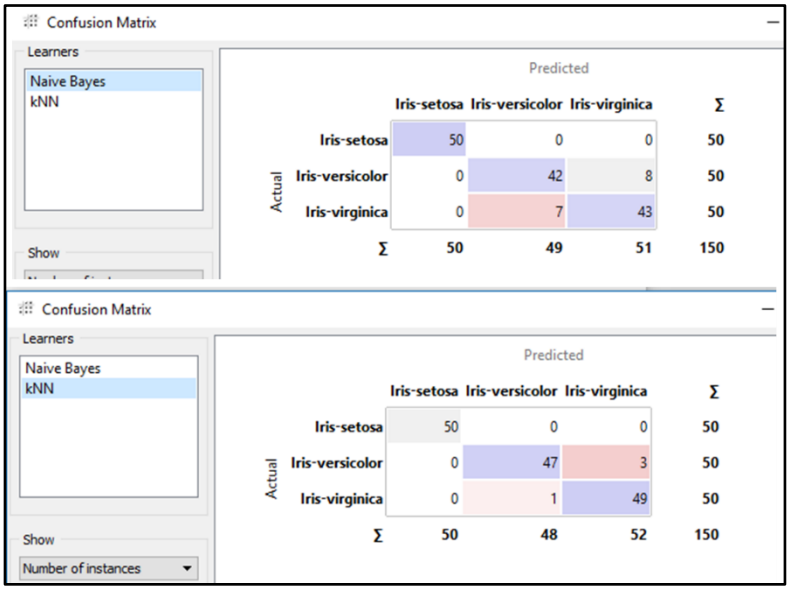
Workflow (Classification)

1. Build the following workflow with the iris dataset.



2. Compare the performance.

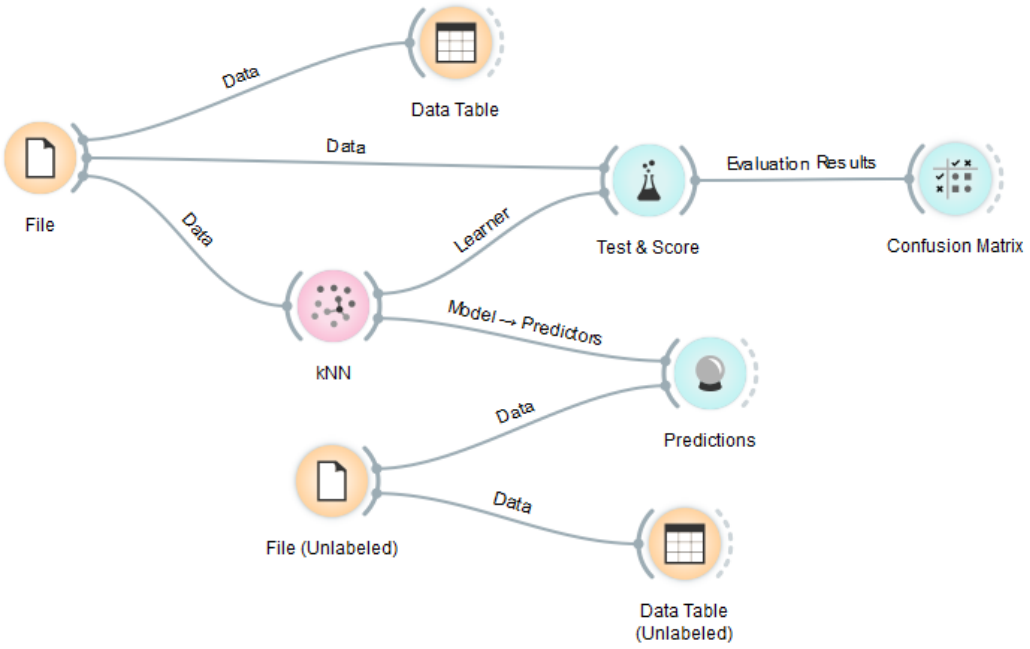




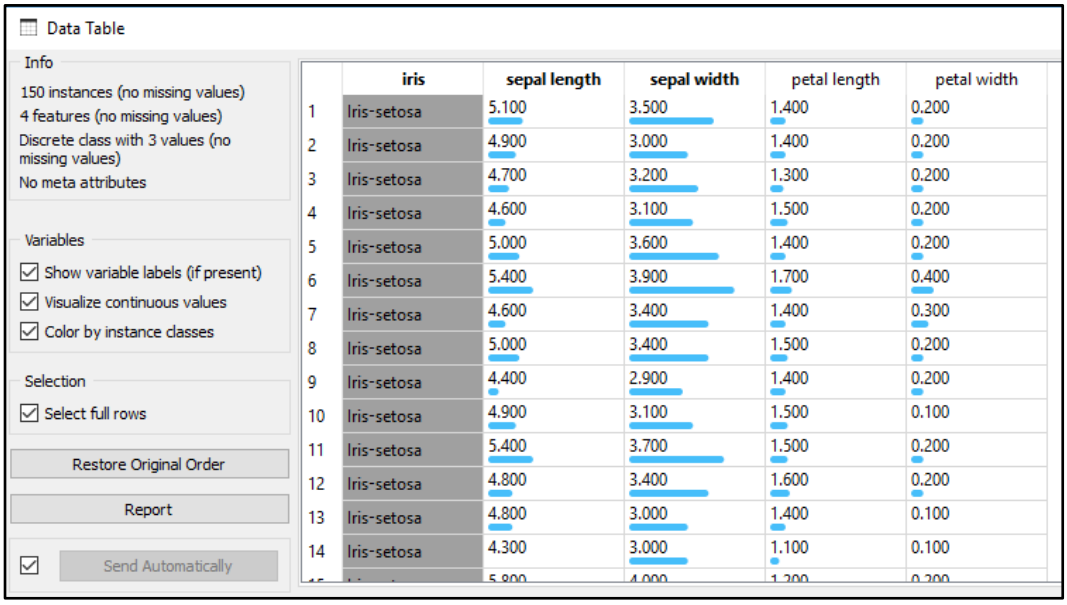
3. Using the trained kNN model to make a classification. Enhance the workflow as shown below.

    a. **File** widget contains the iris.tab training and test dataset with labels.
    b. **File(unlabeled)** widget contains the online iris dataset at **https://goo.gl/qRkcSQ**

Once the kNN model is trained, we can now ask it to make predictions.

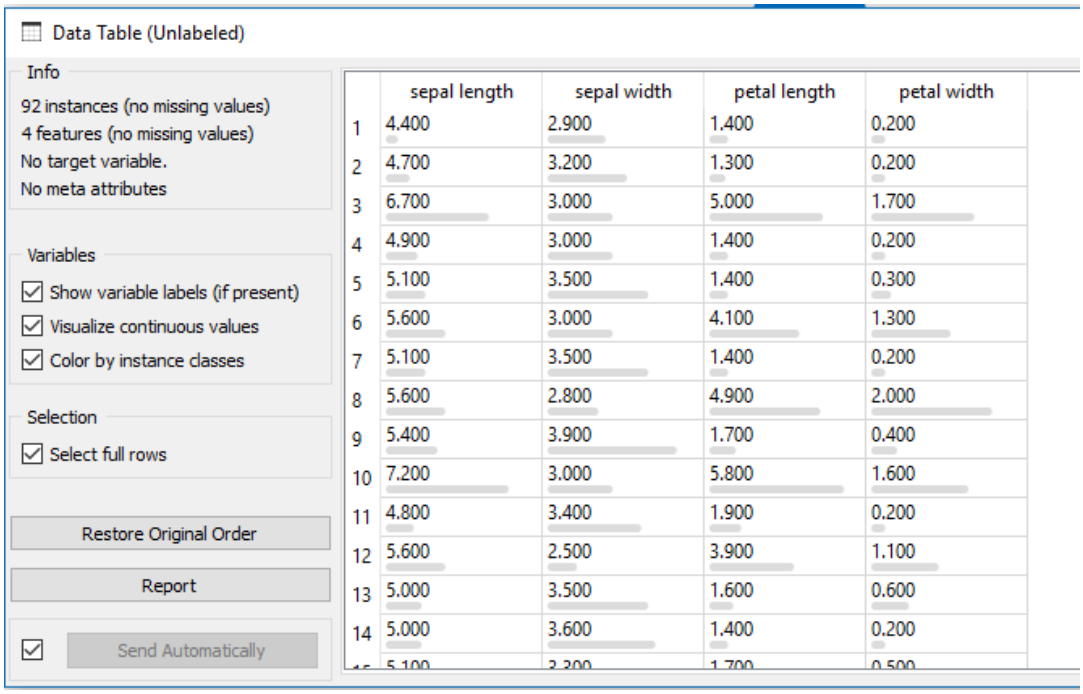


Training and Test dataset (labeled data for supervised training)



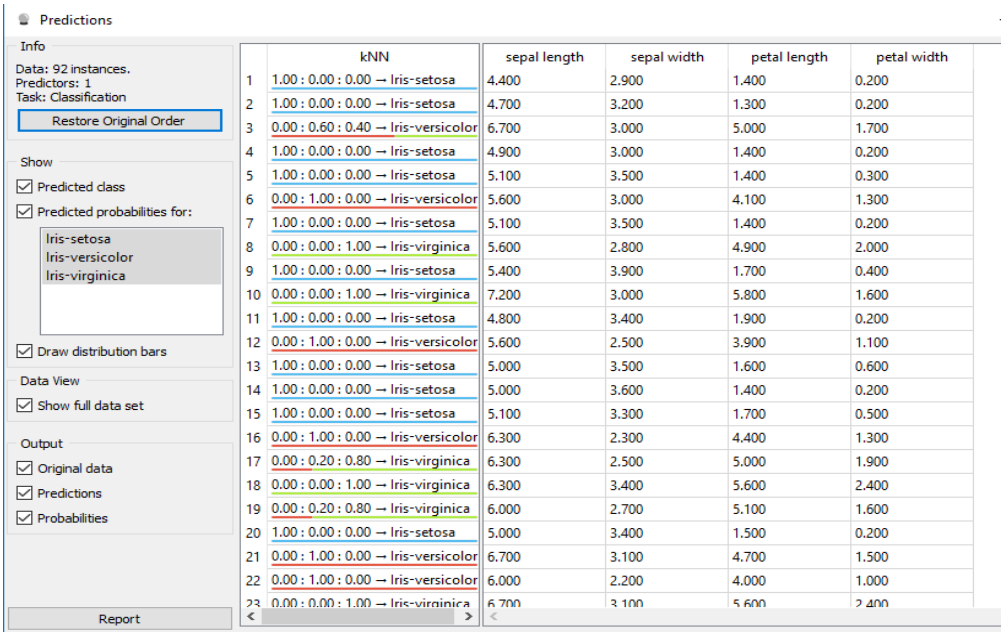
	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100

Data to be classified (no labels in the dataset!)



	sepal length	sepal width	petal length	petal width
1	4.400	2.900	1.400	0.200
2	4.700	3.200	1.300	0.200
3	6.700	3.000	5.000	1.700
4	4.900	3.000	1.400	0.200
5	5.100	3.500	1.400	0.300
6	5.600	3.000	4.100	1.300
7	5.100	3.500	1.400	0.200
8	5.600	2.800	4.900	2.000
9	5.400	3.900	1.700	0.400
10	7.200	3.000	5.800	1.600
11	4.800	3.400	1.900	0.200
12	5.600	2.500	3.900	1.100
13	5.000	3.500	1.600	0.600
14	5.000	3.600	1.400	0.200

4. The **Predictions** widget is used to make the classifications and the results are shown below.

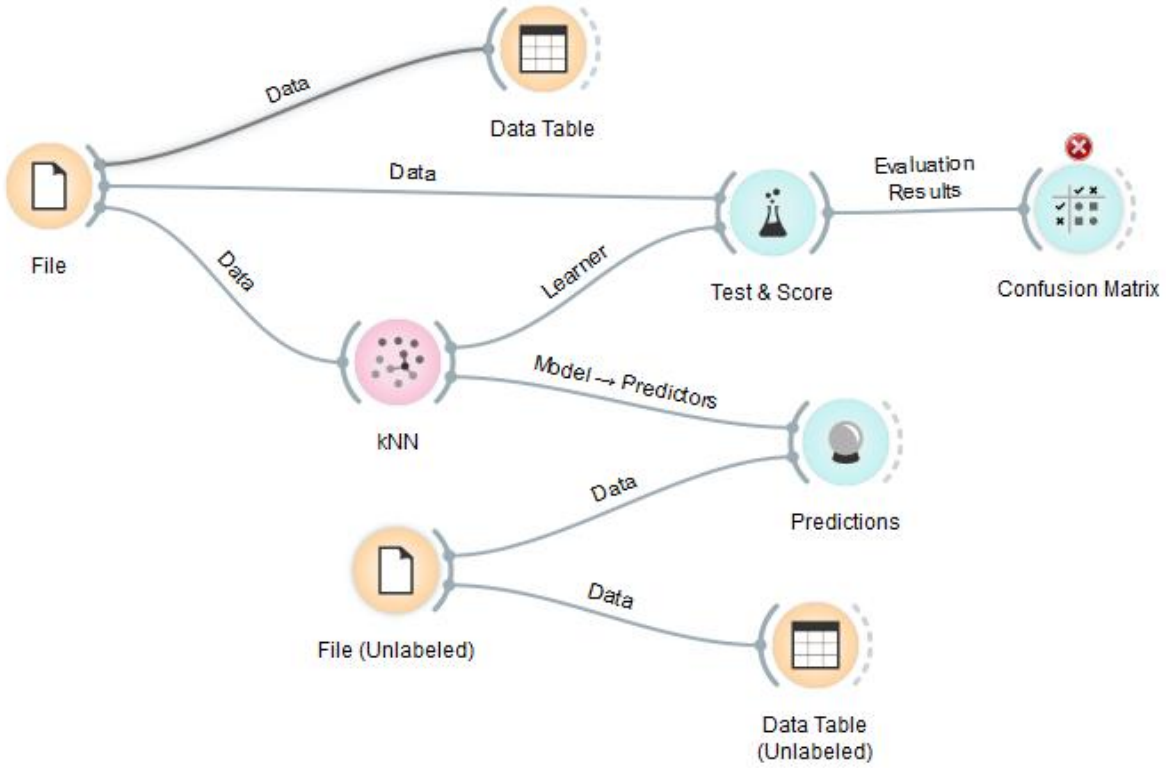


	kNN	sepal length	sepal width	petal length	petal width
1	1.00 : 0.00 : 0.00 → Iris-setosa	4.400	2.900	1.400	0.200
2	1.00 : 0.00 : 0.00 → Iris-setosa	4.700	3.200	1.300	0.200
3	0.00 : 0.60 : 0.40 → Iris-versicolor	6.700	3.000	5.000	1.700
4	1.00 : 0.00 : 0.00 → Iris-setosa	4.900	3.000	1.400	0.200
5	1.00 : 0.00 : 0.00 → Iris-setosa	5.100	3.500	1.400	0.300
6	0.00 : 1.00 : 0.00 → Iris-versicolor	5.600	3.000	4.100	1.300
7	1.00 : 0.00 : 0.00 → Iris-setosa	5.100	3.500	1.400	0.200
8	0.00 : 0.00 : 1.00 → Iris-virginica	5.600	2.800	4.900	2.000
9	1.00 : 0.00 : 0.00 → Iris-setosa	5.400	3.900	1.700	0.400
10	0.00 : 0.00 : 1.00 → Iris-virginica	7.200	3.000	5.800	1.600
11	1.00 : 0.00 : 0.00 → Iris-setosa	4.800	3.400	1.900	0.200
12	0.00 : 1.00 : 0.00 → Iris-versicolor	5.600	2.500	3.900	1.100
13	1.00 : 0.00 : 0.00 → Iris-setosa	5.000	3.500	1.600	0.600
14	1.00 : 0.00 : 0.00 → Iris-setosa	5.000	3.600	1.400	0.200
15	1.00 : 0.00 : 0.00 → Iris-setosa	5.100	3.300	1.700	0.500
16	0.00 : 1.00 : 0.00 → Iris-versicolor	6.300	2.300	4.400	1.300
17	0.00 : 0.20 : 0.80 → Iris-virginica	6.300	2.500	5.000	1.900
18	0.00 : 0.00 : 1.00 → Iris-virginica	6.300	3.400	5.600	2.400
19	0.00 : 0.20 : 0.80 → Iris-virginica	6.000	2.700	5.100	1.600
20	1.00 : 0.00 : 0.00 → Iris-setosa	5.000	3.400	1.500	0.200
21	0.00 : 1.00 : 0.00 → Iris-versicolor	6.700	3.100	4.700	1.500
22	0.00 : 1.00 : 0.00 → Iris-versicolor	6.000	2.200	4.000	1.000
23	0.00 : 0.00 : 1.00 → Iris-virginica	6.700	3.100	5.600	2.400

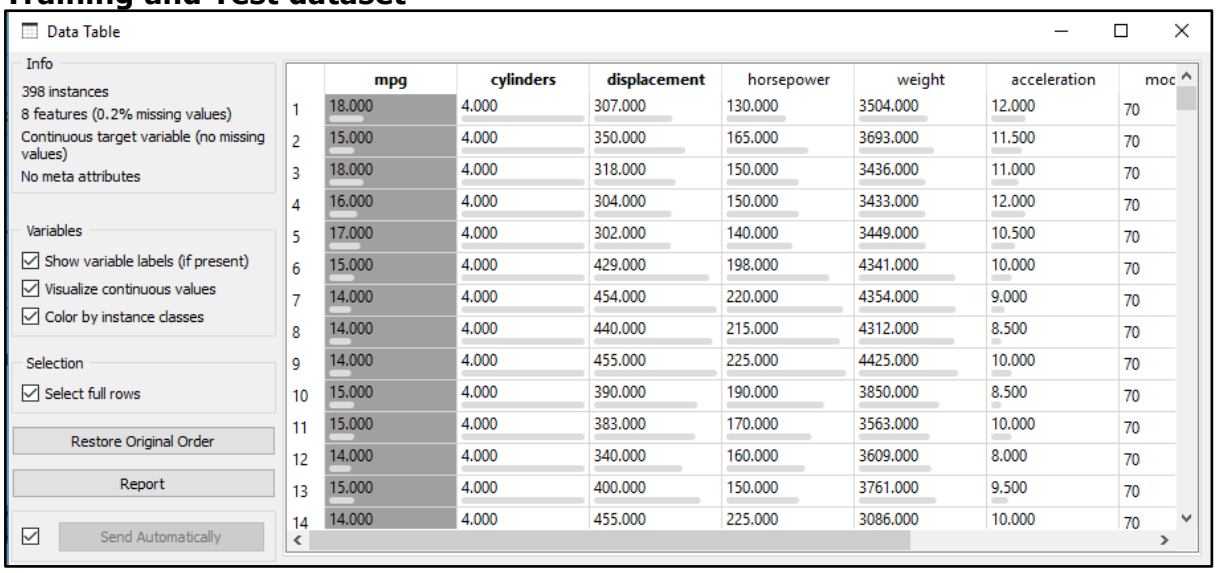
Workflow (Regression)

1. In this section, we are going to use kNN to build a regression model (predict continuous value). Use the same workflow. Load the auto-mpg dataset into **File** widget.

   See that the **Confusion Matrix** has a red x on it. There is an error. Orange is smart enough to flag out incompatibilities. In this case, the Confusion Matrix cannot handle and does not make sense in a regression workflow, hence the red x. You should remove it.

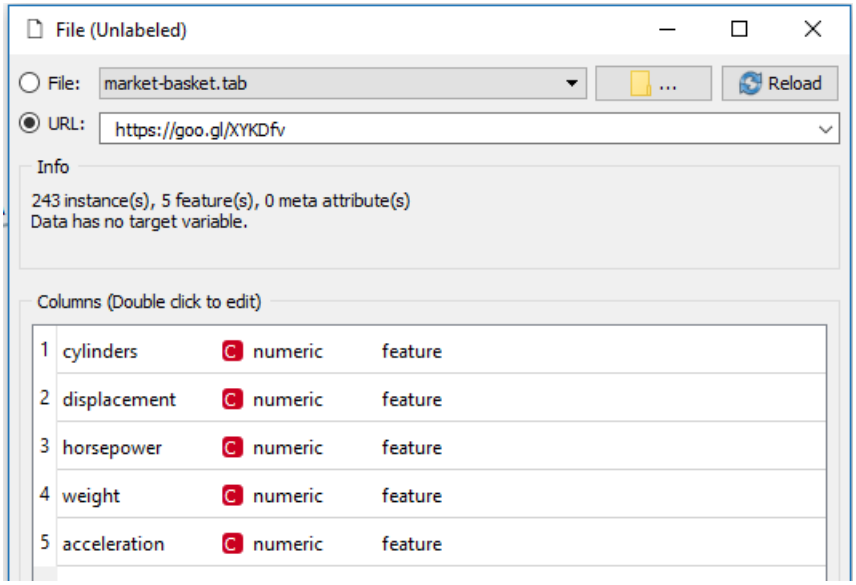


Training and Test dataset

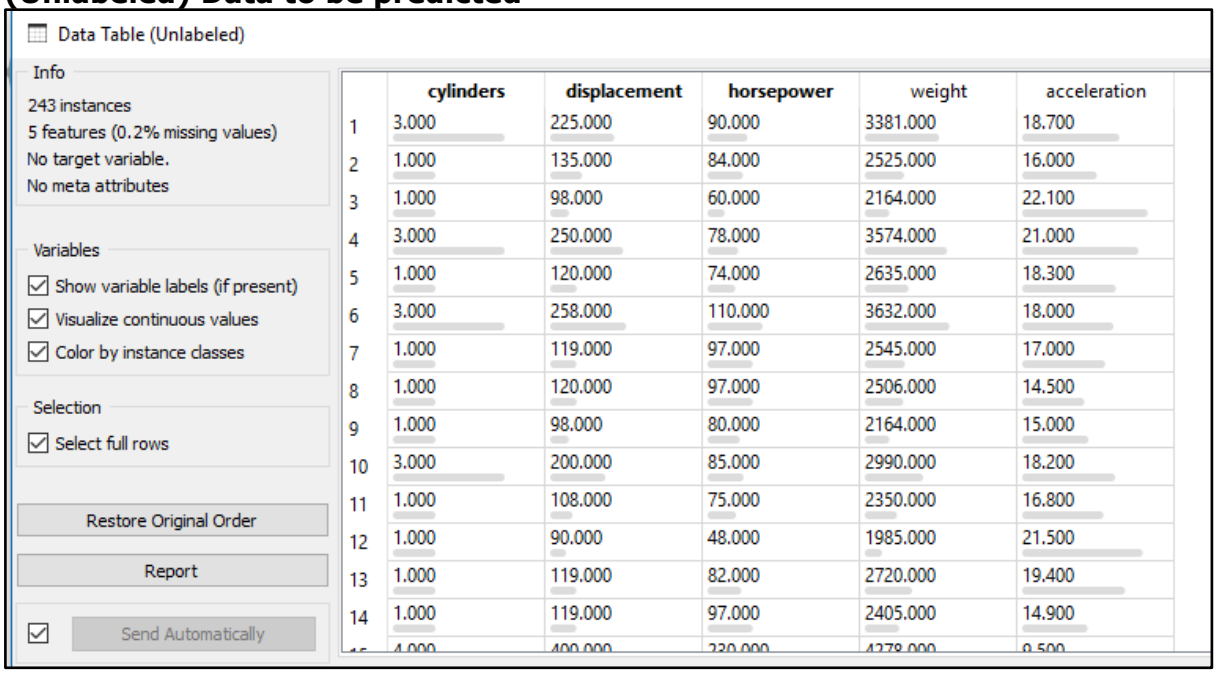


	mpg	cylinders	displacement	horsepower	weight	acceleration	moc
1	18.000	4.000	307.000	130.000	3504.000	12.000	70
2	15.000	4.000	350.000	165.000	3693.000	11.500	70
3	18.000	4.000	318.000	150.000	3436.000	11.000	70
4	16.000	4.000	304.000	150.000	3433.000	12.000	70
5	17.000	4.000	302.000	140.000	3449.000	10.500	70
6	15.000	4.000	429.000	198.000	4341.000	10.000	70
7	14.000	4.000	454.000	220.000	4354.000	9.000	70
8	14.000	4.000	440.000	215.000	4312.000	8.500	70
9	14.000	4.000	455.000	225.000	4425.000	10.000	70
10	15.000	4.000	390.000	190.000	3850.000	8.500	70
11	15.000	4.000	383.000	170.000	3563.000	10.000	70
12	14.000	4.000	340.000	160.000	3609.000	8.000	70
13	15.000	4.000	400.000	150.000	3761.000	9.500	70
14	14.000	4.000	455.000	225.000	3086.000	10.000	70

2. Load the new set of unlabeled auto-mpg data from **https://goo.gl/XYKDfv**

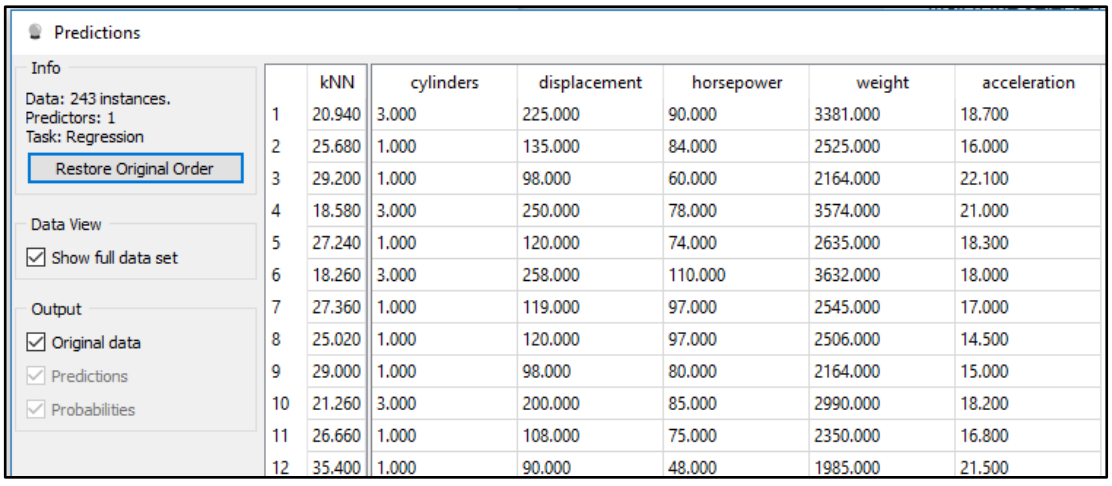


(Unlabeled) Data to be predicted



	cylinders	displacement	horsepower	weight	acceleration
1	3.000	225.000	90.000	3381.000	18.700
2	1.000	135.000	84.000	2525.000	16.000
3	1.000	98.000	60.000	2164.000	22.100
4	3.000	250.000	78.000	3574.000	21.000
5	1.000	120.000	74.000	2635.000	18.300
6	3.000	258.000	110.000	3632.000	18.000
7	1.000	119.000	97.000	2545.000	17.000
8	1.000	120.000	97.000	2506.000	14.500
9	1.000	98.000	80.000	2164.000	15.000
10	3.000	200.000	85.000	2990.000	18.200
11	1.000	108.000	75.000	2350.000	16.800
12	1.000	90.000	48.000	1985.000	21.500
13	1.000	119.000	82.000	2720.000	19.400
14	1.000	119.000	97.000	2405.000	14.900

3. In the workflow, the trained kNN model sends its model-predictors to **Predictions** widget. The **File** (Unlabeled) feeds new data into the **Predictions** widget to make a prediction. See that the **Predictions** widget attach a new column to the dataset. The kNN column is the predicted MPG.



	kNN	cylinders	displacement	horsepower	weight	acceleration
1	20.940	3.000	225.000	90.000	3381.000	18.700
2	25.680	1.000	135.000	84.000	2525.000	16.000
3	29.200	1.000	98.000	60.000	2164.000	22.100
4	18.580	3.000	250.000	78.000	3574.000	21.000
5	27.240	1.000	120.000	74.000	2635.000	18.300
6	18.260	3.000	258.000	110.000	3632.000	18.000
7	27.360	1.000	119.000	97.000	2545.000	17.000
8	25.020	1.000	120.000	97.000	2506.000	14.500
9	29.000	1.000	98.000	80.000	2164.000	15.000
10	21.260	3.000	200.000	85.000	2990.000	18.200
11	26.660	1.000	108.000	75.000	2350.000	16.800
12	35.400	1.000	90.000	48.000	1985.000	21.500

Conclusion

We have built a kNN classifier to classify the iris flowers, and a kNN regression model to predict the MPG of a car.

You have also learnt how to use the Predictions widget to do an actual prediction, and learnt about Orange ability to flag out incompatible widgets.

The difference between k-Means and kNN is also discussed.

LINEAR REGRESSION

Motivation

Linear regression is the workhorse of the supervised learning. It is conceptually easy to understand and works well for most problems.

Popular use of Linear Regression includes:

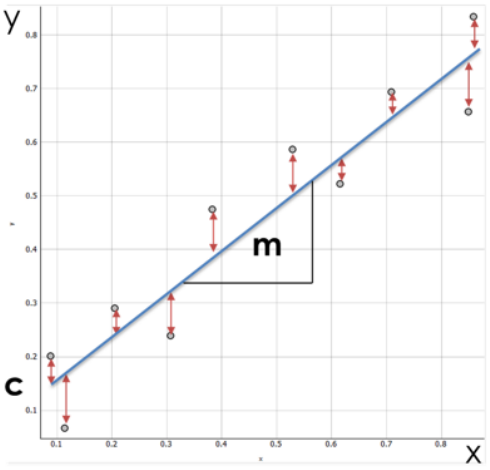
1. Predicting the arrival times of planes into an airport
2. Predicting the price of cars, houses or resale flats

And no, Linear Regression (and all other algorithms that I know) cannot predict the 4D or TOTO numbers.

Theory

Most of us would have learnt linear regression in secondary school (at least during my time).

In linear regression, the data points observed are assumed to be the result of random deviations (red-arrows) from an underlying relationship (the blue line) between the dependent variable (y) and independent variable (x).



$$y = mx + c$$

We have a number of observed **x** values. We want to write a linear equation f(x) or **y**.

We need to find the value of **m** and the y-intercept **c**.

Visually we want to place a straight line through all the values of **x** as shown on the right such that the sum of all the red-arrows is the smallest.

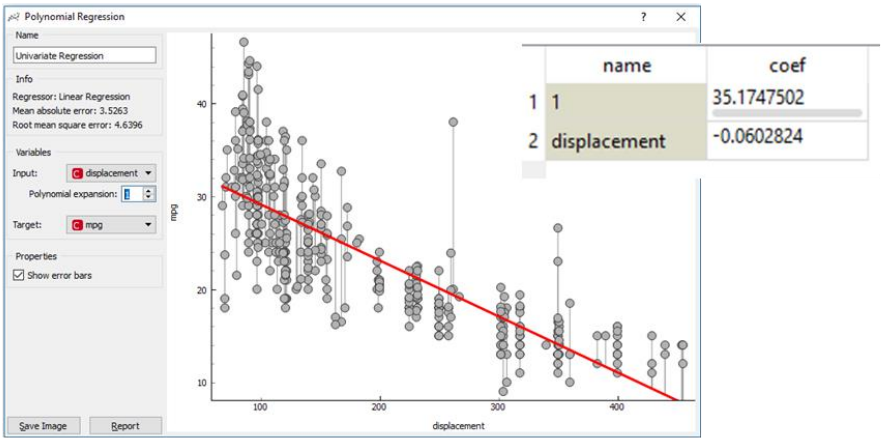
The most common way to fit the line mathematically and in a computer, is a method called **Ordinary Least Squares (OLS)**. OLS basically minimizes the sum of squared residuals (red-arrows in the diagram).

Recall, linear models can have curve lines (linear here does not mean straight line)!

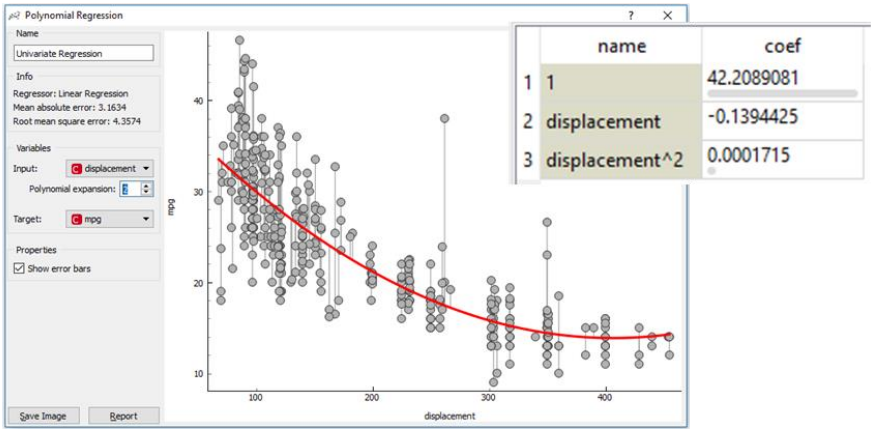
The "linear" term refers to the linear combination of the parameters or regression coefficients (**m**) and the predictor variables (**x**)

The predictor variables (**x**) themselves can be arbitrarily transformed, and in fact multiple copies of the same underlying predictor variable can be added, each one transformed differently.

$$mpg = 35.2 - 0.06.displacement$$

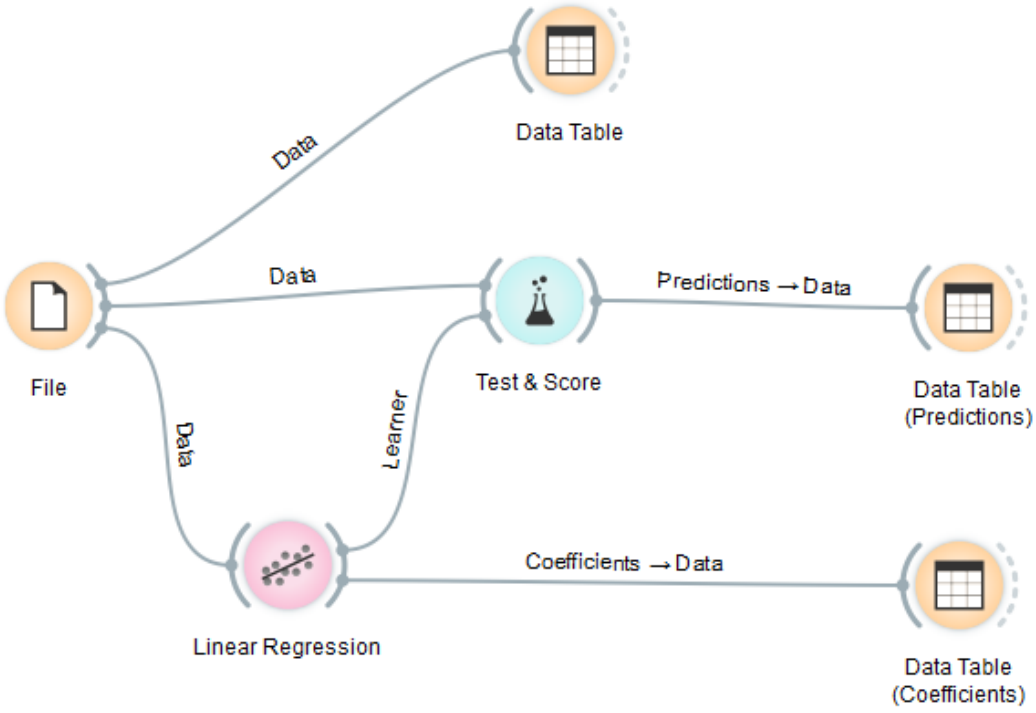


$$mpg = 42.2 - 0.14.displacement + 0.00017.displacement^2$$

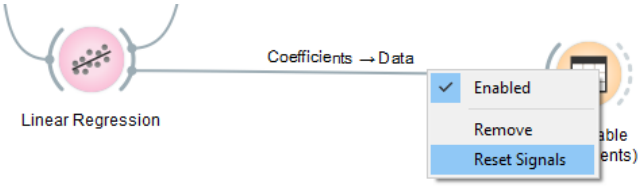


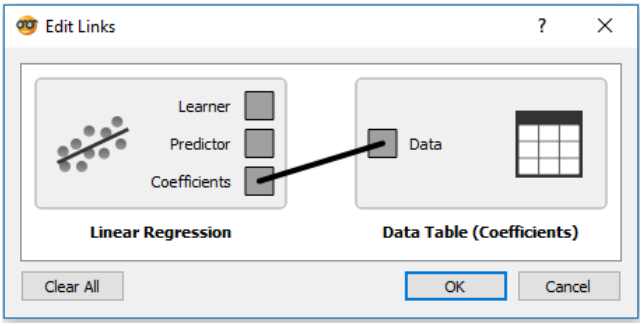
Workflow

1. Build the following workflow. Use the resale-sample.csv dataset.



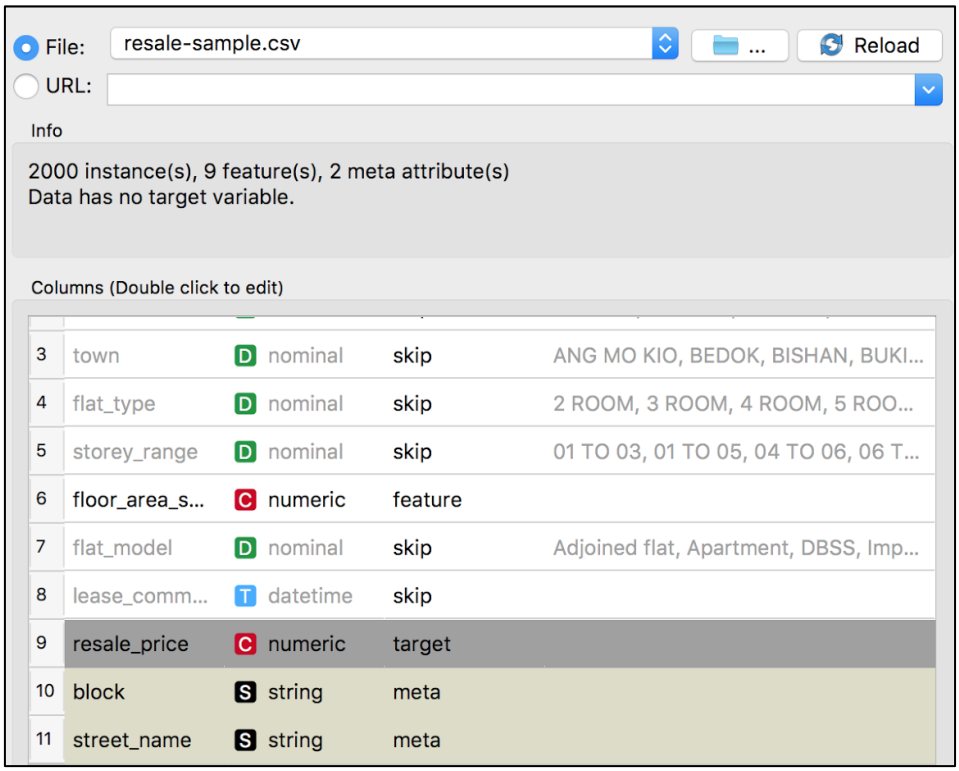
2. Right click on the link shown below and select **Reset Signals**. Make sure you are propagating the **Coefficients** to the **Data Table (Coefficient)** downstream.





3. It's a good idea to start with only ONE predictor variable, in this case the floor-area-sqm variable, then add other variables in one by one to see the relative effect of each on the target variable. To do this, we set all other features in our data to 'skip'

**** also note that linear regression takes only continuous variables as inputs, so discrete variables have to be excluded from this model, or transformed into continuous form.**



4. Later, an option would be to automate the variable selection process by applying something like a grid-search algorithm which can try different variable

combinations for us. So we do not have to hand-tune the variables, but it's a good idea initially to play with variables so we build an intuition of the data.

5. View the various Data Tables widget output.

	resale_price	block	street_name	floor_area_sqm
1	400000.000	119	TECK WHYE...	104.000
2	404000.000	22	HAVELOCK ...	64.000
3	422000.000	906	JURONG WE...	141.000
4	375000.000	510	JURONG WE...	74.000
5	385000.000	232	JURONG EA...	95.000
6	655000.000	284	TOH GUAN ...	120.000
7	590000.000	326	ANG MO KIO...	92.000
8	375000.000	668	CHANDER RD	75.000
9	490000.000	428	ANG MO KIO...	92.000
10	543000.000	2	HAIG RD	92.000
11	435000.000	480	SEGAR RD	94.000
12	465000.000	33	MARINE CRES	65.000
13	448000.000	695	HOUGANG S...	104.000
14	460000.000	412B	FERNVALE L...	114.000
15	405000.000	103	BEDOK RES...	93.000
16	638888.000	361	WOODLAND...	145.000
17	353000.000	338	BT BATOK S...	84.000
18	454000.000	319A	ANCHORVA...	90.000
19	638000.000	5	DELTA AVE	92.000
20	430000.000	688C	CHOA CHU ...	90.000
21	280000.000	617	HOUGANG ...	64.000
22	316988.000	274	YISHUN ST 22	74.000

	resale_price	block	street_name	Linear Regression	Fold
1	400000.000	119	TECK WHYE...	477799.826	1
2	404000.000	22	HAVELOCK ...	333318.230	1
3	422000.000	906	JURONG WE...	611445.301	1
4	375000.000	510	JURONG WE...	369438.629	1
5	385000.000	232	JURONG EA...	445291.467	1
6	655000.000	284	TOH GUAN ...	535592.464	1
7	590000.000	326	ANG MO KIO...	434455.347	1
8	375000.000	668	CHANDER RD	373050.669	1
9	490000.000	428	ANG MO KIO...	434455.347	1
10	543000.000	2	HAIG RD	434455.347	1
11	435000.000	480	SEGAR RD	441679.427	1
12	465000.000	33	MARINE CRES	336930.270	1
13	448000.000	695	HOUGANG S...	477799.826	1
14	460000.000	412B	FERNVALE L...	513920.224	1
15	405000.000	103	BEDOK RES...	438067.387	1
16	638888.000	361	WOODLAND...	625893.461	1
17	353000.000	338	BT BATOK S...	405559.028	1
18	454000.000	319A	ANCHORVA...	427231.267	1
19	638000.000	5	DELTA AVE	434455.347	1
20	430000.000	688C	CHOA CHU ...	427231.267	1
21	280000.000	617	HOUGANG ...	333318.230	1
22	316988.000	274	YISHUN ST 22	369438.629	1
23	319000.000	616	BEDOK RES...	347766.390	1

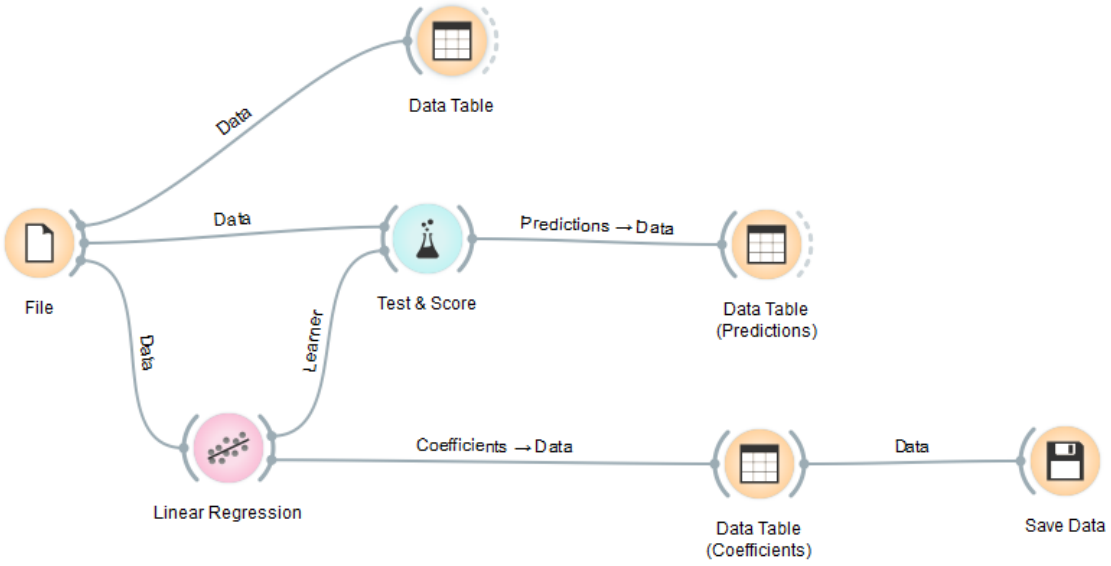
	name	coef
1	intercept	102147.678...
2	floor_area_s...	3612.03987...

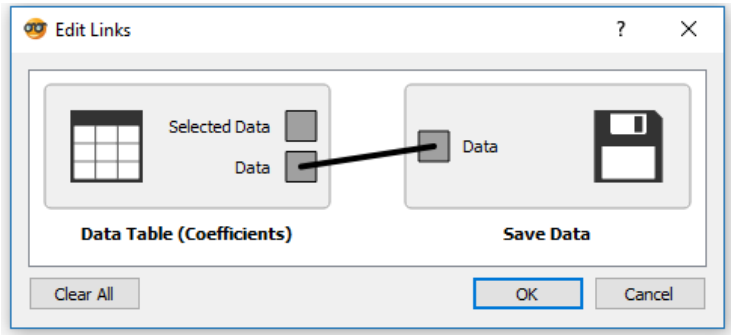
6. From the Data Table (Coefficients), you can see the values of **m**. Note that the line suggests a positive relationship between floor-area-sqm and resale_price, which agrees with our earlier visualization.

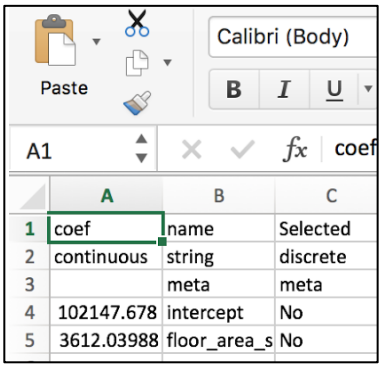
resale_price = 102147.678 + 3612.04floor_area_sqm

7. With the equations above, you can code it up in ANY language or pass the equations to your IT/developer to put the model in production. Of course, you will need to have a system to update the equations when required, for example, when new type of cars (electric or self-driving) are introduced into the market.

8. You can extract out the coefficients by saving the coefficients to a file as shown.





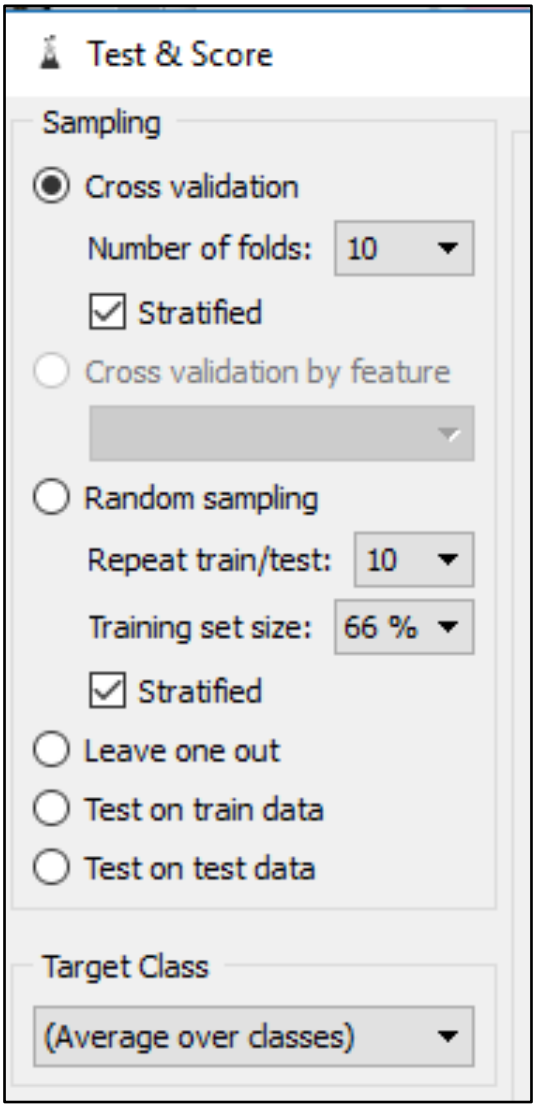


	A	B	C
1	coef	name	Selected
2	continuous	string	discrete
3		meta	meta
4	102147.678	intercept	No
5	3612.03988	floor_area_s	No

Sampling methods

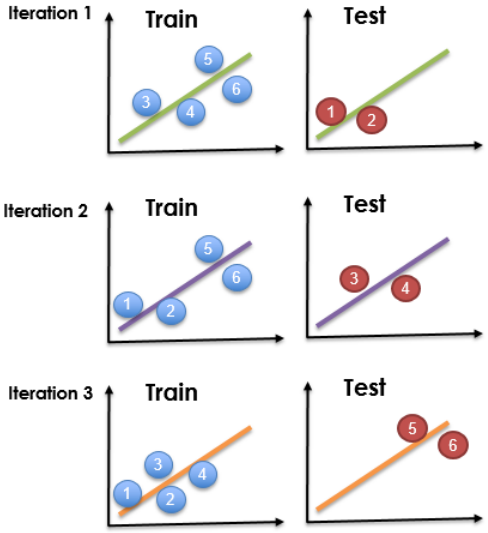
1. When a Regression model is built, training datasets are used to train the model, and a separate test dataset is used to test how good the model is. This is iterative and repeats a fixed number of time, or until a specified criterion is reached. The traditional method was to split the dataset into training/test datasets typically 60/40, 70/30, 80/20.

2. The **Test & Score** widget provide the most popular sampling techniques built-in and makes building models faster, so there is no need to "manually" split the dataset except for intermediate and advanced models or when you wish to preprocess and merge datasets from multiple sources.



3. The gold standard today to train and test models is the cross-validation sampling methods (also known as k-fold sampling where k is typically 5 or 10). It resolves the issue highlighted in #2 above, and brings other benefits. It does come at a cost of higher computational costs.

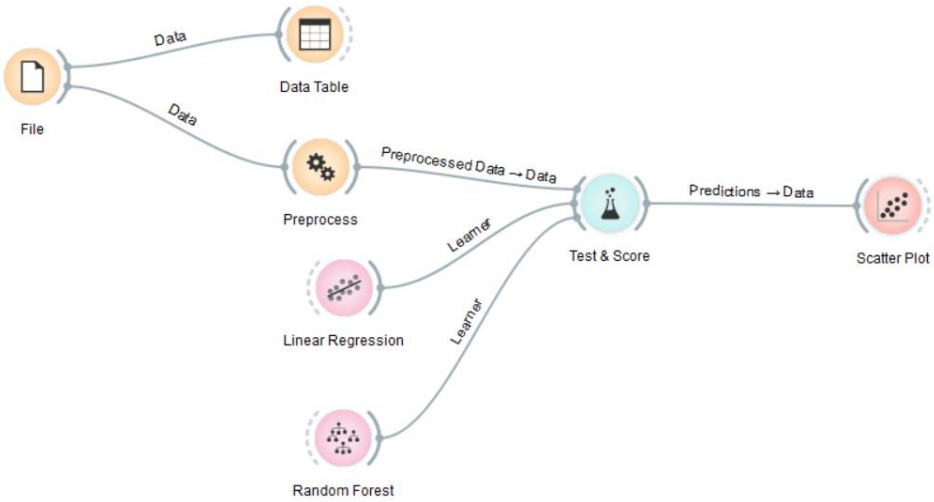
Then take the average of all 3 iterations.

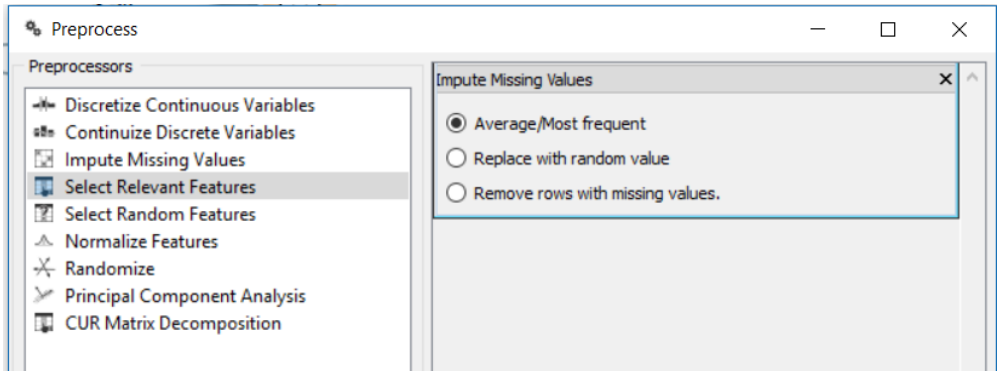


4. We recommend to just leave it the **Test & Score** widget at the default of **Cross Validation** and at **10-folds** and with **Stratified** checked. (Stratified ensures a balanced selection of all class categories.)

Preprocessing for better results

1. We mentioned earlier that the dataset had both numeric and categorical data. Would using all the data make sense? Would we get better results is we do some pre-processing?
2. What about missing values? Please go ahead to try to see if you can get better scores by removing with the **Select Columns** widget or combining features with **Feature Constructor** widget.
3. Add the **Preprocess** widget as show. The **Preprocess** widget provides several commonly used preprocessing methods. For missing values, you can select **Impute Missing Values** and use the **Average/Most Frequent** option to compute any missing values.





Conclusion

You have built a linear regression model in this section and learnt how to extract out the equations for production. It was also explained that a linear regression model is **NOT** necessarily a straight line.

LOGISTIC REGRESSION

Motivation

Logistic Regression is the workhorse for classification problems. Logistic regression is sometimes all you need for your classification problems. It is easy to use and the output naturally provides a probability of the classification.

Typical use:

1. Has fraud been committed?
2. Is he/she a target customer?
3. Is the patient likely to have diabetes?
4. Will I pass an exam if I study for 4 hours a day (and what is the probability)?

Theory

Logistic regression is a misnomer. It does not do REGRESSION! It is a CLASSIFICATION algorithm, and provides a binary output of YES | NO with a probability (0.0 – 1.0).

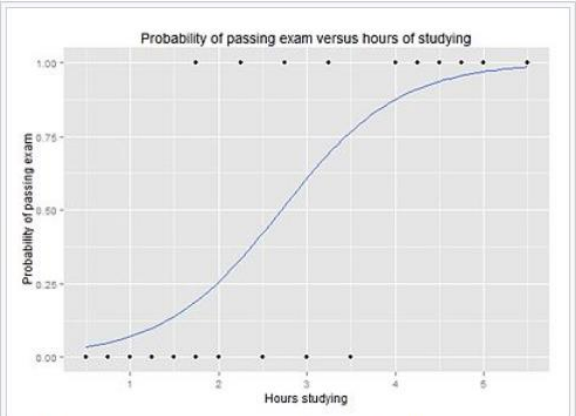
The logistic regression graph is the well-known S-shape. The formula is

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

which can be re-written as

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$$

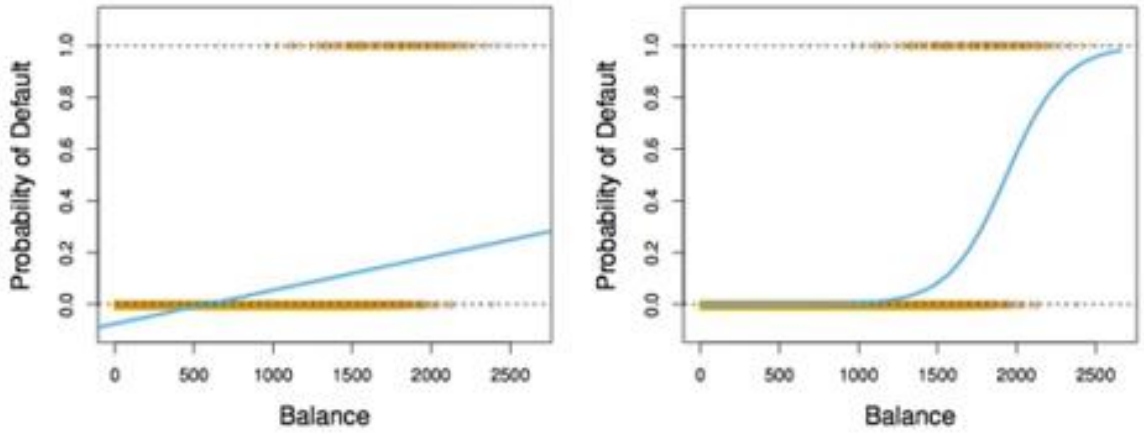
And becomes a linear model which gives a probability between 0 and 1.



Graph of a logistic regression curve showing probability of passing an exam versus hours studying

If it is a two-class problem, you may ask, why can't we use a linear regression model?

Logistic Regression vs Linear Regression

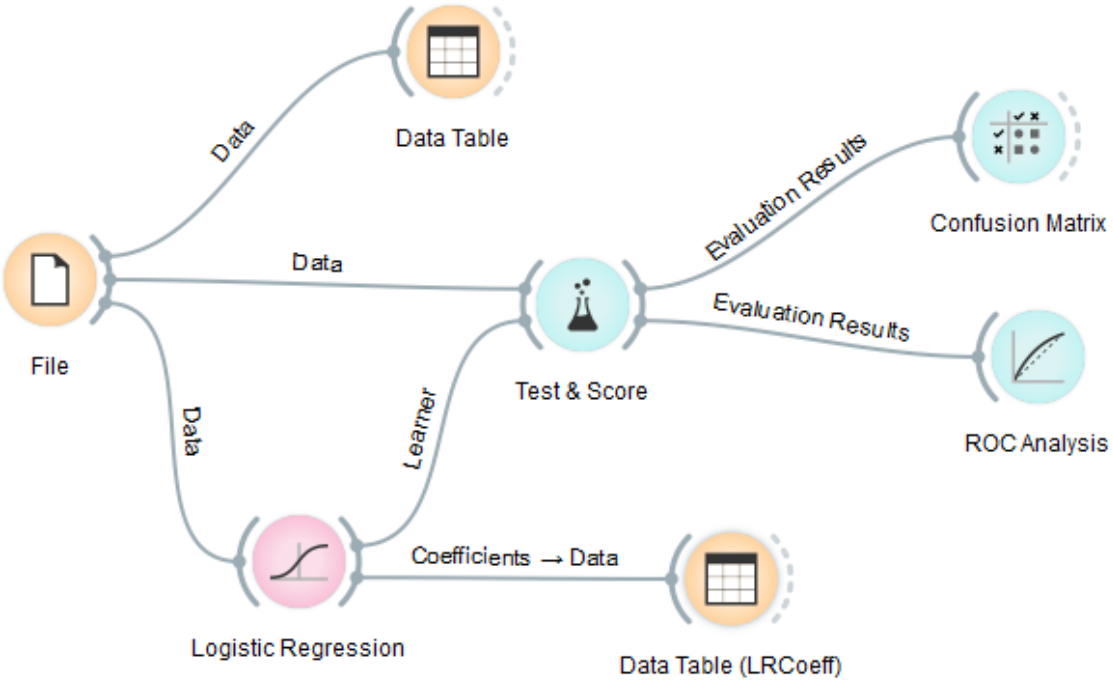


See that a straight line linear model may give probabilities less than 0 or greater than 1 (which does not make sense), hence, the logistic regression is more suitable.

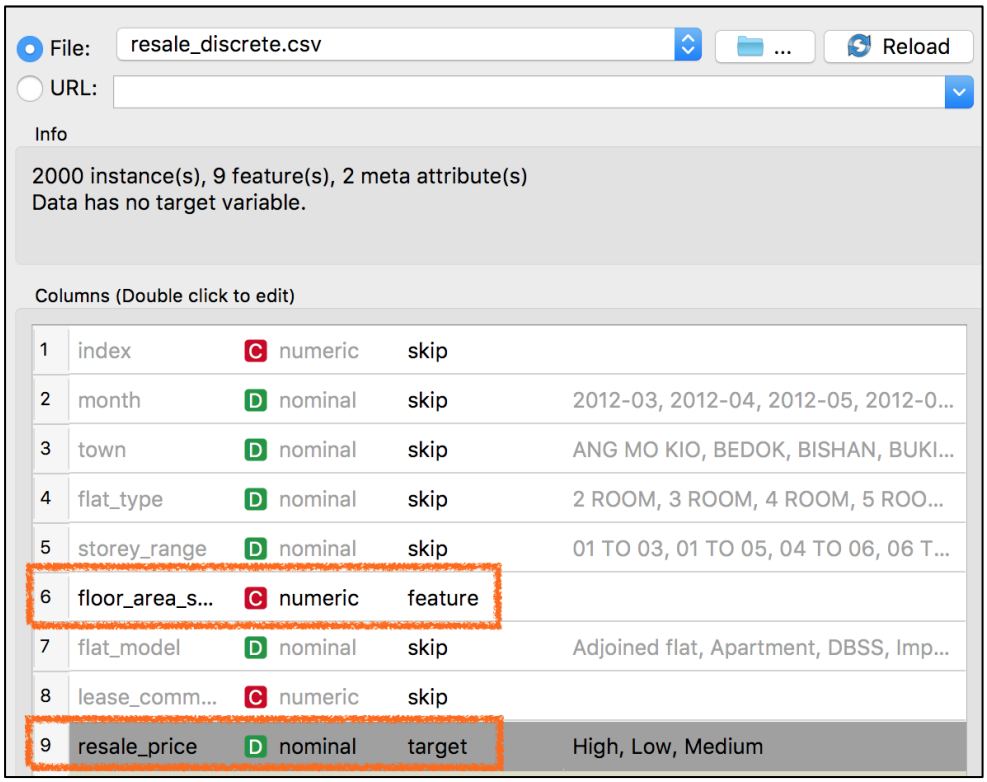
Workflow (resale-discrete)

1. Remember that with Linear Regression, we predicted resale_price as a **continuous variable**. In fact, we can reframe the question to make it amenable to Logistic Regression. For this, we can discretize resale_price into Low, Medium and High. Low, Medium and High then become **categorical variables** to predict.

2. Build the following workflow. Use the data from the **resale_discrete.csv**



3. Again, narrow down the dataset to focus on understanding one variable deeply.



4. Review the dataset. It is a table consisting of the floor-area-sqm and whether flat prices are Low, Medium or High

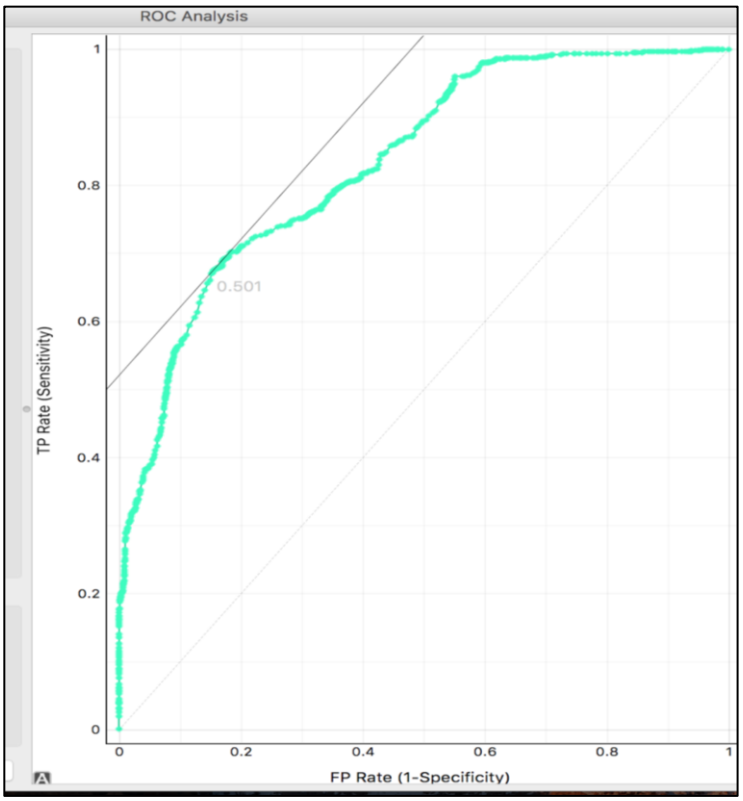
	resale_price	block	street_name	floor_area_sqm
1	Medium	119	TECK WHYE...	104.000
2	Medium	22	HAVELOCK ...	64.000
3	Medium	906	JURONG WE...	141.000
4	Low	510	JURONG WE...	74.000
5	Medium	232	JURONG EA...	95.000
6	High	284	TOH GUAN ...	120.000
7	High	326	ANG MO KIO...	92.000
8	Low	668	CHANDER RD	75.000
9	High	428	ANG MO KIO...	92.000
10	High	2	HAIG RD	92.000
11	Medium	480	SEGAR RD	94.000
12	Medium	33	MARINE CRES	65.000
13	Medium	695	HOUGANG S...	104.000
14	Medium	412B	FERNVALE L...	114.000
15	Medium	103	BEDOK RES...	93.000
16	High	361	WOODLAND...	145.000

	name	High	Low	Medium
1	intercept	-6.9684047	6.9358730	-1.1609071
2	floor_area_s...	0.0615403	-0.0841379	0.0050480

5. We can then evaluate our model by clicking on the Confusion Matrix and ROC Curve widgets. According to the Confusion Matrix, most of the error seems to come from misclassifying Medium-priced flats as High.

		Predicted			
		High	**Low**	**Medium**	**Σ**
Actual	**High**	493	83	80	**656**
	Low	70	533	64	**667**
	Medium	334	168	175	**677**
	Σ	**897**	**784**	**319**	**2000**

6. An ROC Curve should ideally hug the upper-left corner of the box, so it looks like our curve isn't too bad looking.



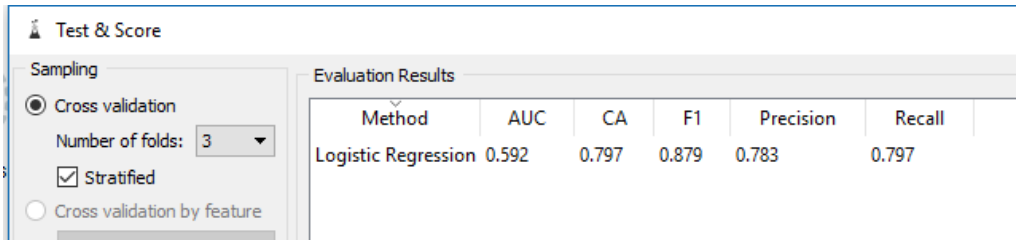
7. The model can then be provided to your developer to build into your process/system.

Workflow (Adult Income)

1. Now let's try a more complex example. Use back the same workflow, but reload the data as follows:
    a. **File** widget: adult.tab

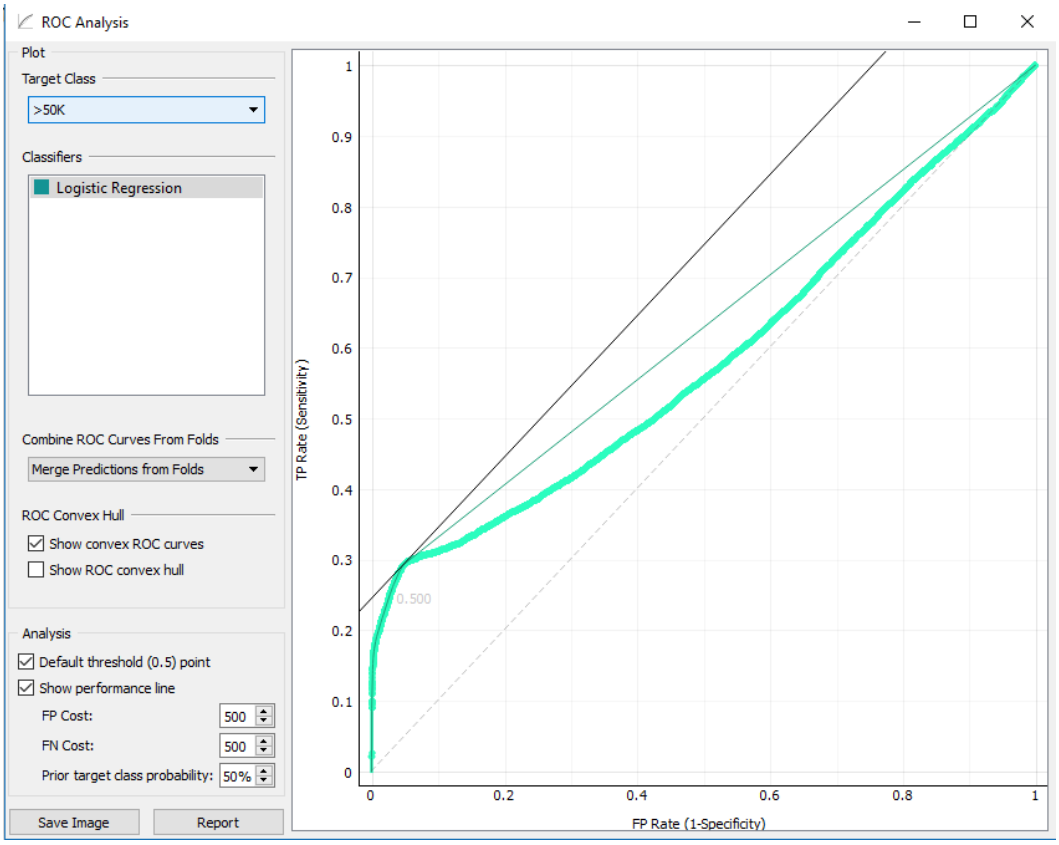
   The adult.tab data consists of demographic data and whether the person earn more than $50,000 or not. So to ask a question like "Does he/she earn more than $50,000?" – the Logistic Regression algorithm is an appropriate choice.

2. Review the Test & Score widget for the Evaluation results.

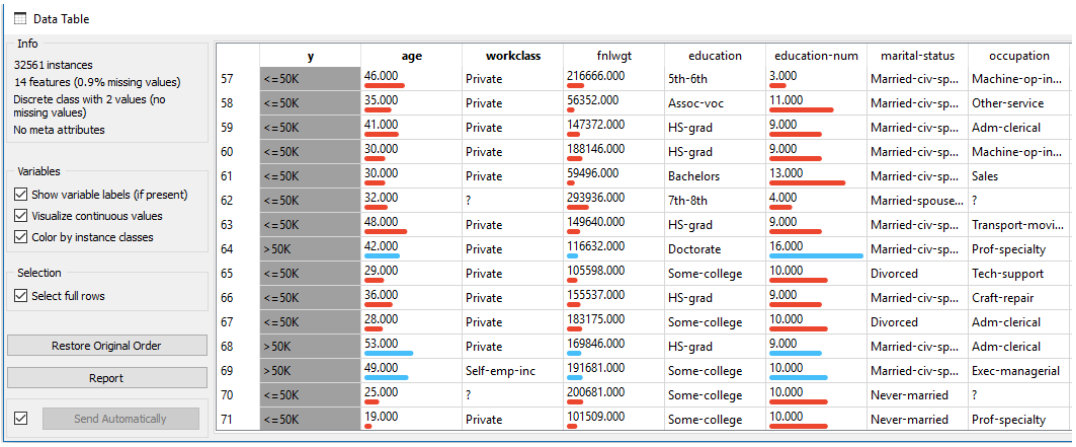


3. Intuition (after working on data and models for a while) will tell you that the model is not working well. The AUC is a low score of 0.592.

4. See the ROC Analysis widget.

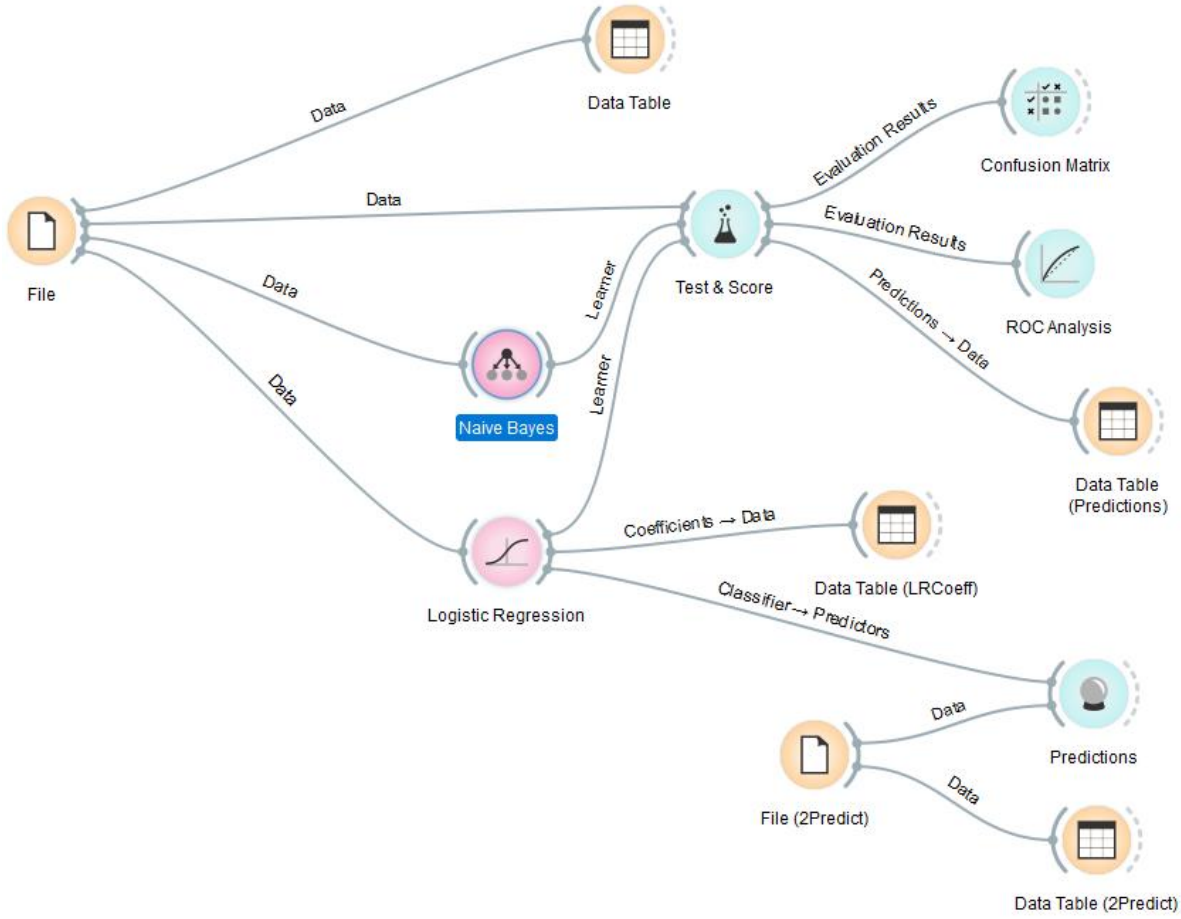


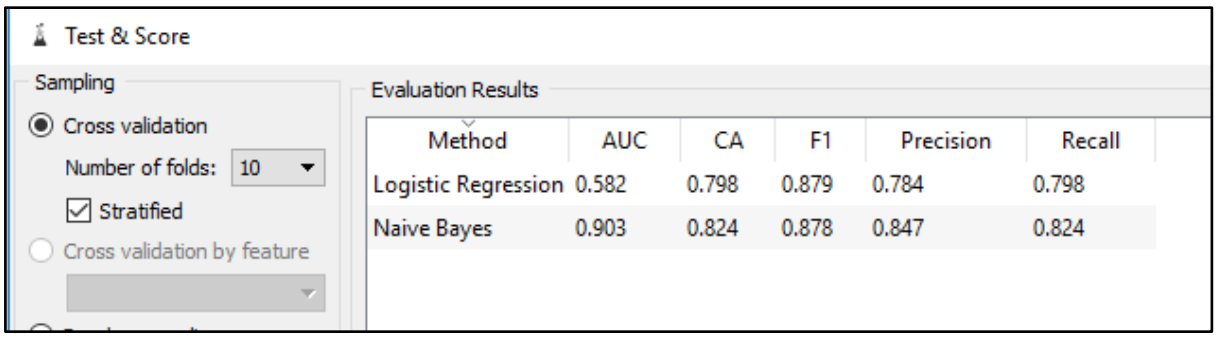
5. Let's investigate further. Look at the data in **Data Table** widget. Note that there are missing data and columns which may not really matter to the model.



	y	age	workclass	fnlwgt	education	education-num	marital-status	occupation
57	<=50K	46.000	Private	216666.000	5th-6th	3.000	Married-civ-sp...	Machine-op-in...
58	<=50K	35.000	Private	56352.000	Assoc-voc	11.000	Married-civ-sp...	Other-service
59	<=50K	41.000	Private	147372.000	HS-grad	9.000	Married-civ-sp...	Adm-clerical
60	<=50K	30.000	Private	188146.000	HS-grad	9.000	Married-civ-sp...	Machine-op-in...
61	<=50K	30.000	Private	59496.000	Bachelors	13.000	Married-civ-sp...	Sales
62	<=50K	32.000	?	293936.000	7th-8th	4.000	Married-spouse...	?
63	<=50K	48.000	Private	149640.000	HS-grad	9.000	Married-civ-sp...	Transport-movi...
64	>50K	42.000	Private	116632.000	Doctorate	16.000	Married-civ-sp...	Prof-specialty
65	<=50K	29.000	Private	105598.000	Some-college	10.000	Divorced	Tech-support
66	<=50K	36.000	Private	155537.000	HS-grad	9.000	Married-civ-sp...	Craft-repair
67	<=50K	28.000	Private	183175.000	Some-college	10.000	Divorced	Adm-clerical
68	>50K	53.000	Private	169846.000	HS-grad	9.000	Married-civ-sp...	Adm-clerical
69	>50K	49.000	Self-emp-inc	191681.000	Some-college	10.000	Married-civ-sp...	Exec-managerial
70	<=50K	25.000	?	200681.000	Some-college	10.000	Never-married	?
71	<=50K	19.000	Private	101509.000	Some-college	10.000	Never-married	Prof-specialty

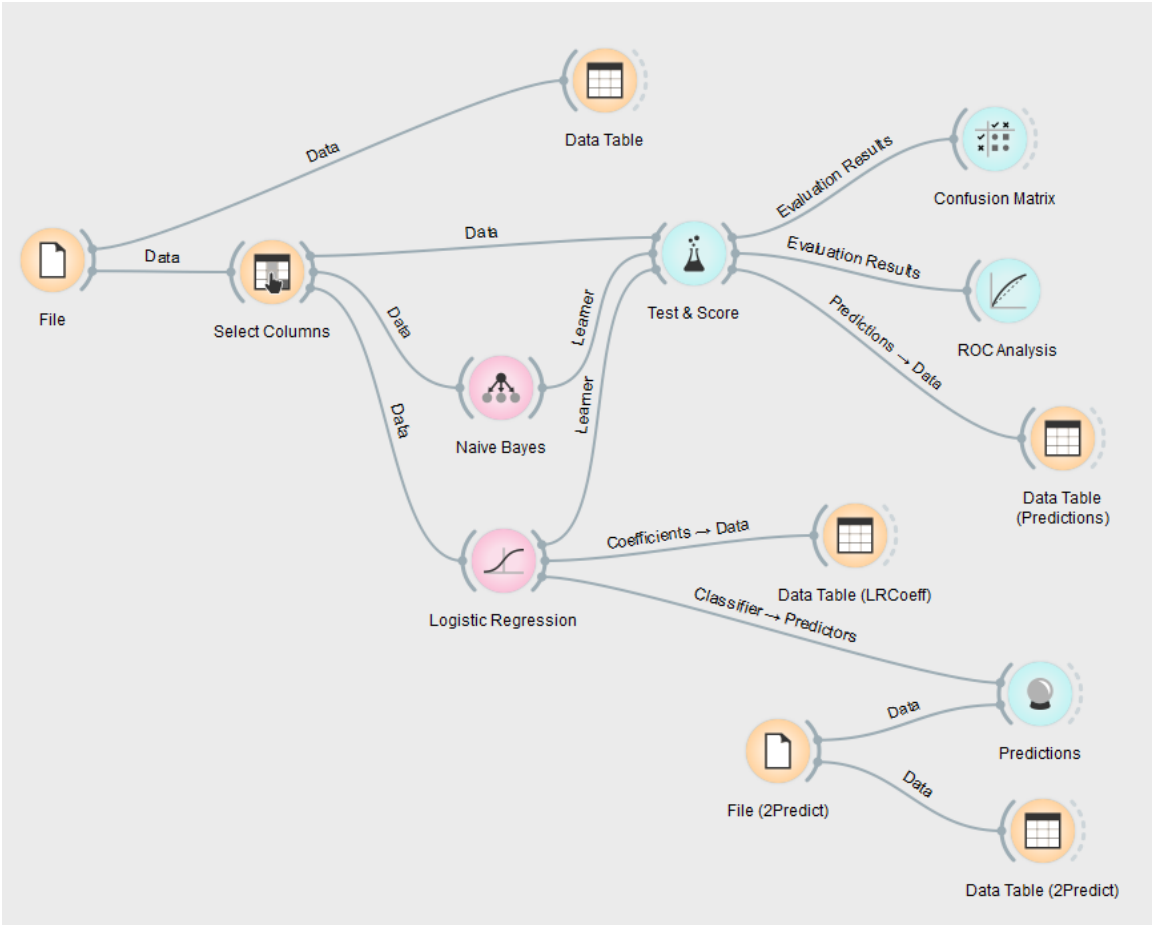
6. Let's validate our thinking. Add a **Naïve Bayes** widget to the workflow and compare the performances.



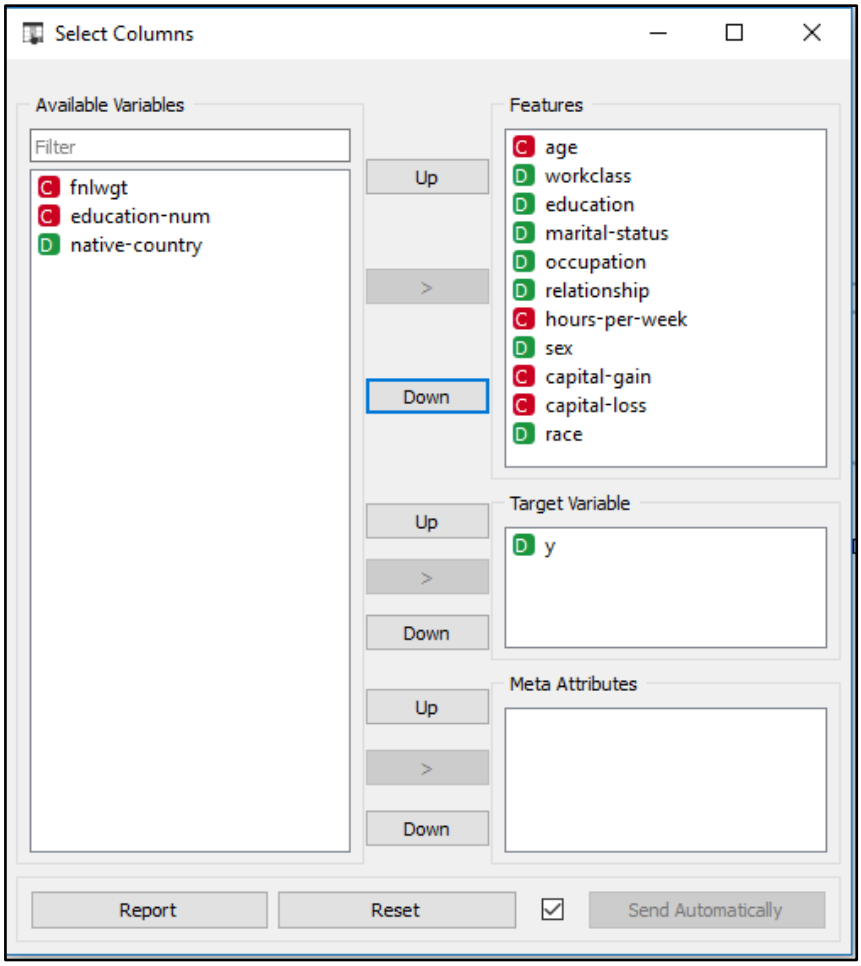


7. I would expect **Logistic Regression** to be close to or better than **Naïve Bayes**. Let's do some feature engineering to see if we can improve the scores. You may want to FREEZE the signal propagation while adding and re-arranging the widgets to prevent Orange from re-computing the attached models while you make your changes. When you are done, unfreeze the canvas.

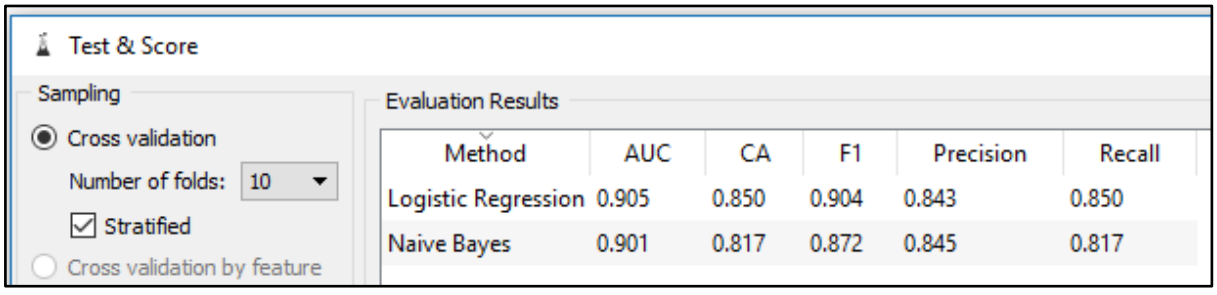
Note that when the canvas is in FREEZE mode, the background is grey in colour.



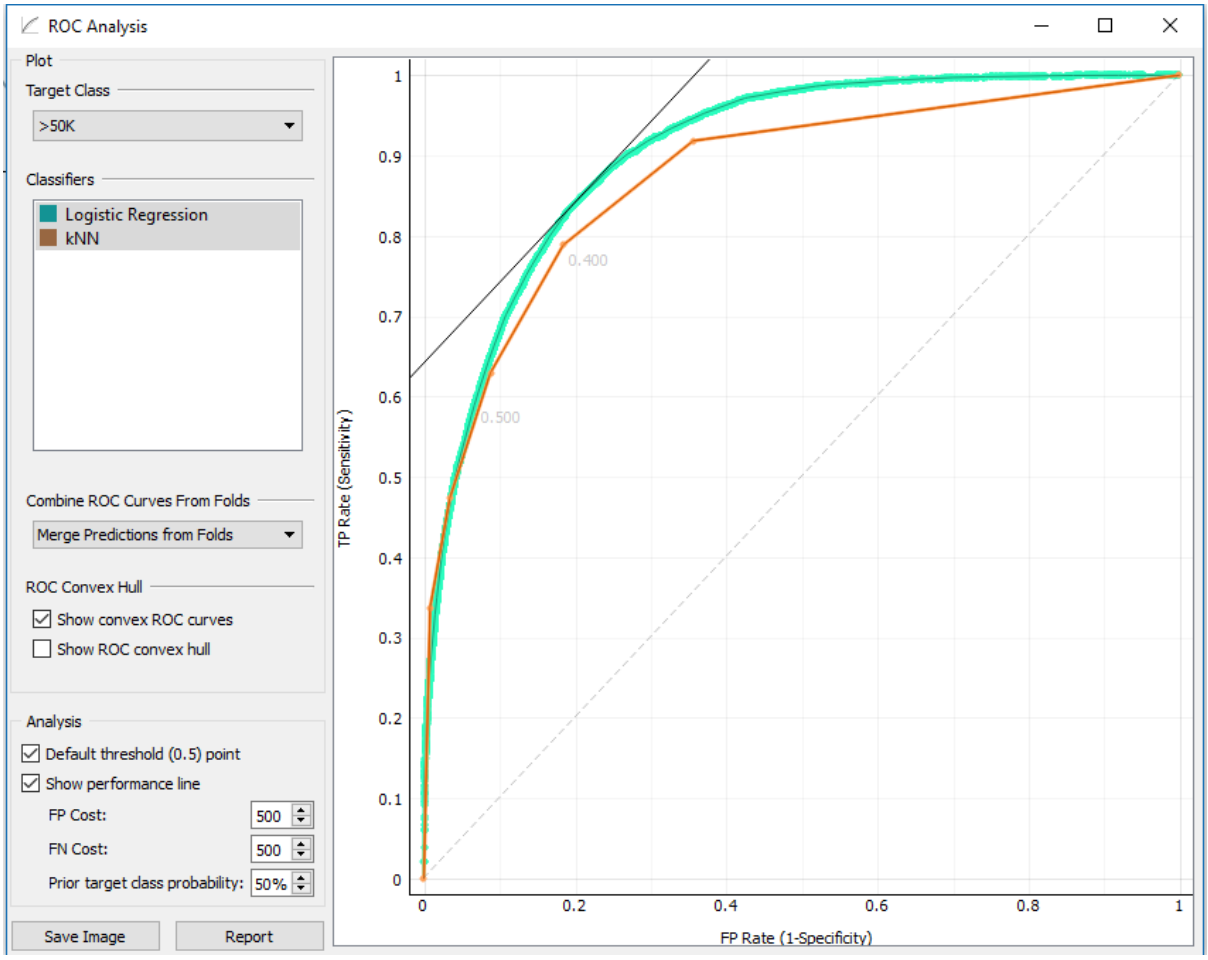
8. In **Select Columns** widget, setup the variables as shown below.



9. Review the results again. See that by removing unnecessary columns we have improved the performance of the Logistic Regression model from AUC=0.588 to the new AUC=0.905 and with a better recall rate.



10. Compare also the previous and current ROC curves.



Conclusion

The statement "Garbage in, garbage out" is shown here. Some algorithms are more sensitive to garbage then others (Logistic Regression vs Naïve Bayes).

Using all the data (big data) we have sometimes lead to poorer results.

Feature engineering can improve "performance" of the algorithm hence accuracy.

Use of multiple algorithms sometimes helps you to identify key features and you may want to think further why those features (columns) are important!

TREES AND RANDOM FOREST

Motivation

Tree is a simple algorithm that splits the data into nodes by class purity. It is conceptually easy to understand and can be shown graphically to explain the results.

Random Forest is an enhanced version and builds a set of trees. Random Forest was for a long time the go-to algorithm for many data scientist.

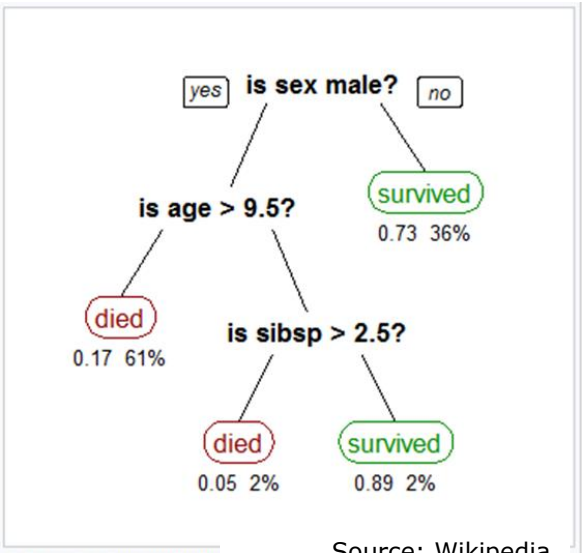
Theory

Trees

In classification - tree analysis is when the predicted outcome is the class to which the data belongs.

In regression tree - analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The split is based on information theory – basically the split is done which generates the biggest differentiator at each branch.



Source: Wikipedia

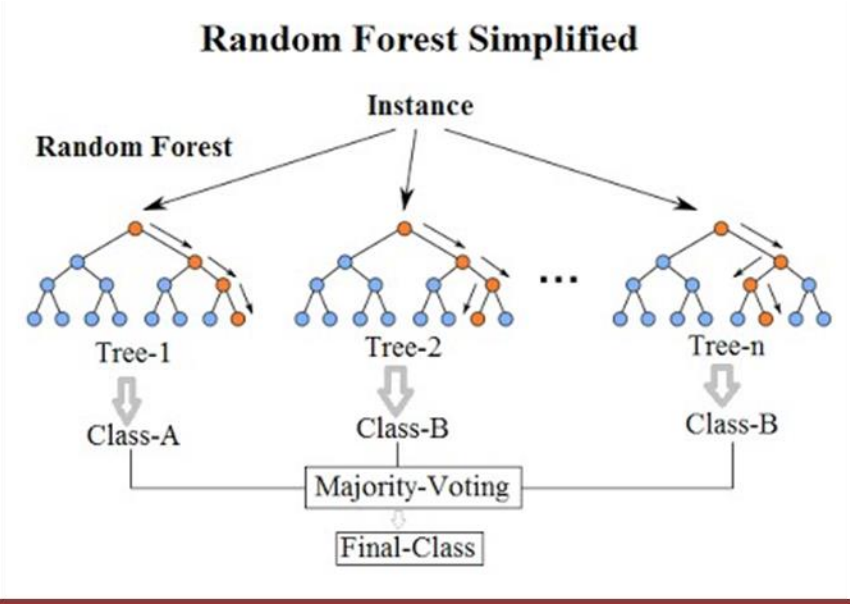
A tree showing survival c
the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

Random Forest

Random Forest builds a set of decision trees. Each tree is developed from a bootstrap sample from the training data.

When developing individual trees, an arbitrary subset of attributes is drawn (hence the term "Random"), from which the best attribute for the split is selected.

The final model is based on the majority vote from individually developed trees in the forest.

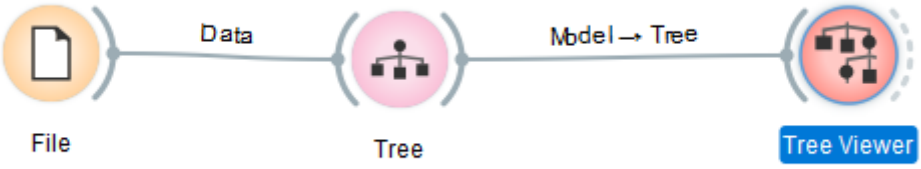


Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of overcoming over-fitting problem of individual decision tree.

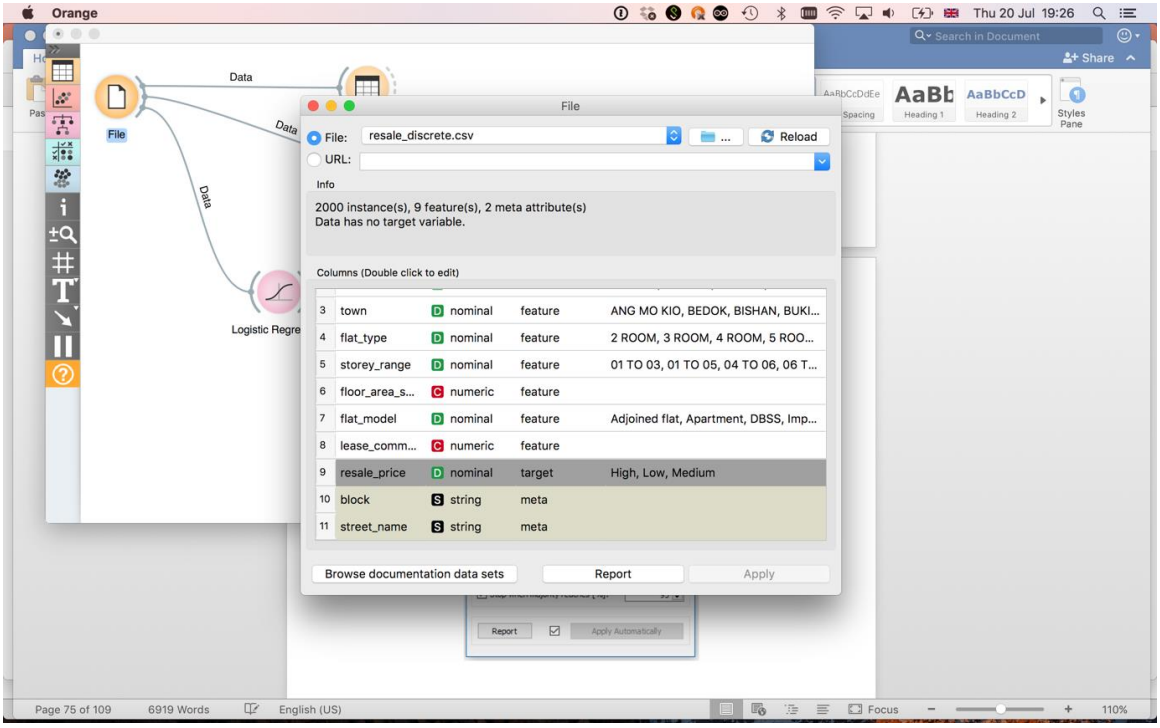
Source: www.listendata.com

Workflow

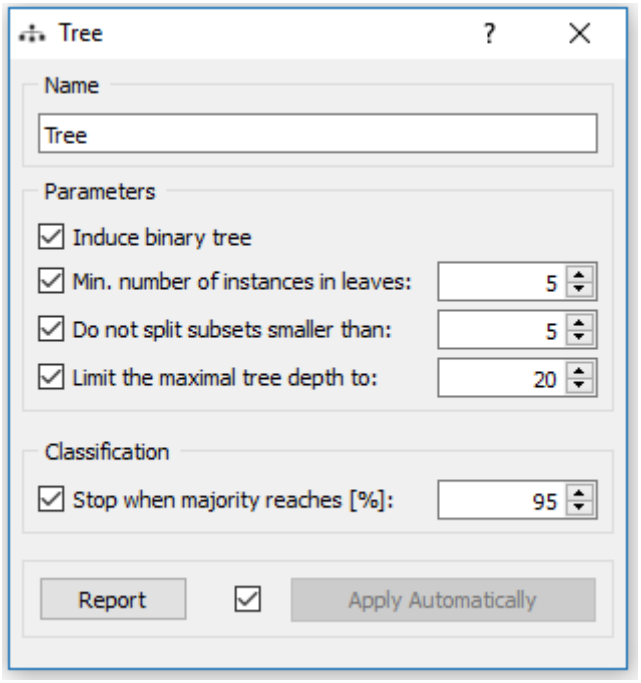
1. Build the following with the **resale_discrete.csv** dataset. Here we are using Tree to induce the model and displaying it in a Tree Viewer. This is often part of the data exploration phase with a labeled dataset.



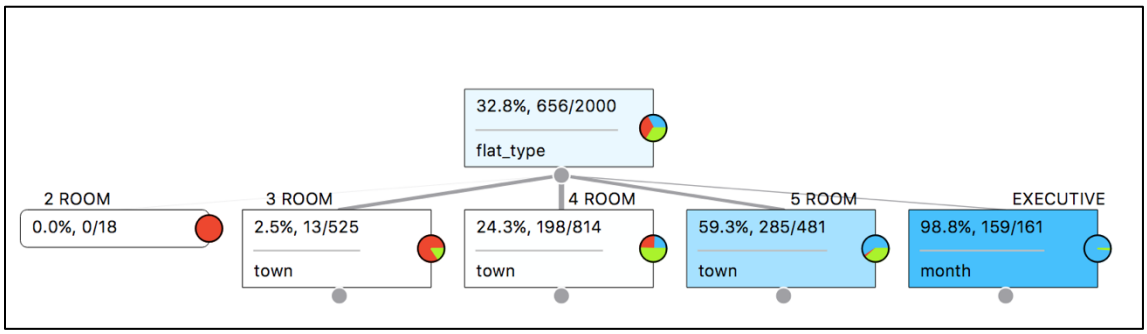
2. An advantage of Tree models is that they can handle both discrete and continuous variables. Hence we can specify the discrete variables in our dataset as features.



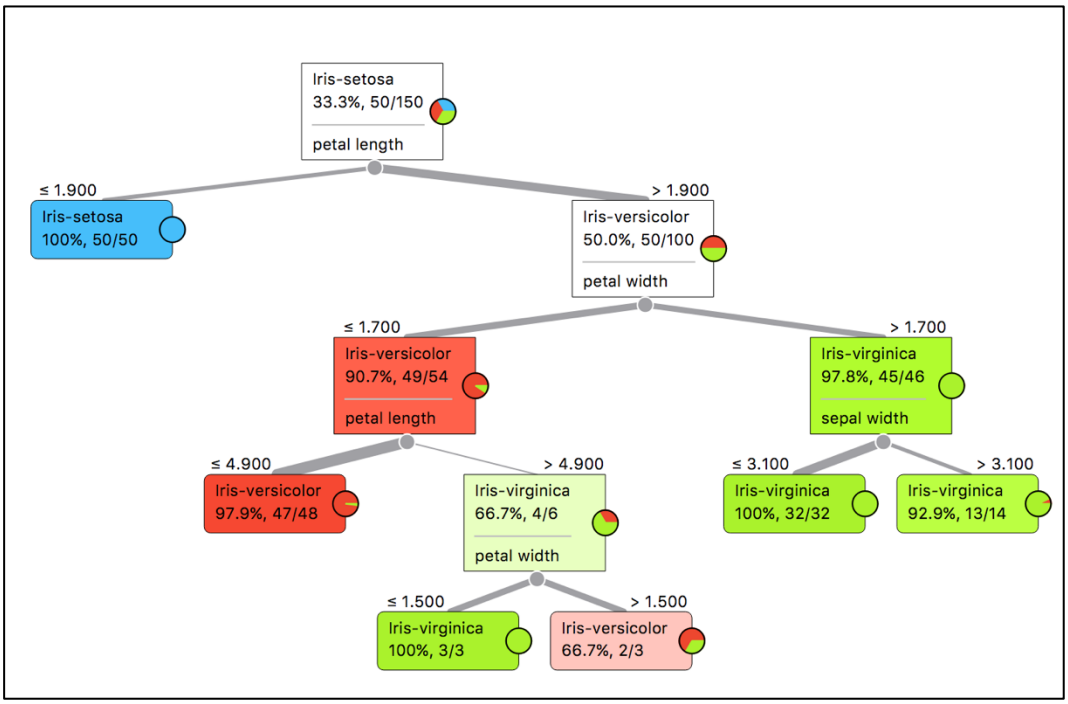
3. Click on Tree widget and open the dialog. You can leave the defaults for this exercise. Just be aware in Tree type algorithms, parameters to tweak includes the number of instances in leaves, tree depth, how to split.



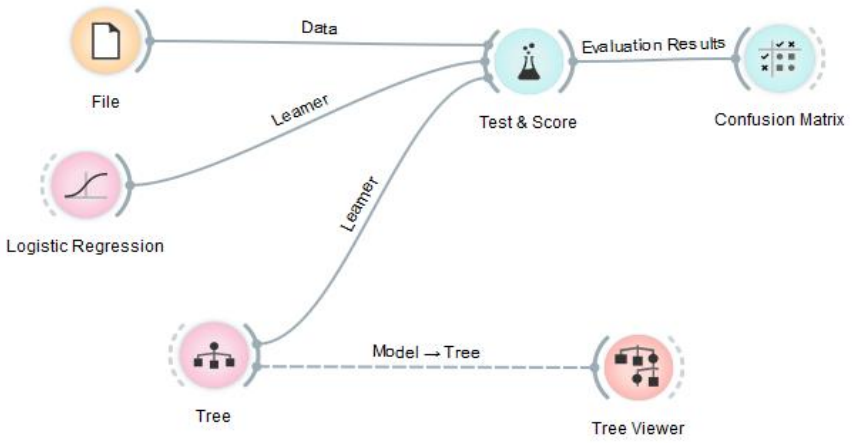
4. For this exercise, in the Tree Viewer widget, we limit the view to 2 levels to see the data more easily. You can try adding more levels, though notice this makes the window much bigger as the next split is by the town variable



5. To see an ideal Tree, see this example of the iris dataset, which predicts flower species according to variables like petal width.



6. Let's add Logistic Regression and compare.

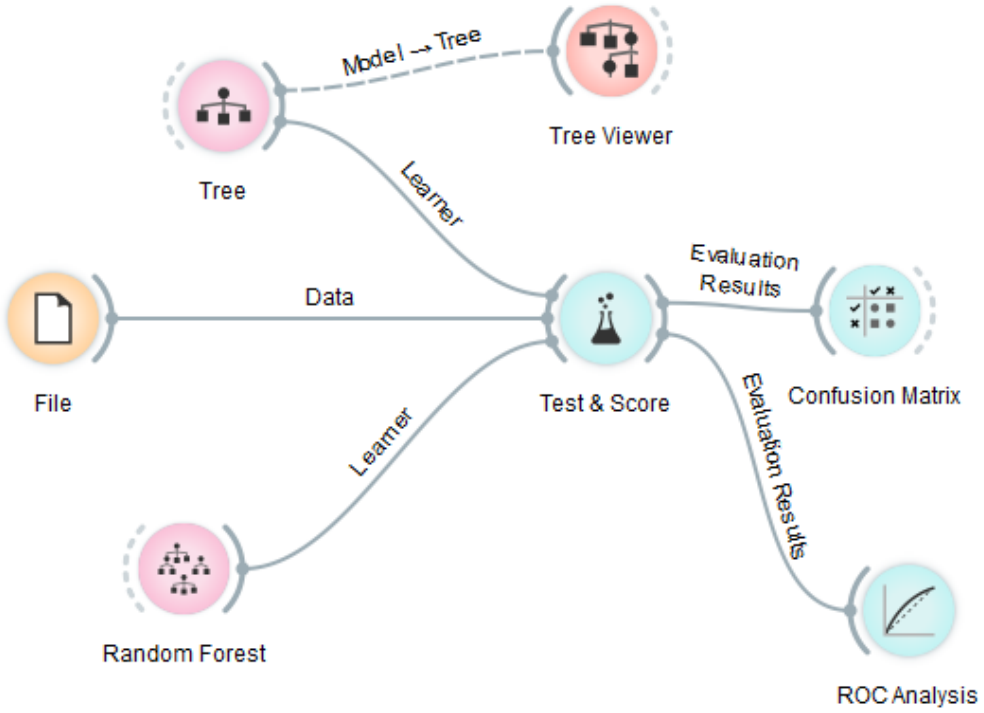


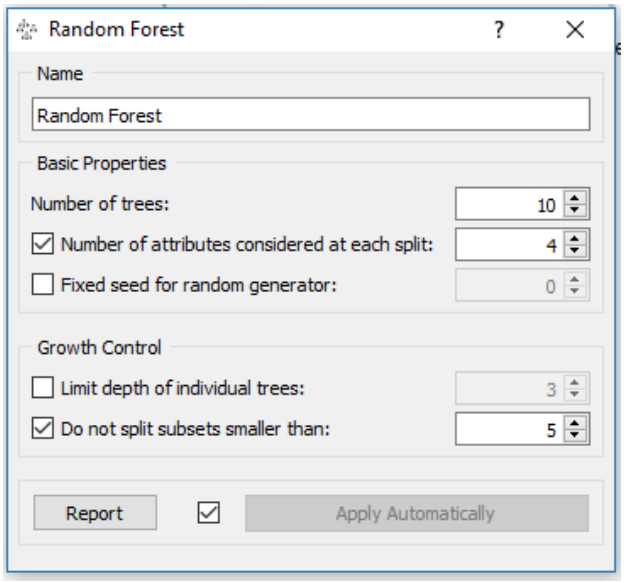
Looks like Logistic Regression over predicts the High class; the Tree algorithm meanwhile has more even predictions across classes.

Note that Orange's implementation of **Tree** allows **Tree** to be used for Regression tasks also. Please test out **Tree** on a suitable regression tasks – you may wish to use the auto-mpg dataset.

Workflow (Random Forest)

1. Build the following workflow with the same dataset: resale-discrete.csv. You can leave the Random Forest parameters in their defaults.





File: resale_discrete.csv ... Reload

URL:

Info

2000 instance(s), 9 feature(s), 2 meta attribute(s)
Data has no target variable.

Columns (Double click to edit)

1	index	numeric	skip	
2	month	nominal	feature	2012-03, 2012-04, 2012-05, 2012-0...
3	town	nominal	feature	ANG MO KIO, BEDOK, BISHAN, BUKI...
4	flat_type	nominal	feature	2 ROOM, 3 ROOM, 4 ROOM, 5 ROO...
5	storey_range	nominal	feature	01 TO 03, 01 TO 05, 04 TO 06, 06 T...
6	floor_area_s...	numeric	feature	
7	flat_model	nominal	feature	Adjoined flat, Apartment, DBSS, Imp...
8	lease_comm...	datetime	feature	
9	resale_price	nominal	target	High, Low, Medium

In this example**, Random Forest** seems to edge out **Tree** by a little. Generally, most Random Forest models provides better results.

In addition, **Tree** encounters the overfitting problem and ignorance of a variable in case of small sample size and large p-value (not important features). Whereas, **Random Forests** are a type of recursive partitioning method particularly well-suited to small sample size and large p-value problems

Conclusion

You have used the Tree widget to induce a graphical view of the data to help you explore and understand the dataset. You have also built both a Tree and Random Forest models to predict the occurrence of breast cancer reoccurring in the breast-cancer dataset.

At this point, it should be clear that modeling is relatively easy. The hard part which **YOU DID NOT** do was to gather the data, clean it and prepare it for your model.

Recall: Modeling and refining the model may only take 30% of the time, while data preparation can take 70% of your project time!

SUPPORT VECTOR MACHINES (SVM)

Motivation

Support vector machines (SVMs) are a set of supervised learning methods used for

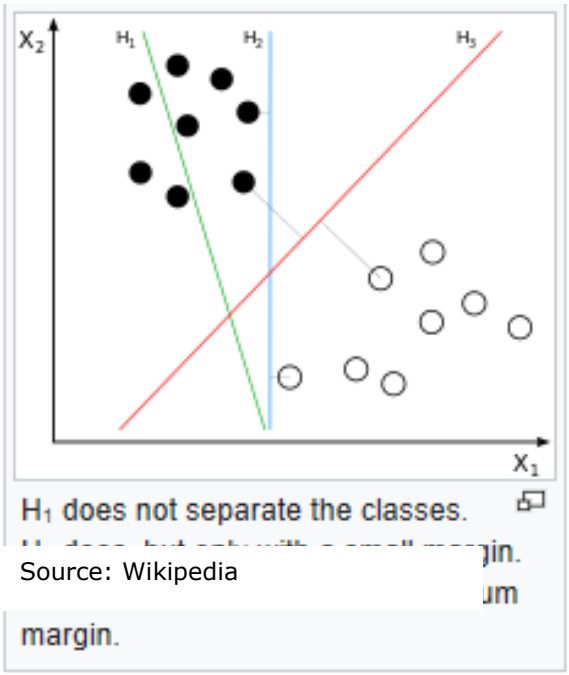
- Classification
- Regression
- Outliers detection

We have used it successfully in several projects, including the Intel Factory Optimization project where we used SVM to classify good / bad CPUs on the production line.

Together with Random Forest, SVM is a very popular classification and regression algorithm popular in the last two decade.

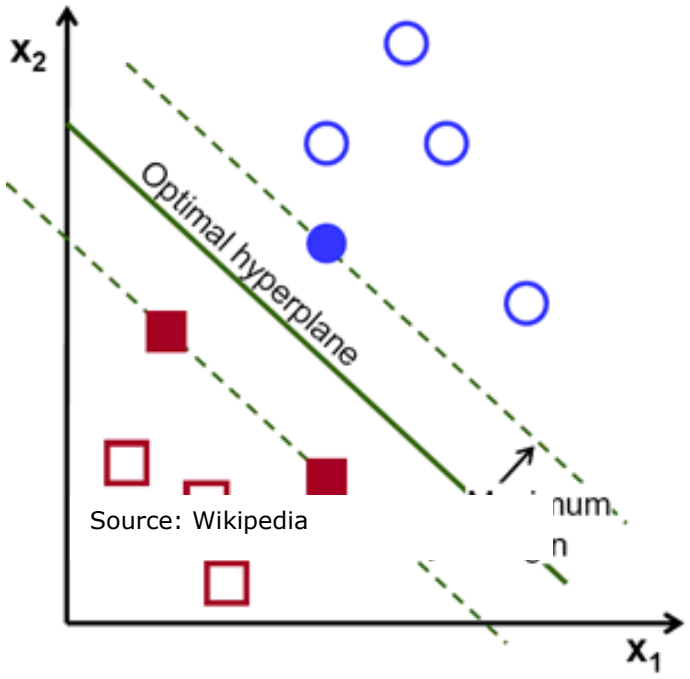
Theory

SVM constructs a set of hyperplanes in a high-dimensional space.



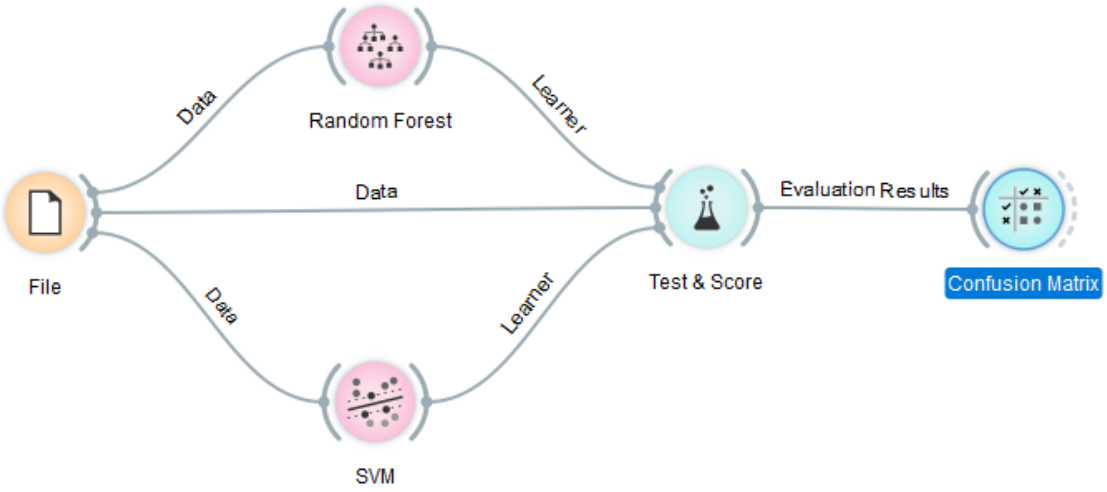
Source: Wikipedia

Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class

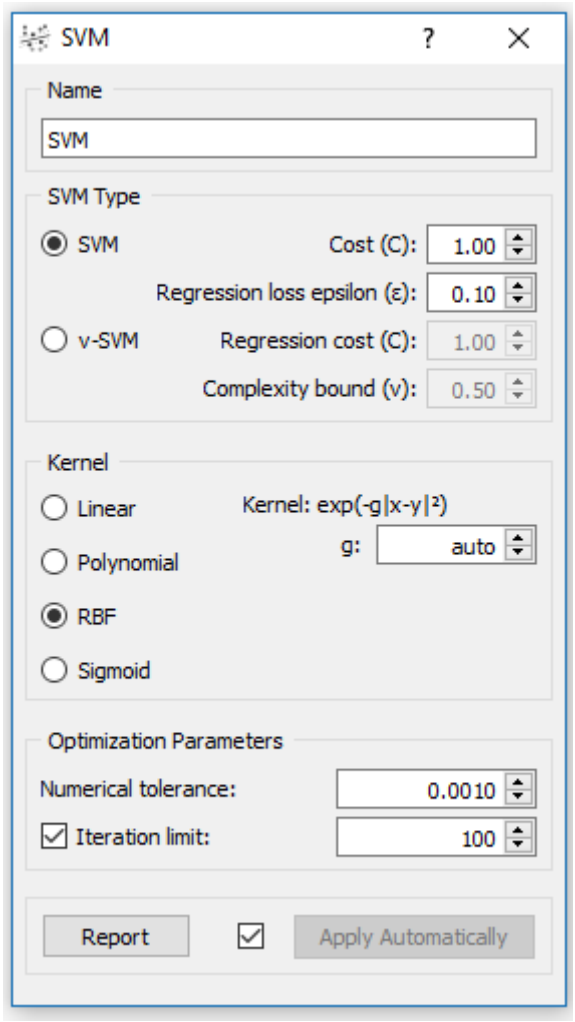


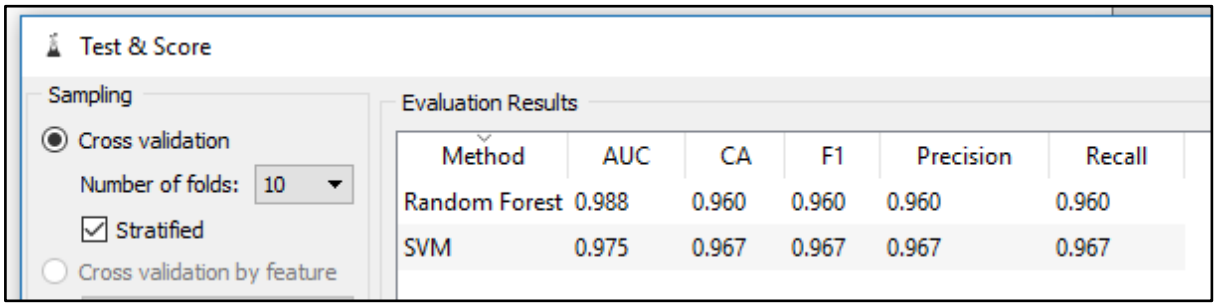
Workflow (Classification)

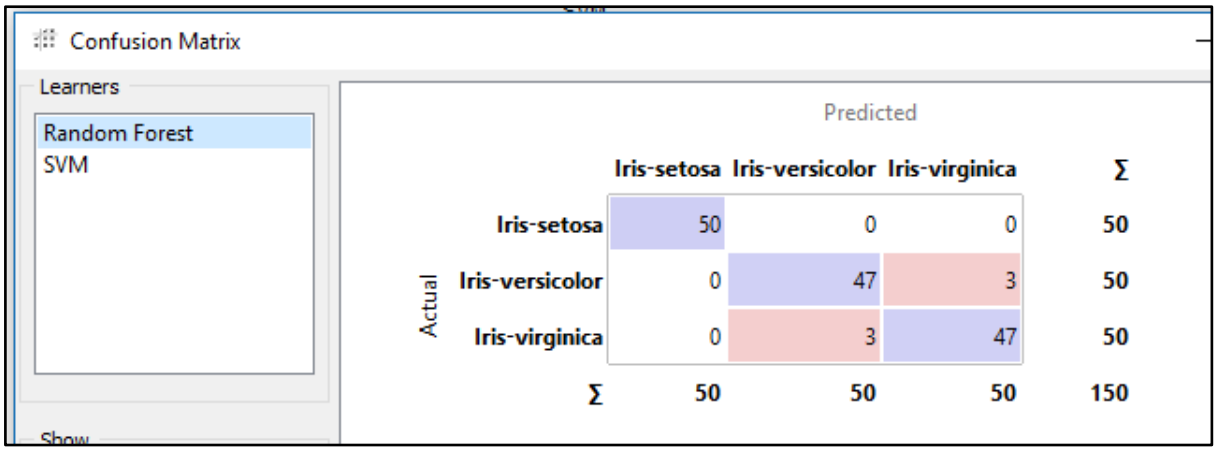
1. Build the following workflow with the iris dataset and compare the results.

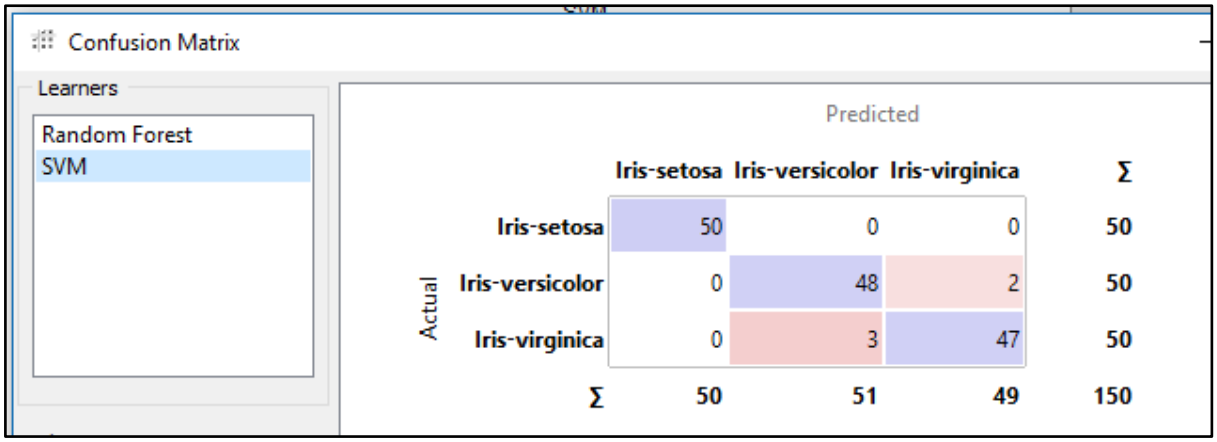


The SVM dialog is shown. The RBF (radial basis function) is a common option, and works well for most cases.







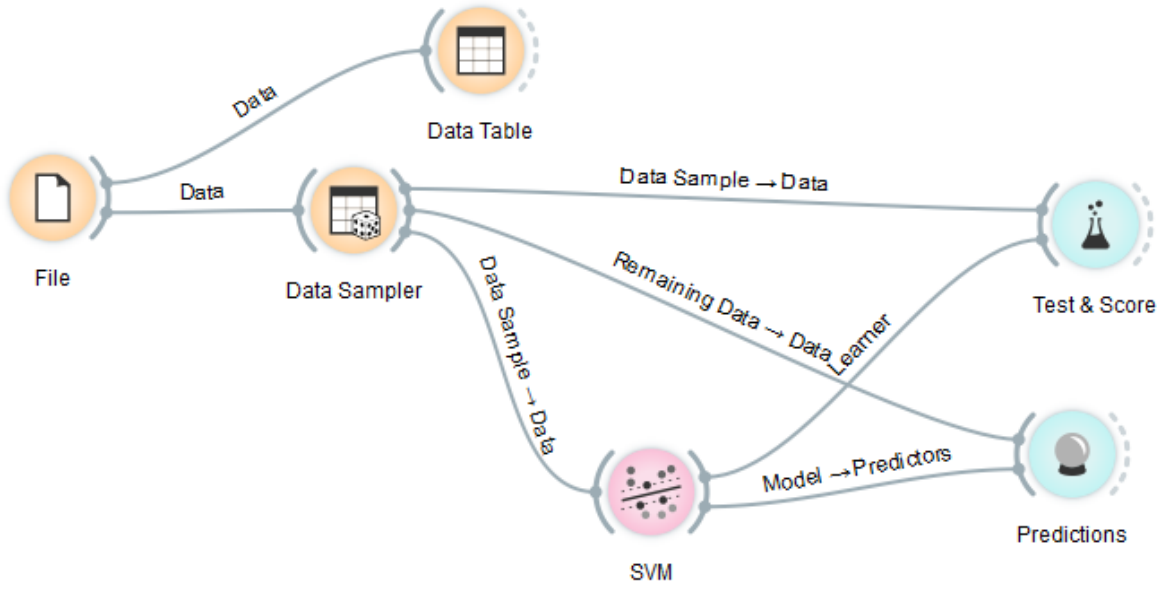


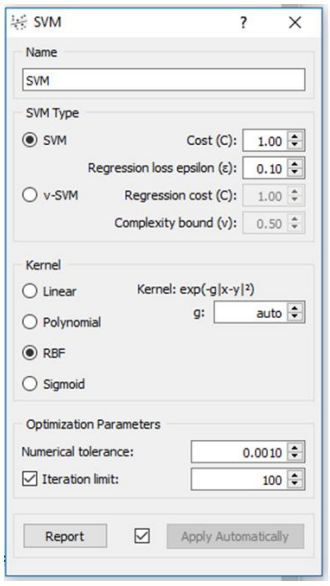
See that both Random Forest and SVM perform well. Which you choose depends on your personal preferences in this case.

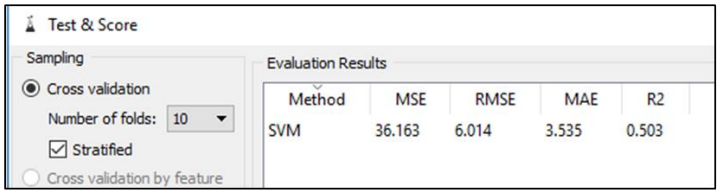
A possible strategy is to use an ensemble, where you combine the outputs of multiple algorithms, for example Random Forest, Linear Regression and SVM and then take the average (regression) or majority-wins (classification) approach. This was the approach in the Intel Penang project.

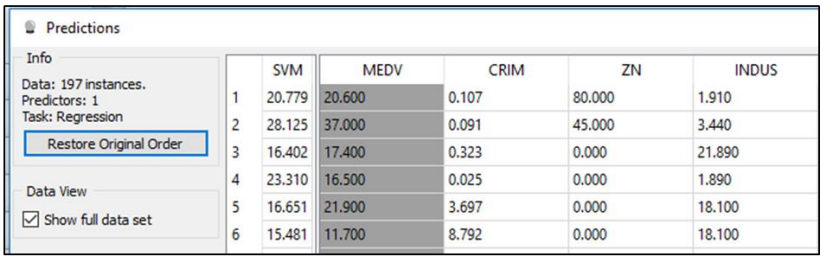
Workflow (Regression)

1. SVM can also be used for regression. Build the following workflow and use the housing dataset. Here we use the Data Sampler widget to split the dataset into Training/Test set and a Prediction set.

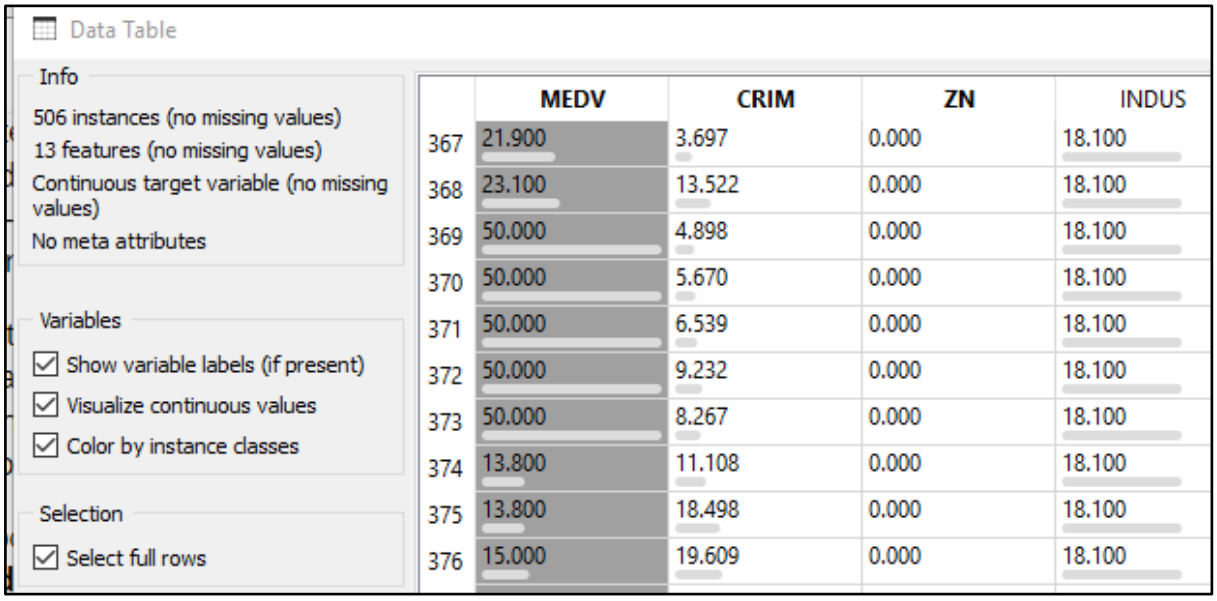






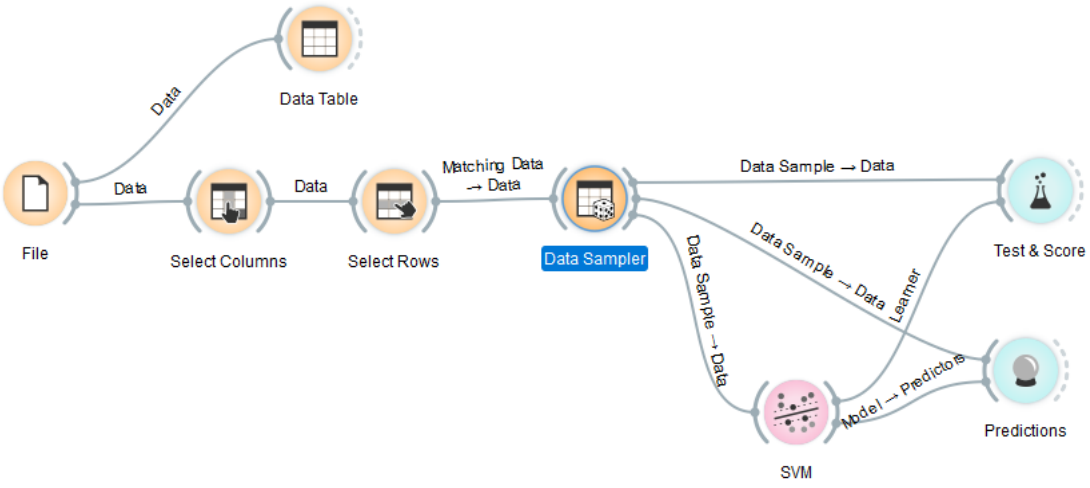


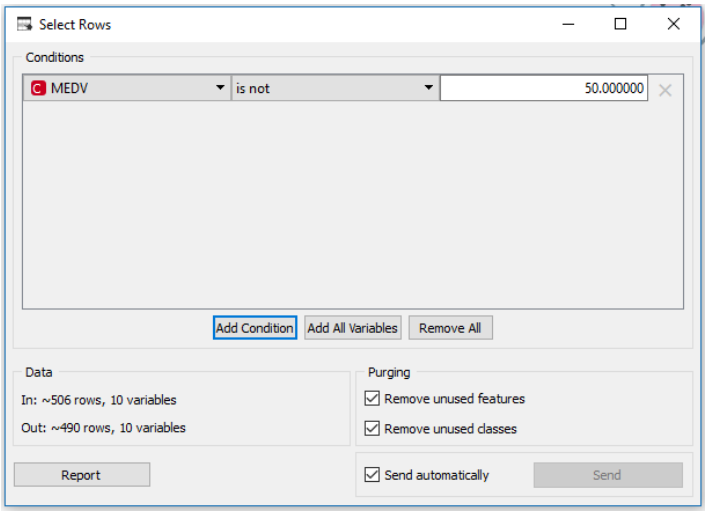
2. The results do not look promising for SVM. This is where the domain understanding and data cleaning and preprocessing comes in. Review the Data Table output again. Notice that there several data points with MEDV value of 50.000. Further investigation reveals the data was missing and the collector of the data just entered a value of 50,000. You decide to remove these values.

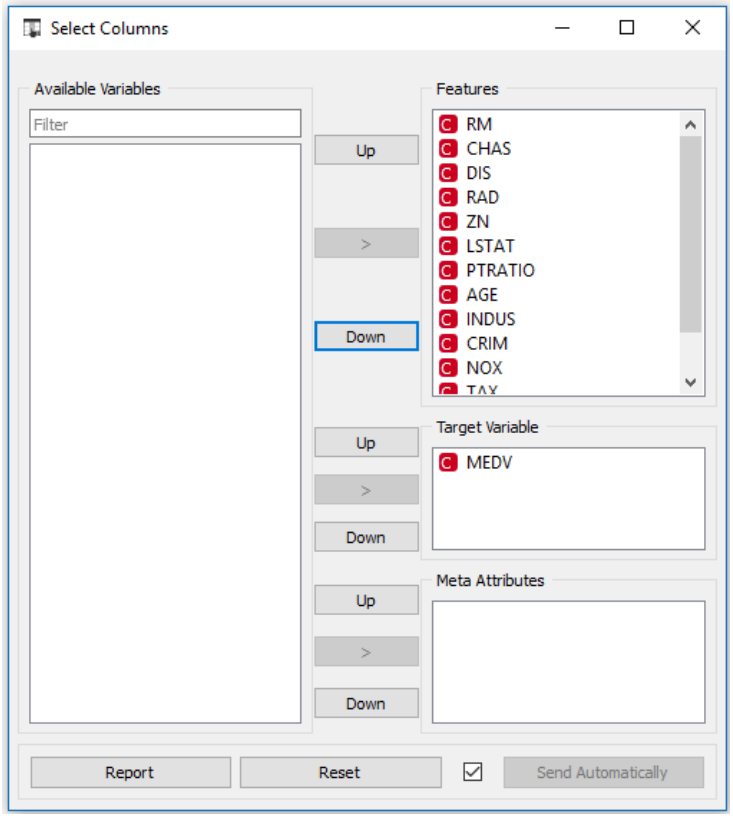


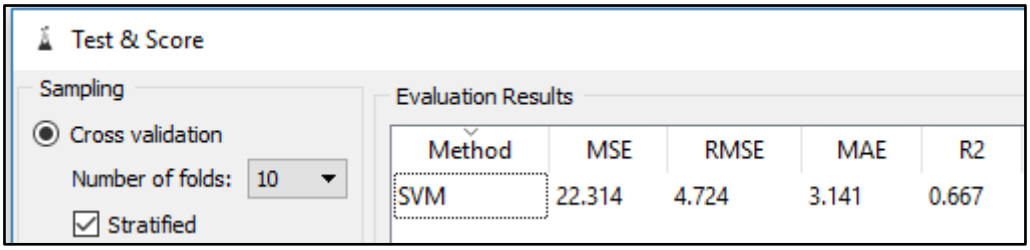
	MEDV	CRIM	ZN	INDUS
367	21.900	3.697	0.000	18.100
368	23.100	13.522	0.000	18.100
369	50.000	4.898	0.000	18.100
370	50.000	5.670	0.000	18.100
371	50.000	6.539	0.000	18.100
372	50.000	9.232	0.000	18.100
373	50.000	8.267	0.000	18.100
374	13.800	11.108	0.000	18.100
375	13.800	18.498	0.000	18.100
376	15.000	19.609	0.000	18.100

3. Construct the new workflow and filter out all MDEV=50,000

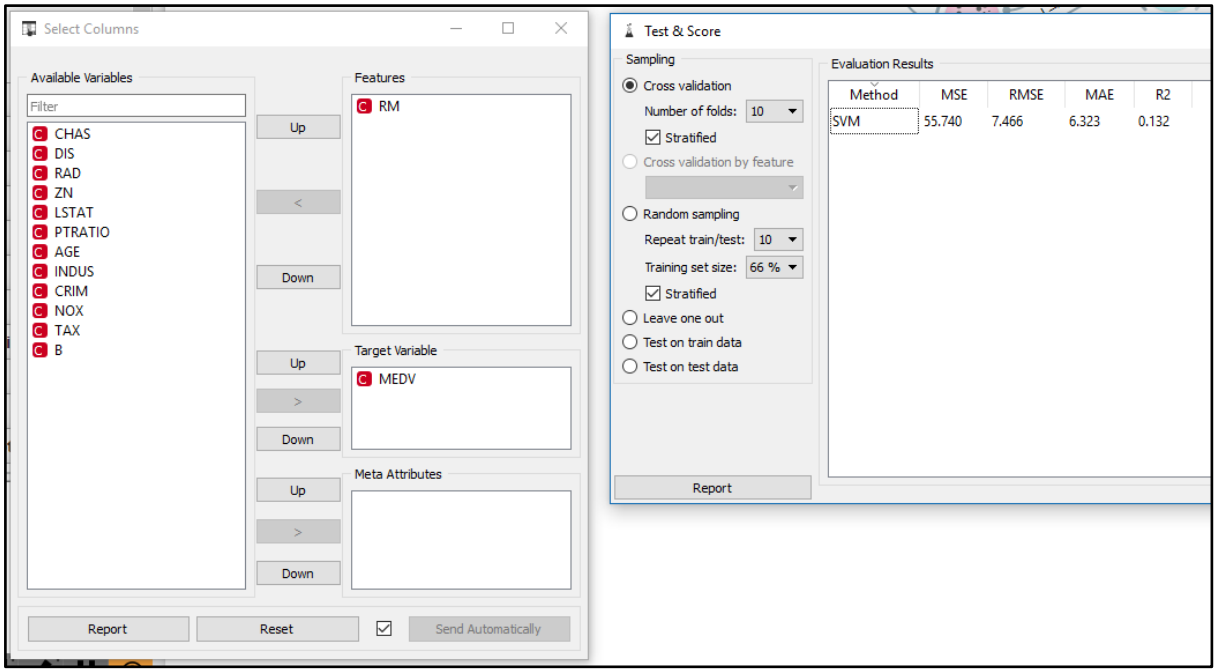








4. However, you still feel you can improve the model. Let's remove all the **Features** and leave only RM. Using a trial and error, you add new features to the model and see how the **Evaluation Results** changes.



Conclusion

You have built two SVM models. One for classification and one for regression. You learnt that just dumping data blindly into the algorithm may work but most time you will not get the best results.

You learned the importance of pre-processing your data and this drove the performance of your model much higher.

My best score :

Evaluation Results

Method	MSE	RMSE	MAE	R2
SVM	14.769	3.843	2.703	0.755

CN2 RULE INDUCTION

Motivation

You learned the Association Rules algorithm in the UNSUPERVISED learning section. The CN2 Rule Induction algorithm is a SUPERVISED learning algorithm. It provides an easy to understand explanation of the **rules** found in your dataset.

Theory

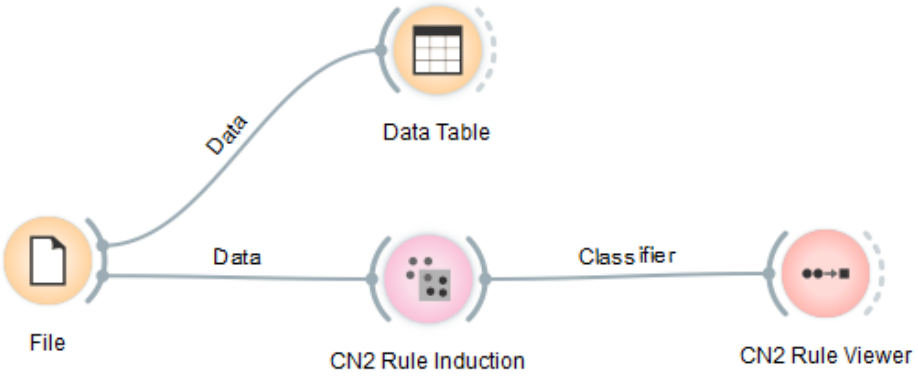
The CN2 algorithm is a classification technique designed for the efficient induction of simple, comprehensible rules of form **IF {condition} THEN predict {class},** and the associated probability.

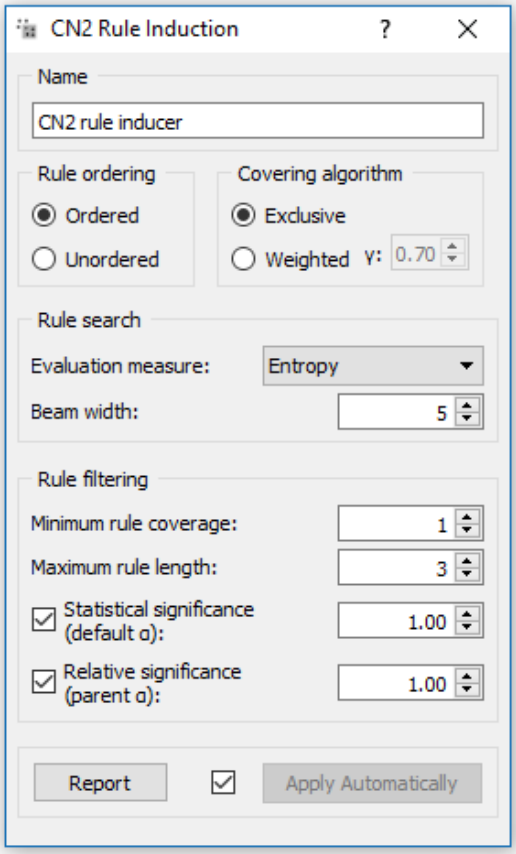
CN2 works even in domains where noise may be present in the data, or when data is missing. CN2 is for **CLASSIFICATION** tasks only! The CN2 pseudocode is shown below.

```
Let E be a set of classified examples.
Let SELECTORS be the set of all possible selectors.

Procedure CN2(E)
  Let RULE_LIST be the empty list.
  Repeat until BEST_CPX is nil or E is empty:
     Let BEST_CPX be Find_Best_Complex(E).
     If BEST_CPX is not nil,
     Then let E' be the examples covered by BEST_CPX.
          Remove from E the examples E' covered by BEST_CPX.
          Let C be the most common class of examples in E'.
          Add the rule 'If BEST_CPX then the class is C'
              to the end of RULE_LIST.
  Return RULE_LIST.

Procedure Find_Best_Complex(E)
  Let STAR be the set containing the empty complex.
  Let BEST_CPX be nil.
  While STAR is not empty,
     Specialize all complexes in STAR as follows:
     Let NEWSTAR be the set {x ∧ y|x ∈ STAR, y ∈ SELECTORS}.
     Remove all complexes in NEWSTAR that are either in STAR (i.e.,
        the unspecialized ones) or null (e.g., big = y ∧ big = n).
     For every complex C_i in NEWSTAR:
          If C_i is statistically significant and better than
             BEST_CPX by user-defined criteria when tested on E,
          Then replace the current value of BEST_CPX by C_i.
     Repeat until size of NEWSTAR ≤ user-defined maximum:
        Remove the worst complex from NEWSTAR.
     Let STAR be NEWSTAR.
  Return BEST_CPX.
```

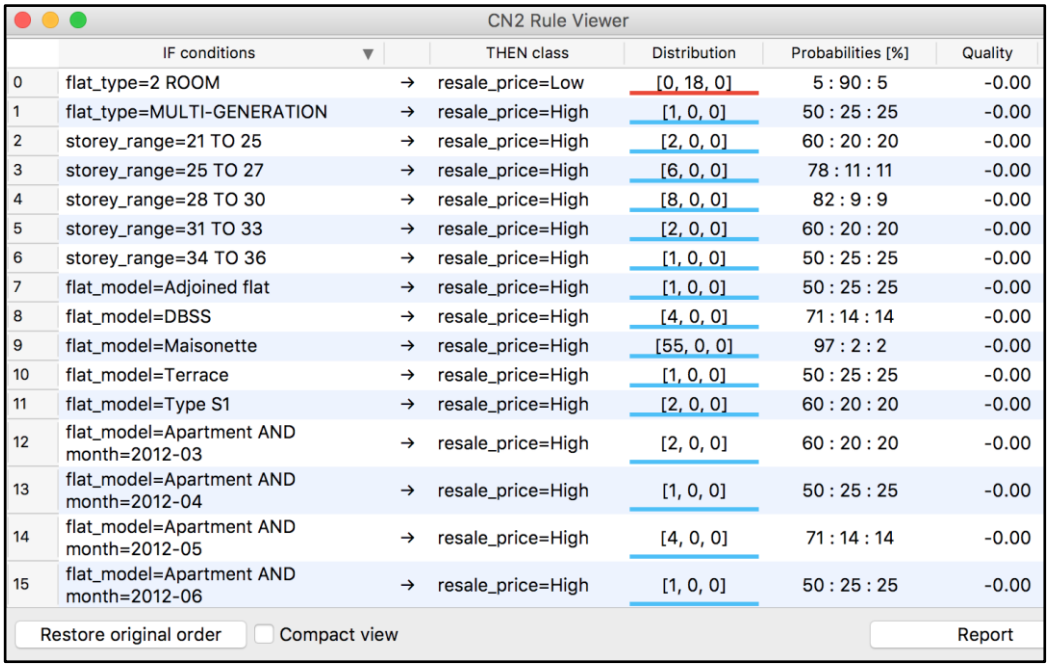
Workflow

1. Build the **CN2 Induction** workflow as shown. You can leave the CN2 parameters as is.





2. Explore the rules.



3. Keep both the CN2 Rule Induction dialog and CN2 Rule Viewer open side by side. Make changes to CN2 parameters and observe what happens. The choice of your parameters affects the number of rules generated and the number of conditions of the rules.

   Your choice will depend very much on your requirements, sometimes too many rules makes it hard to implement and update, while a smaller set of rules may not capture all your edge cases.

   There is no right or wrong. It is an art.

4. You may wish to add in the Tree Widget and Tree Viewer to compare.

Conclusion

You have built a CN2 Rule Induction workflow and explored how changing its parameters changes the generation of rules.

Rules are useful from the point of sharing your insights, as you can easily interpret them and share what the rules are in layman terms.

IF {condition} THEN predict {class}

RECOMMENDER SYSTEMS

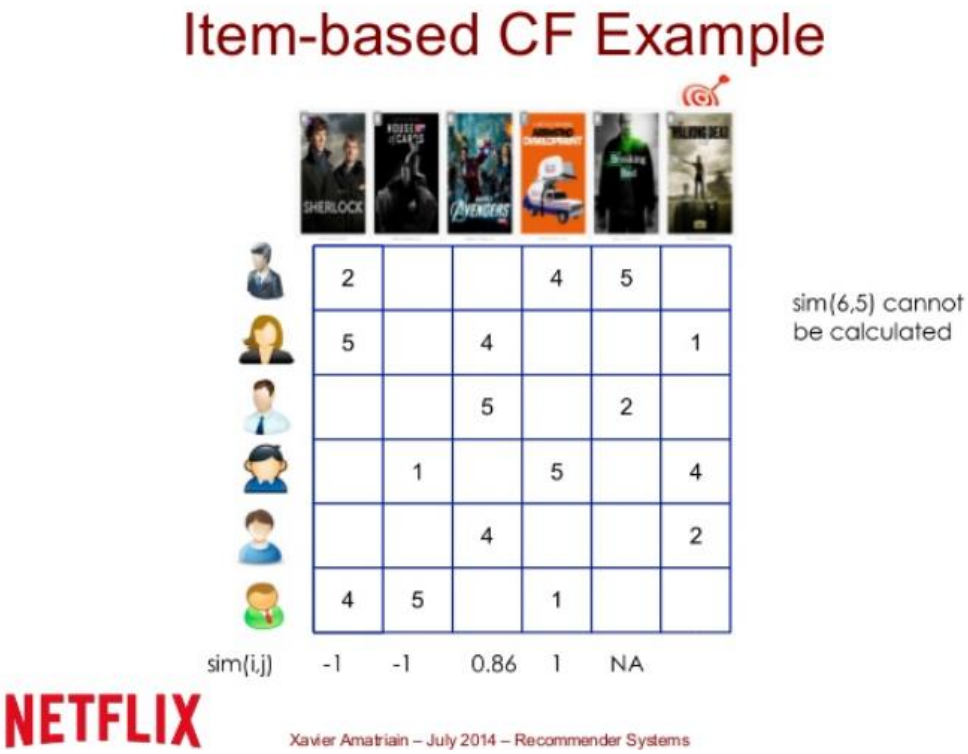
Motivation

Recommender systems are used in many online e-commerce sites.

For example, every time you browse the Amazon website, Amazon tracks you with your past purchase and browsing history, and make real-time recommendation of what you may be interested in.

Theory

- Collaborative
  - Filtering: Recommend items based only on user past behavior
    - User-based: Find similar users to me and recommend what they liked
    - Item-based: Find similar items to those that I have liked
- Content-based
  - Recommend based on item features
- Personalized learning to rank
  - Treat recommendations as a ranking problem
- Demographics
  - Recommend based on user features
- Social recommendations
  - Trust based
- Hybrid
  - Combination of the above



Source: Netflix

The Recommender model automatically predicts how a user will like and item. This is typically based on:

a. Historical/past behavior
b. Relationships to other users (friend, friend of friend)
c. Item similarity
d. etc

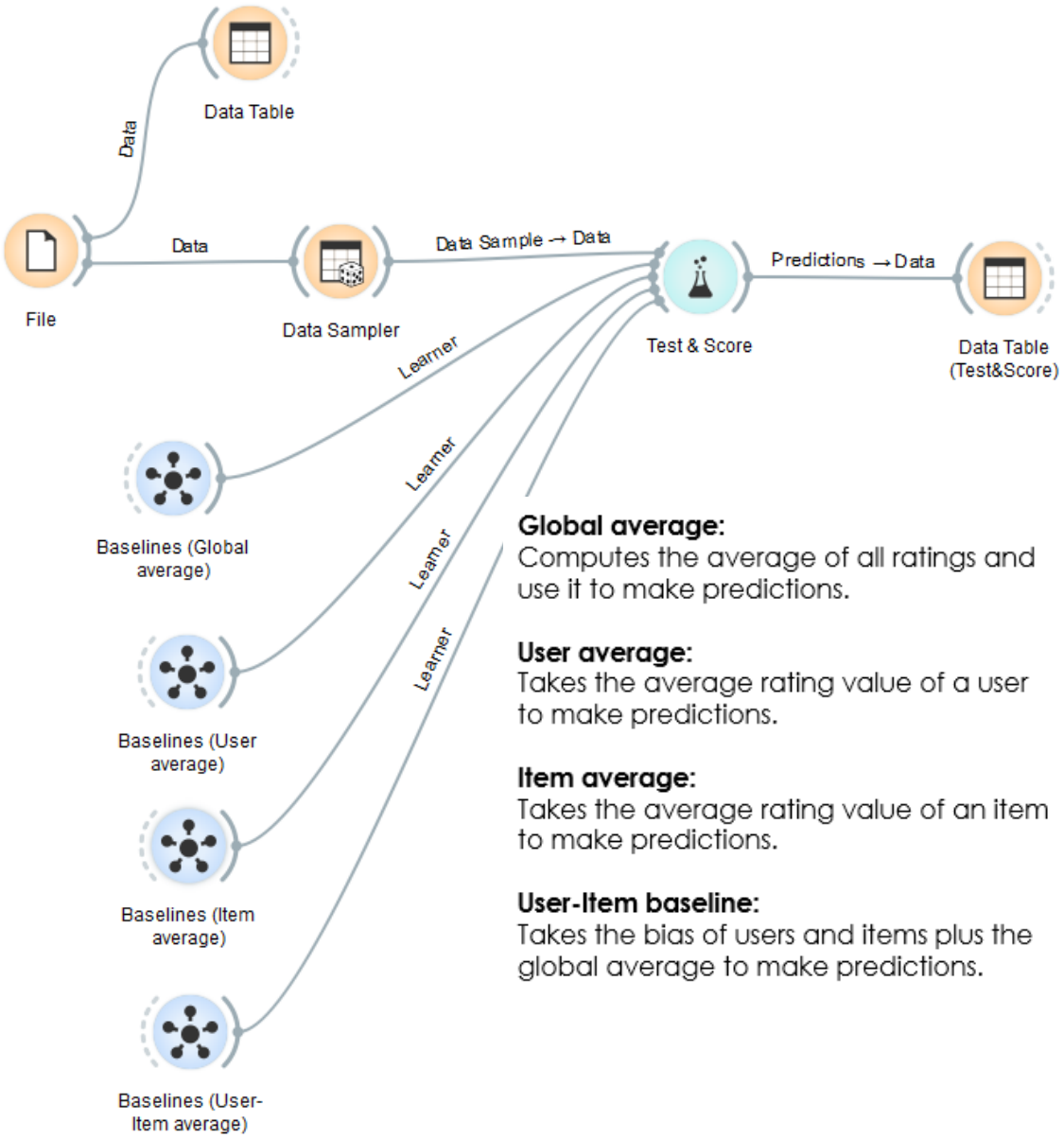
Workflow

1. Download and install the Recommender add-on.

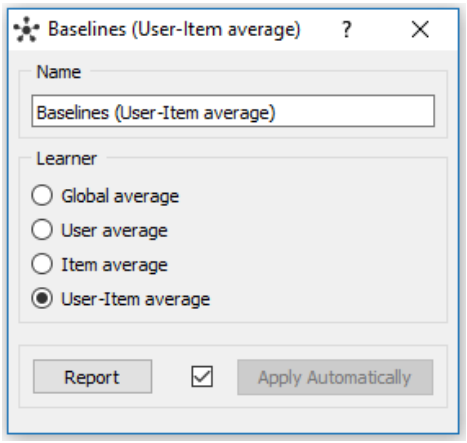
2. Download the movielens data:

   Download the movielens data from:
   http://orange3-recommendation.readthedocs.io/en/latest/_downloads/movielens100k.tab

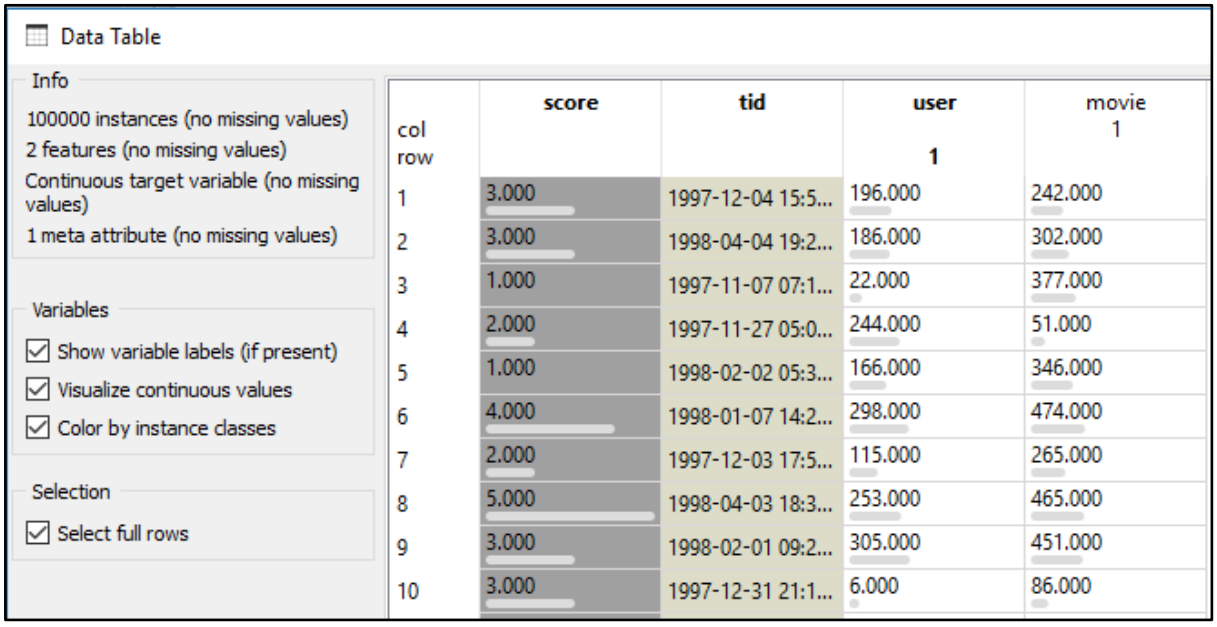
3. Build the workflow as shown. You may wish to rename the Baselines widget to match the learner selected.



Da

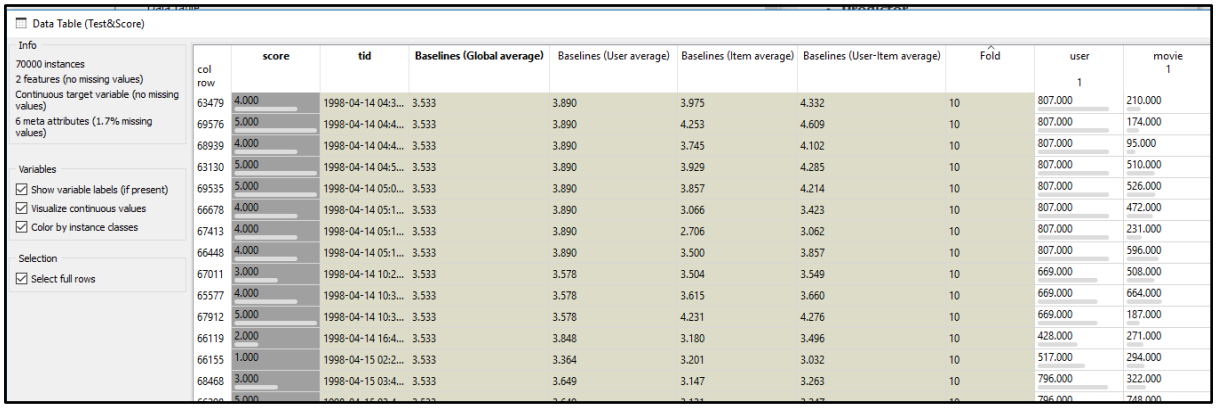


Movielens dataset

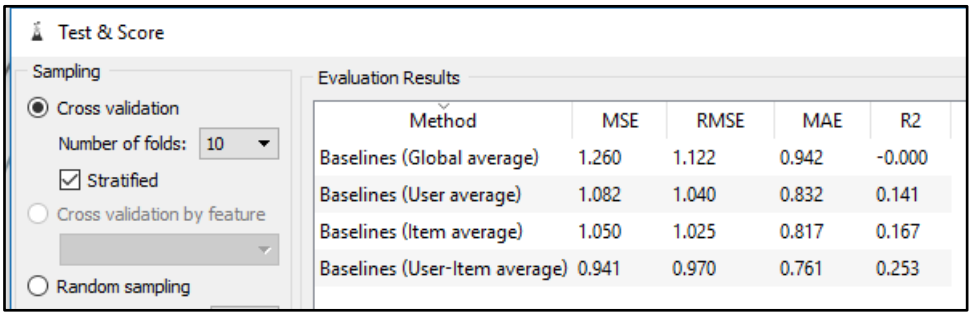


col / row	score	tid	user	movie 1
			1	
1	3.000	1997-12-04 15:5...	196.000	242.000
2	3.000	1998-04-04 19:2...	186.000	302.000
3	1.000	1997-11-07 07:1...	22.000	377.000
4	2.000	1997-11-27 05:0...	244.000	51.000
5	1.000	1998-02-02 05:3...	166.000	346.000
6	4.000	1998-01-07 14:2...	298.000	474.000
7	2.000	1997-12-03 17:5...	115.000	265.000
8	5.000	1998-04-03 18:3...	253.000	465.000
9	3.000	1998-02-01 09:2...	305.000	451.000
10	3.000	1997-12-31 21:1...	6.000	86.000

Data Table (Test & Score)



Test & Score Evaluation Results



As expected the User-Item baseline works the best. You may wish to try the other Recommender algorithms like BRISMF and SVD++ at home. It takes quite a fair bit longer to run on the movielens (70,000 records) dataset, so we are not going through this in class.

Conclusion

You have built a basic Recommender system to analyze the famous movielens database. The movielens dataset is well studied and the performance of the Recommender various algorithms are shown below:

Algorithm	RMSE	MAE	Train time
Global Average	1.126	0.945	0.001s
Item Average	1.000	0.799	0.001s
User Average	1.031	0.826	0.001s
User-Item Baseline	0.938	0.738	0.001s
BRISMF	0.810	0.642	2.027s/iter
SVD++	0.823	0.648	7.252s/iter

See that **Baselines** are 3 orders of magnitude faster than **BRISMF** and **SVD++** (0.001s vs 2s/iter and 7s/iter). Often you will want to use either a smaller dataset, or a faster algorithm while you get your workflow correct or data cleaned and formatted properly to speed up the analysis before committing to a big production dataset.

TEXT ANALYTICS

Motivation

Text analytics is a common request and includes:

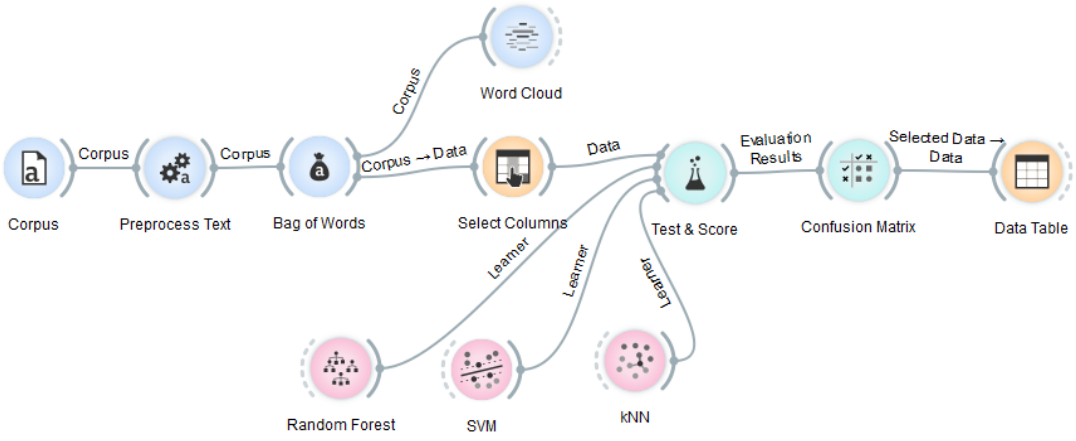
- Entities and places extraction
- Sentiment Analysis
- Similarity detection
- Topic modelling
- etc

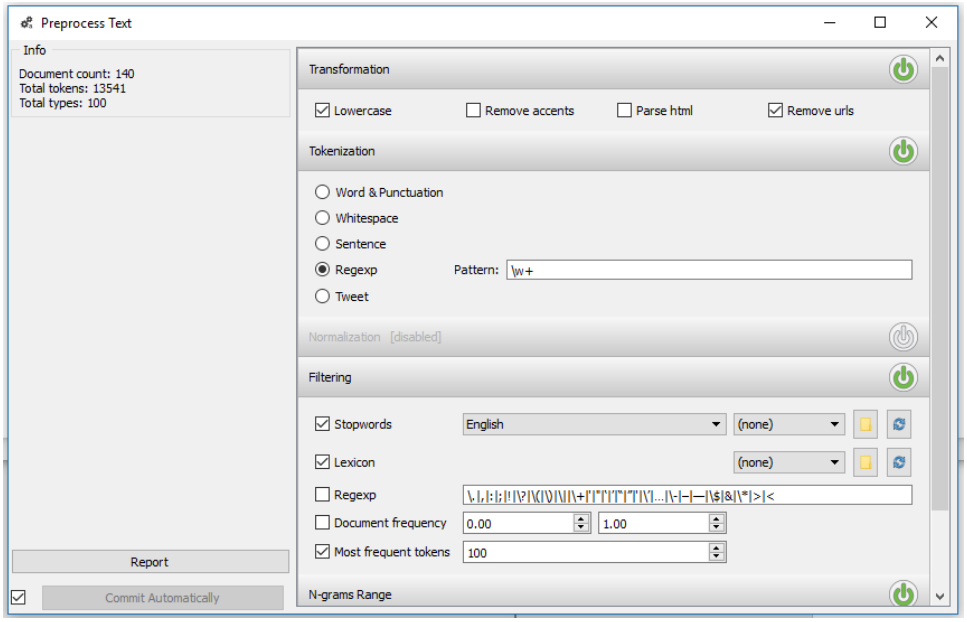
Theory

Text analytics or Natural Language Processing (NLP) is a widely research topic with many excellent and freely available text engines. The Orange add-on package Text Mining provides a basic suite of tools for text analytics with connectivity to popular online social media platforms like Twitter and NY Times.

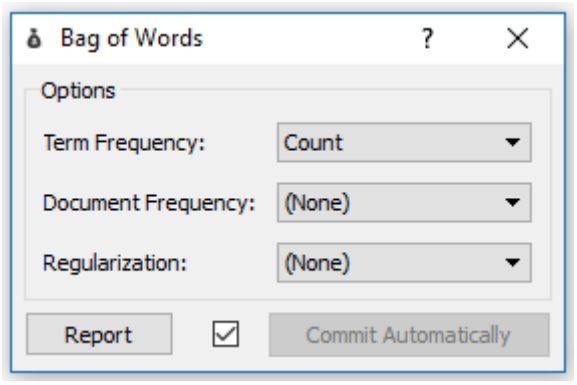
Workflow (Classification of documents with Bag of Words)

1. Build the following workflow to classify whether a book is a children's or adult's book. Use the bookexcerpts.tab data.

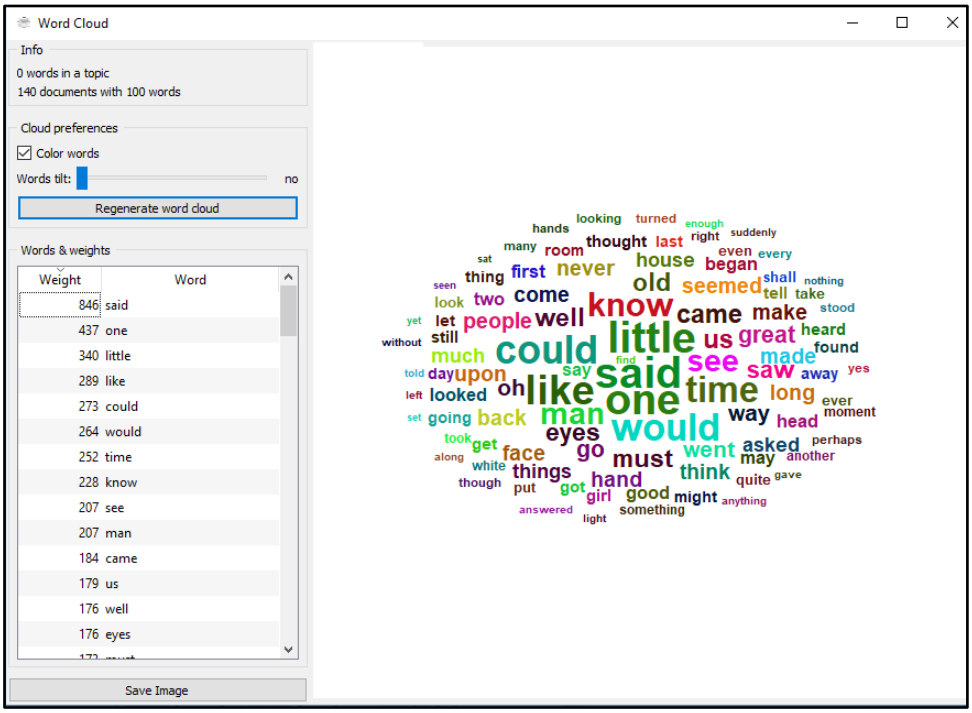




The **Preprocess Text** widget provides the typical capabilities to pre-process text such as removing stopwords, punctuations, stemming etc.



The **Bag of Words** widget allows you to choose the type of Term Frequency (Count, Binary, Sublinear) and other options.

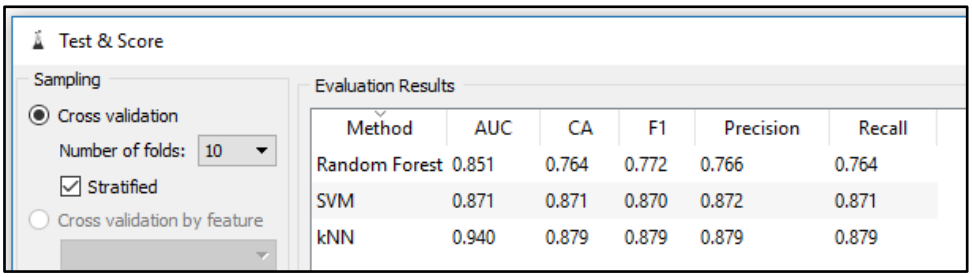


The **Word Cloud** widget provides the word cloud graphing function.

hidden skip-normalizati	category	text True	{...}
1	children	the house Jim says he rum ; and as he spoke he ...	man=1.717, anything=1.010, face=0.535, told=0.953, first=0.665, way=0.487, said=0.598,
2	children	has lived rough and I'll raise Cain Your doctor hi...	know=0.693, never=1.717, hands=0.953, could=0.250, day=1.597, ever=1.913, time=0.278
3	children	Now boy he said take me in to the captain Sir sa...	began=0.693, even=0.783, right=0.934, something=1.662, stood=1.905, perhaps=1.050, t
4	children	thanks to you big hulking chicken-hearted men...	without=1.869, set=1.050, looking=0.847, sat=1.030, turned=1.869, found=2.592, along=
5	children	the empty chest; and the next we had opened t...	answered=1.030, find=2.914, let=0.815, turned=0.934, found=0.864, along=0.916, might=
6	children	stood irresolute on the road You have your han...	every=0.953, find=0.971, set=1.050, turned=1.869, found=0.864, along=0.916, might=2.2
7	children	WE rode hard all the way till we drew up before ...	white=0.953, let=0.815, looking=0.847, sat=2.059, along=0.916, room=0.990, stood=0.953
8	children	same as the tattoo mark Billy Bones his fancy ; t...	every=0.953, find=0.971, found=1.728, might=0.737, right=1.869, something=0.831, perh
9	children	IT was longer than the squire imagined ere we ...	may=0.916, every=1.905, find=0.971, found=4.321, might=0.737, room=0.990, even=1.56

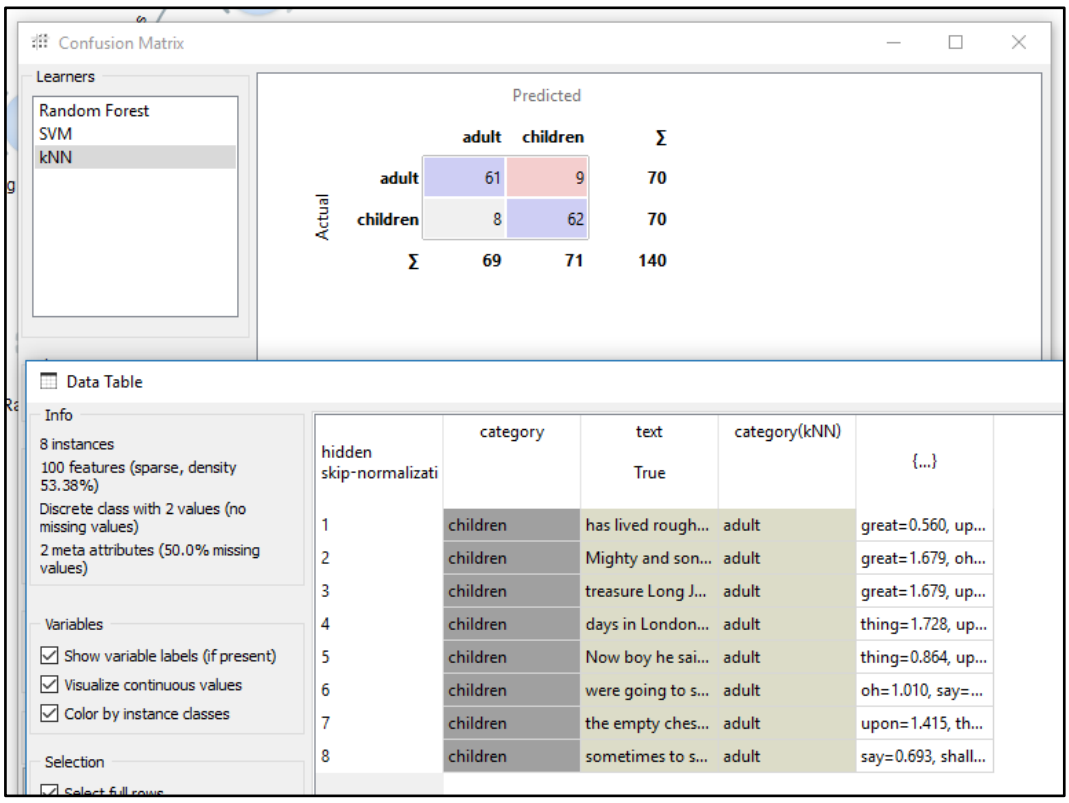
If you connect a Data Table to Bag of Words, you can see the outputs and the count values of each word in the document.

2. The **Select Columns** widget shows the Features generated after passing thru the Preprocess and Bag of Words widget. Using these Features and the Target Variable (category), we can now run the standard classification algorithms. Here we have chosen to use SVM, Random Forest and kNN.



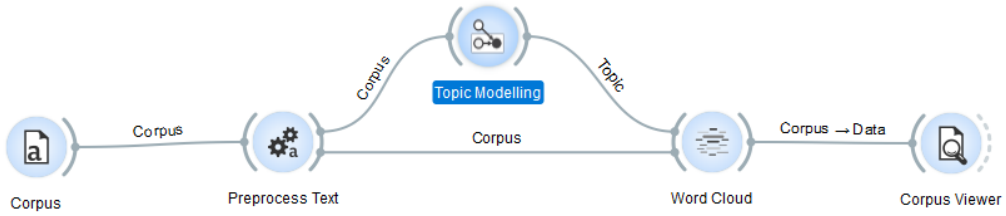
3. To summarize, to classify a book the steps are:
    a. preprocess the corpus and remove all stop words, punctuations
    b. use a bag of words to group/count all the words
    c. use Select Columns to transform the data into a format understandable by Test & Score and the various classification algorithms
    d. connect the classifier

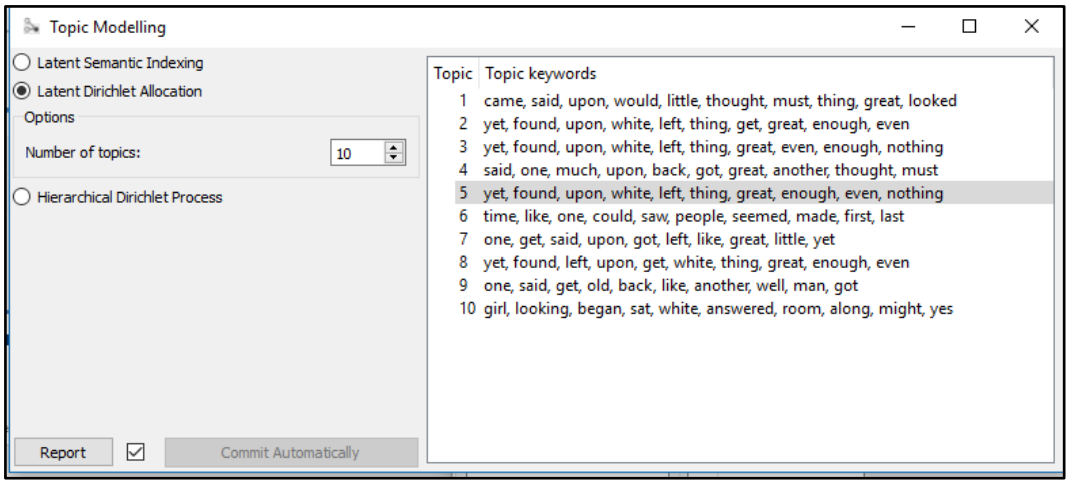
4. You can explore the misclassification and try to understand why the misclassification happened. Click on the cells in the Confusion Matrix, the Data Table will show the corresponding selection. This interactivity allows you to understand what went wrong in your workflow – whether it is a human mistake (data gathering phase) or you need to refine your preprocessing workflow further.



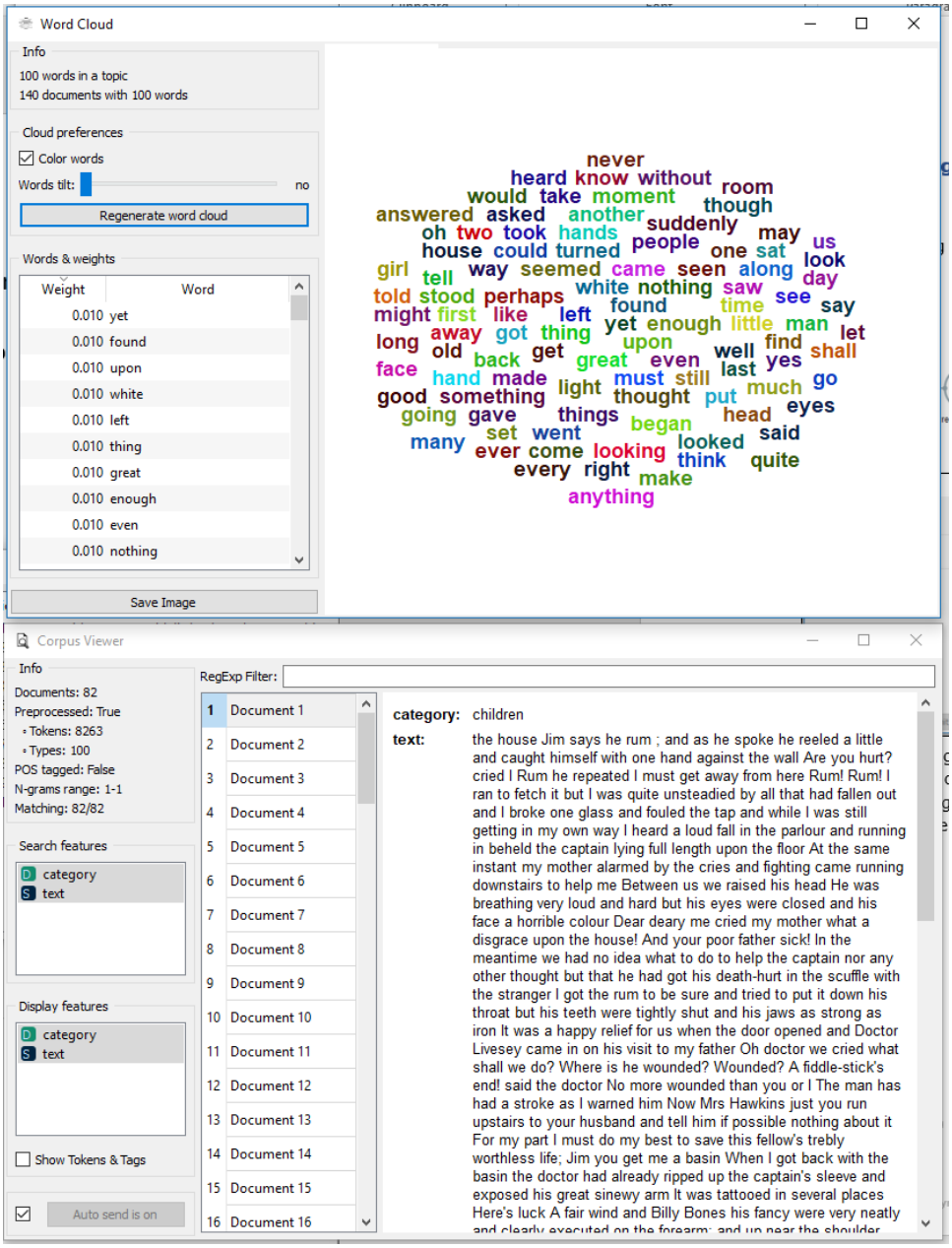
Workflow (Finding abstract ideas in documents with Topic Modelling)

1. Build the following workflow with the bookexcerpts.tab data.





The Topic Modelling widget provides the topics found in the document. You can limit the number of topics. The Topic Modeling widget supports both Latent Semantic Indexing and Latent Dirichlet Allocation (LDA). LDA is the most common and preferred method to extract topics from a corpus of text.

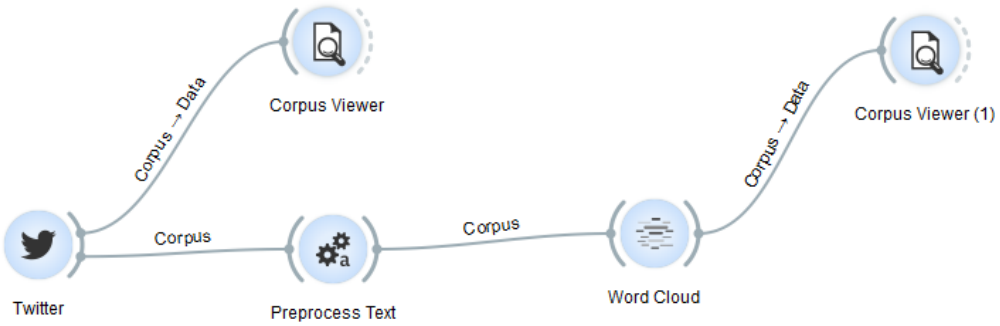


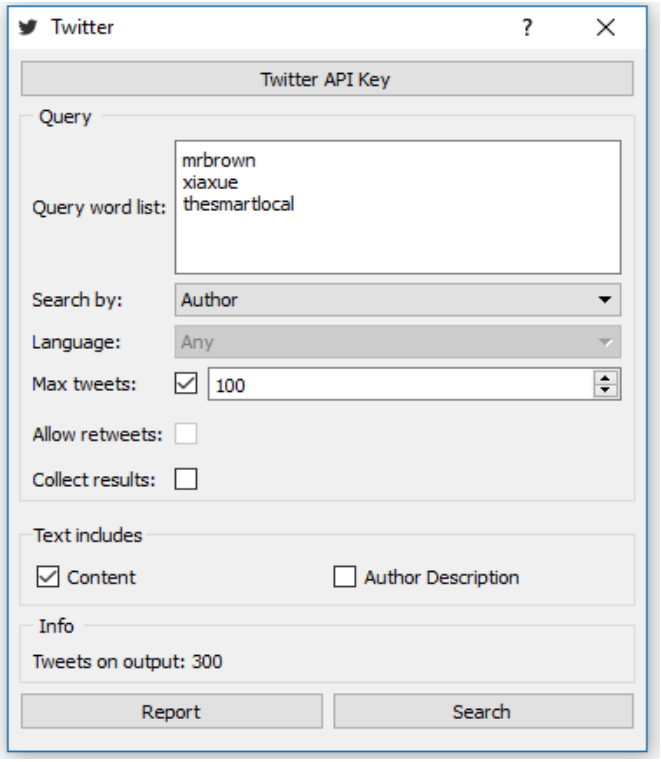
You can click on each word in the word cloud to retrieve the documents associated with the word.

Workflow (Analyzing Twitter feeds)

1. Refer to the slides to create a Twitter account for this exercise. It should not take you more than 5-10 minutes to create a Twitter account and Twitter App.

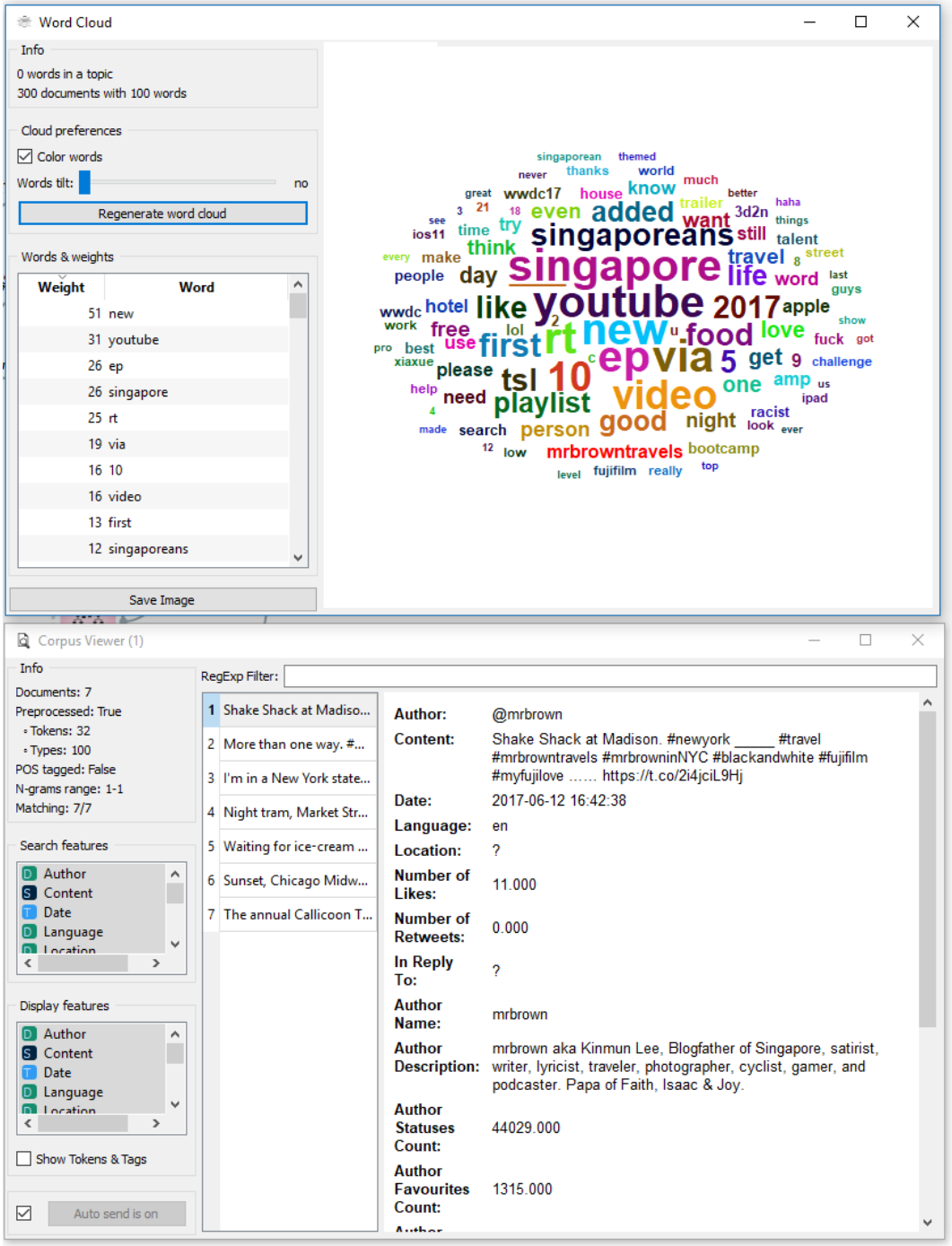
2. Build the following workflow.





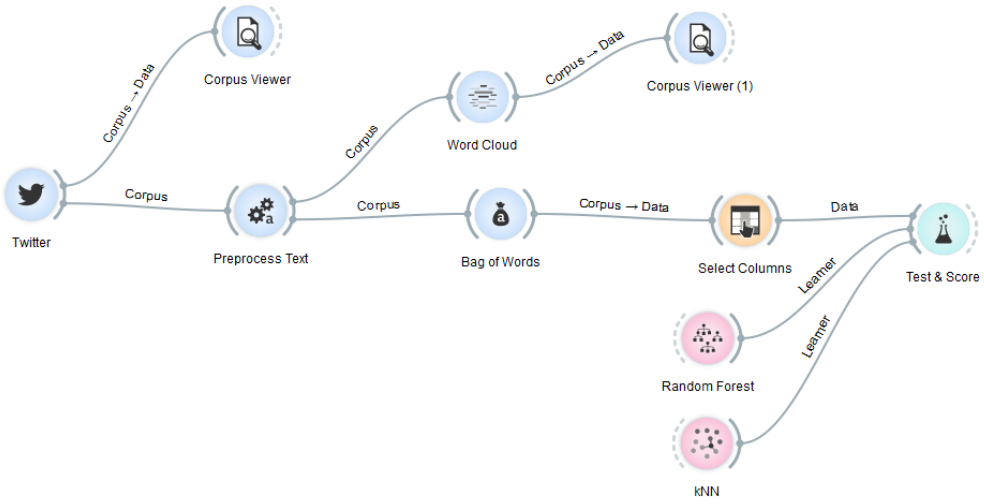
Enter your API key into the **Twitter** widget. Enter some interesting

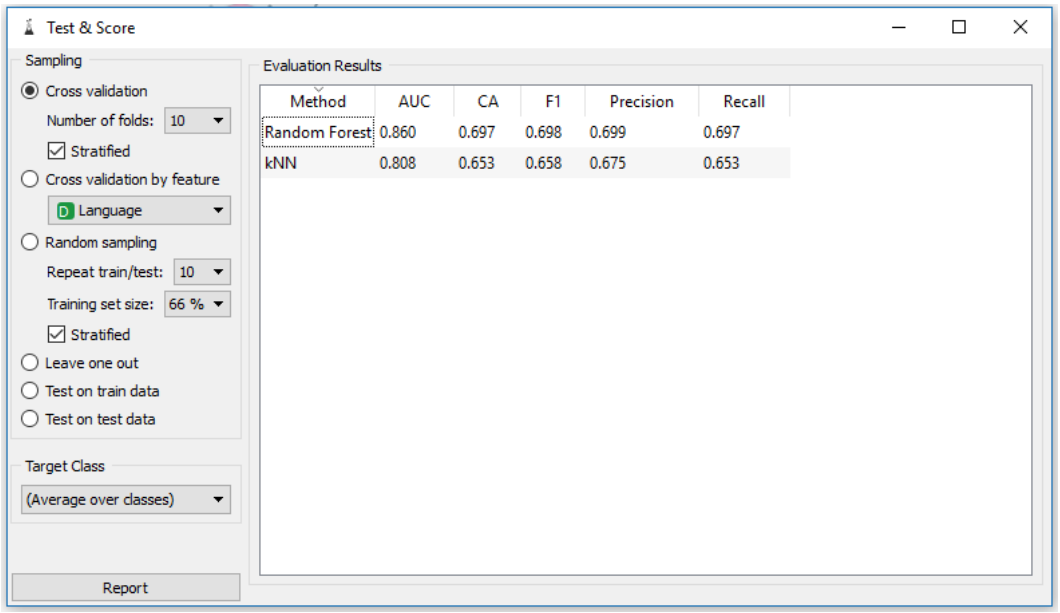
topics/words you want to search for on Twitter-sphere.



As before, the workflow setup allows you to interactively query the word cloud to bring up the associated Tweets.

3. As before we can train a classifier to classify the author of the tweet.

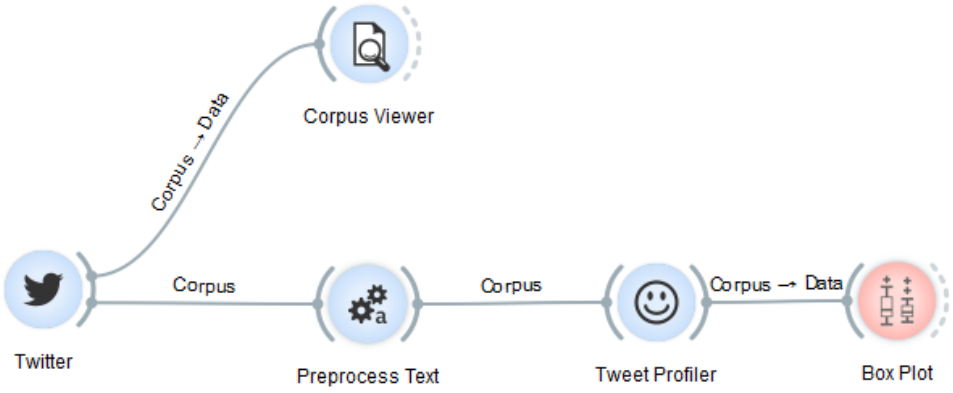


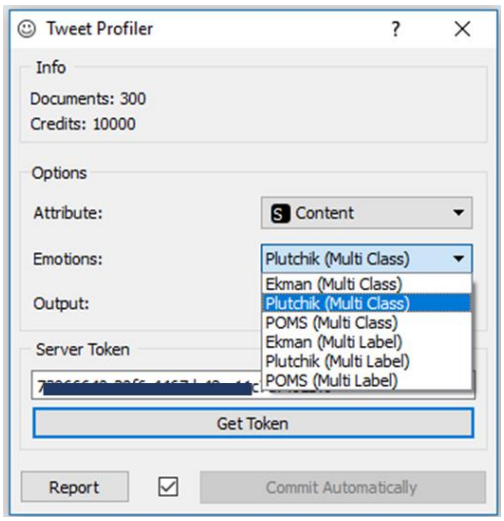


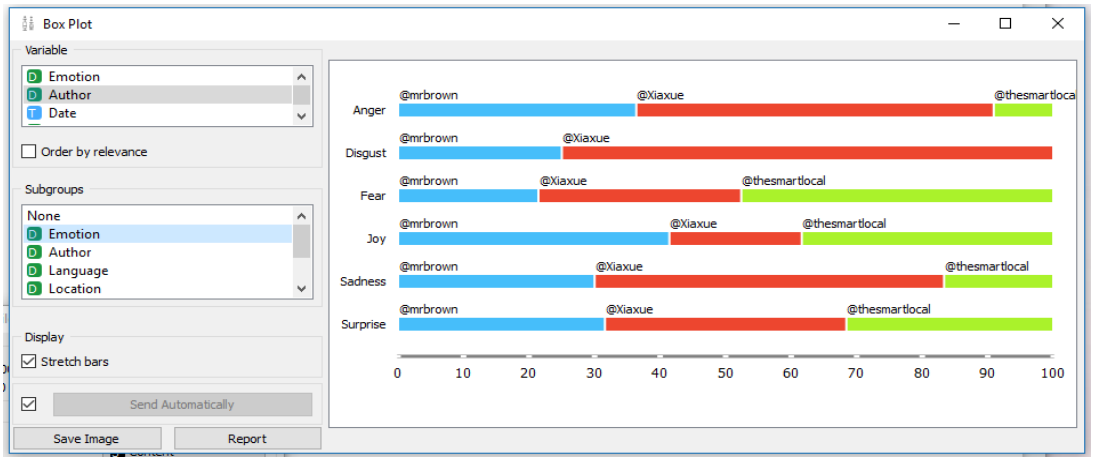
The above model could be saved, and then applied to new incoming tweets or text messages to determine who wrote the tweet/message.

Workflow (Sentiment Analysis)

1. Build the following workflow for sentiment analysis.







2. The **Tweet Profiler** widget supports different Emotions classification such as Ekman and Plutchik. For example, the Ekman classification have the following:
    a. Anger
    b. Disgust
    c. Fear
    d. Happiness
    e. Sadness
    f. Surprise

3. From the BoxPlot, we can see that @mrbrown tweets are is mostly Joy and not much Fear, while @Xiasue is Angry and Disgust.

Conclusion

In the last of the Analytics Dozen, you have built several text analytics pipelines, analyzing both local files and from a social media outlet like Twitter.

You have used the Bag of Words and Topic Modeling widget which are the commonly used methods to do basic to intermediate text processing and analysis.

Basic sentiment analysis was done using the Tweet Profiler.

Newer techniques like Deep Learning are not discussed however, but the research direction of text analytics is moving towards the use of deep learning.

EPILOGUE

This is the end of the Analytics Dozen (Orange edition).

You now have an idea of the most commonly used data analytics/science/machine-learning techniques the data scientist uses.

The purpose of this course is not to make you into a data scientist (it is not possible in 2 – 3 days), but to give you sufficient knowledge to identify hype from reality, and a common vocabulary in your conservation with your data science team or data scientist.

I wish you many years of happy learning!

-- End --