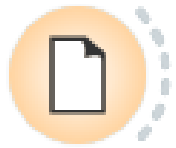


# Orange Tool

Dr. Hemraj S L.

# File Widget



File

- Reads attribute-value data from an input file.
- The File widget reads the input data file (data table with data instances) and sends the dataset to its output channel.
- The widget reads data from Excel (.xlsx), simple tab-delimited (.txt), comma-separated files (.csv) or URLs.

# CSV File Import

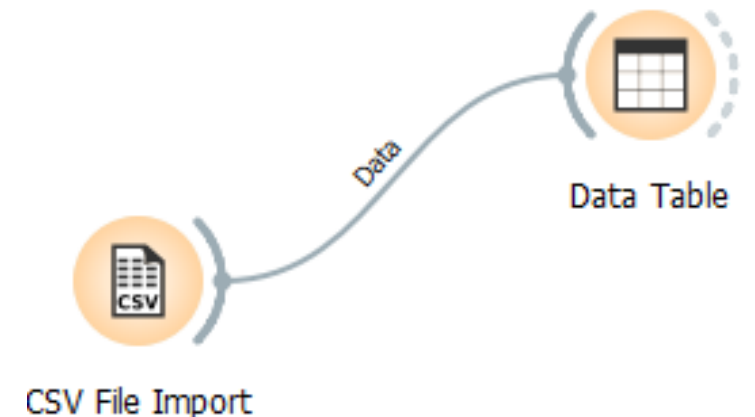


CSV File Import

- Import a data table from a CSV formatted file.
- The CSV File Import widget reads comma-separated files and sends the dataset to its output channel.

## Outputs

- Data: dataset from the .csv file



# Datasets



Datasets

- Datasets widget retrieves selected dataset from the server and sends it to the output. File is downloaded to the local memory and thus instantly available even without the internet connection. Each dataset is provided with a description and information on the data size, number of instances, number of variables, target and tags.

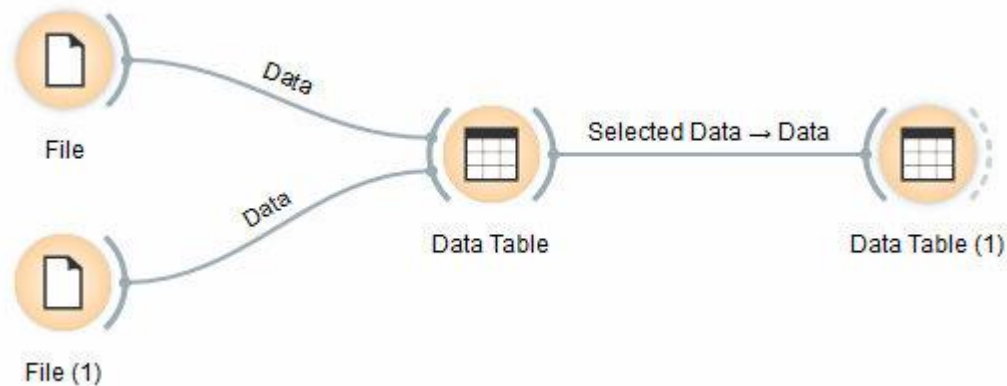


# Data Table



Data Table

- The Data Table widget receives one or more datasets in its input and presents them as a spreadsheet. Data instances may be sorted by attribute values. The widget also supports manual selection of data instances.

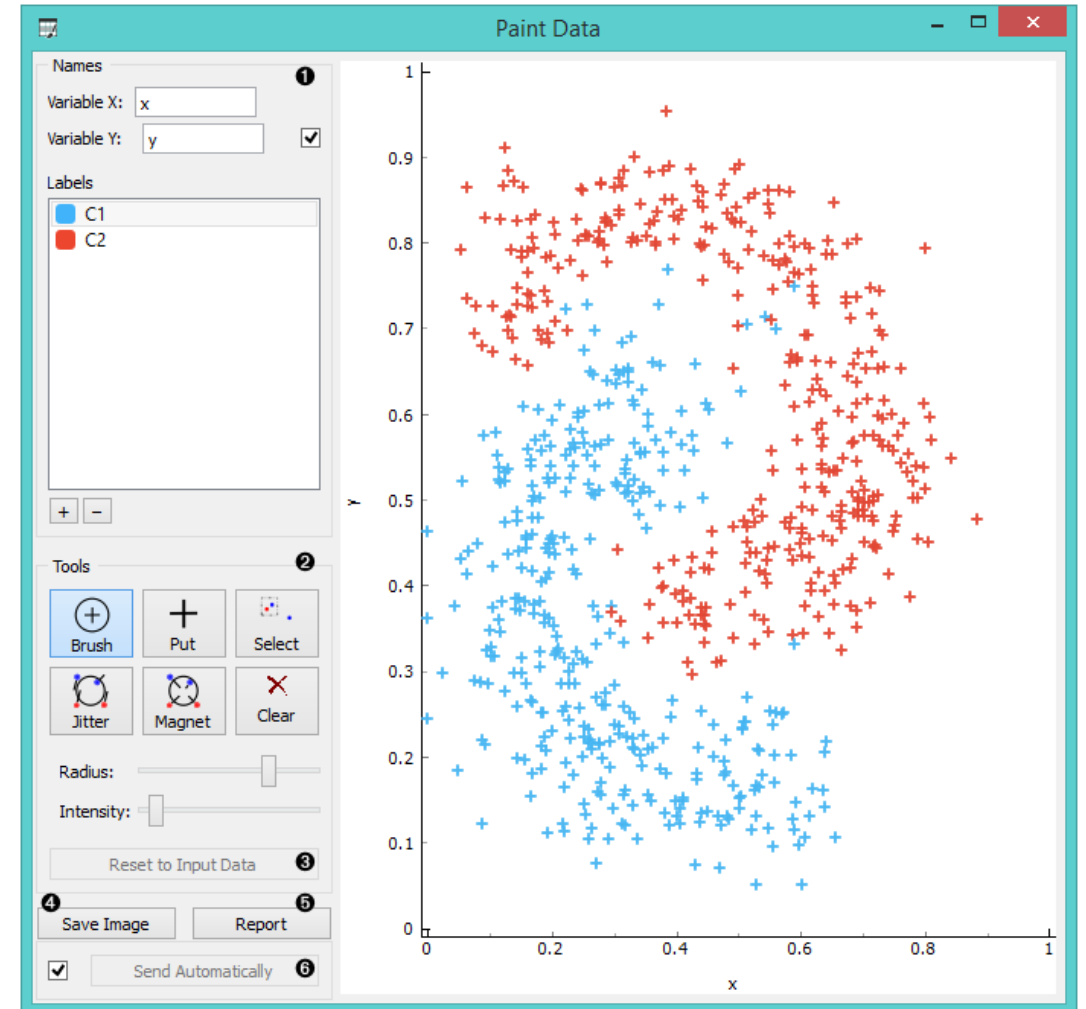


# Paint Data



Paint Data

- Paints data on a 2D plane. You can place individual data points or use a brush to paint larger datasets
- The widget supports the creation of a new dataset by visually placing data points on a two-dimension plane. Data points can be placed on the plane individually (Put) or in a larger number by brushing (Brush). Data points can belong to classes if the data is intended to be used in supervised learning.



# Data Info



- A simple widget that presents information on dataset size, features, targets, meta attributes, and location:
- Information on dataset size
- Information on discrete and continuous features
- Information on targets
- Information on meta attributes
- Information on where the data is stored
- Produce a report.



Data Info - ...

?

×

Data Set Name

traffic-signs

Data Set Size

Rows: 70  
Columns: 3

Features

None

Targets

Categorical outcome with 3 values

Meta Attributes

Categorical: -  
Numeric: -  
Text: 2

Location

Data is stored in memory

Data Attributes

?

|

→

70

# Aggregate Column



Aggregate Columns  
(1)

- Aggregate Columns outputs an aggregation of selected columns, for example a sum, min, max, etc.
- Operator for aggregation: sum, product, min, max, mean, variance, median



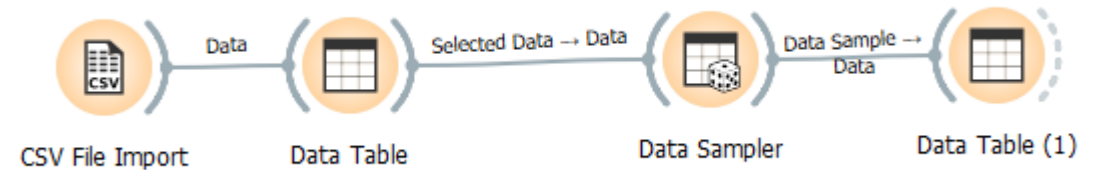


# Data Sampler



Data Sampler (1)

- The Data Sampler widget implements several data sampling methods. It outputs a sampled and a complementary dataset (with instances from the input set that are not included in the sampled dataset).
- The output is processed after the input dataset is provided and Sample Data is pressed.

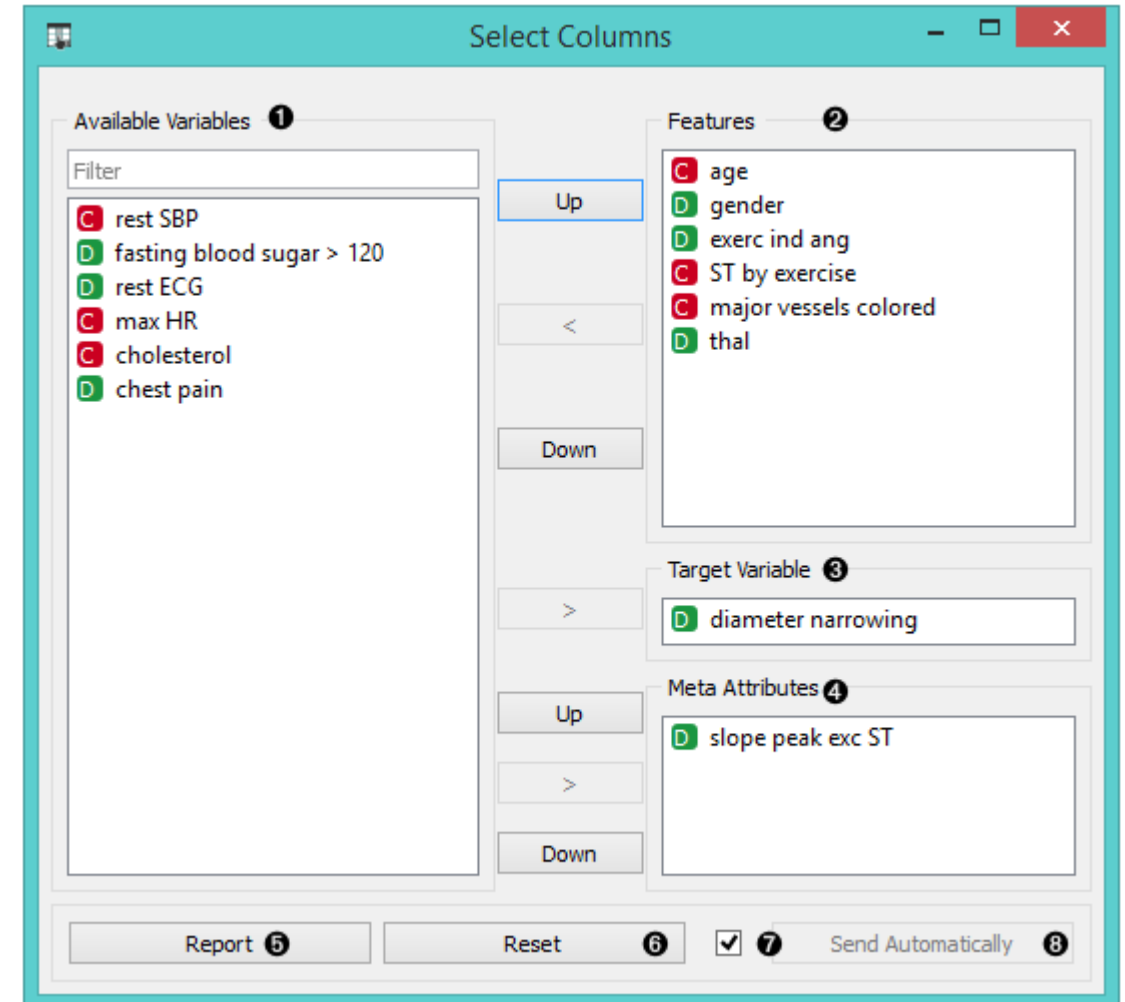


# Select Columns



Select Columns

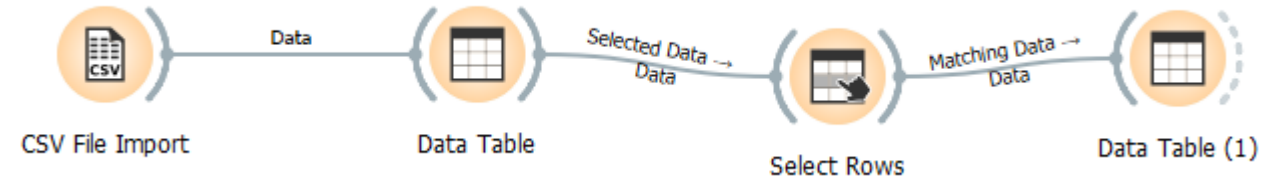
- The Select Columns widget is used to manually compose your data domain. The user can decide which attributes will be used and how. Orange distinguishes between ordinary attributes, (optional) class attributes and meta attributes



# Select Rows



Select Rows



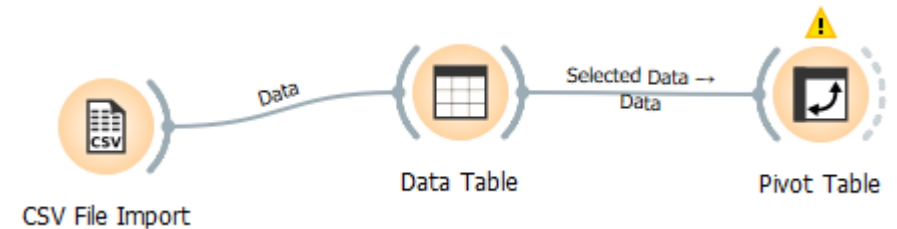
- Selects data instances based on conditions over data features.
- This widget selects a subset from an input dataset, based on user-defined conditions. Instances that match the selection rule are placed in the output Matching Data channel.

# Pivot Table

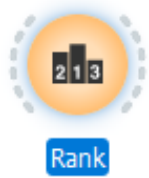


Pivot Table (1)

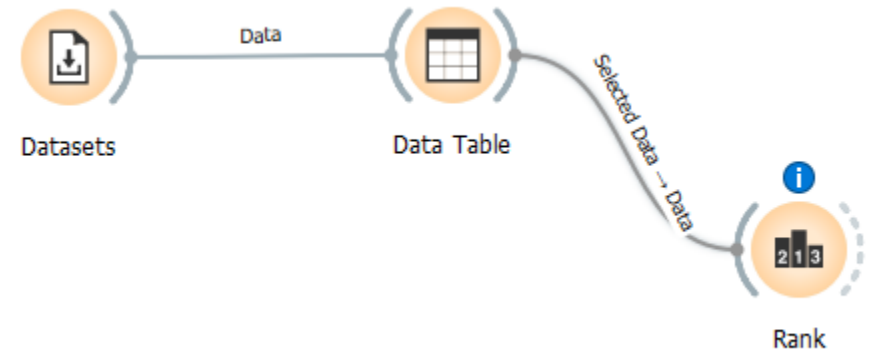
- Pivot Table summarizes the data of a more extensive table into a table of statistics. The statistics can include sums, averages, counts, etc. The widget also allows selecting a subset from the table and grouping by row values, which have to be a discrete variable. Data with only numeric variables cannot be displayed in the table.



# Rank



- The Rank widget scores variables according to their correlation with discrete or numeric target variable, based on applicable internal scorers (like information gain, chi-square and linear regression) and any connected external models that supports scoring, such as linear regression, logistic regression, random forest, etc.

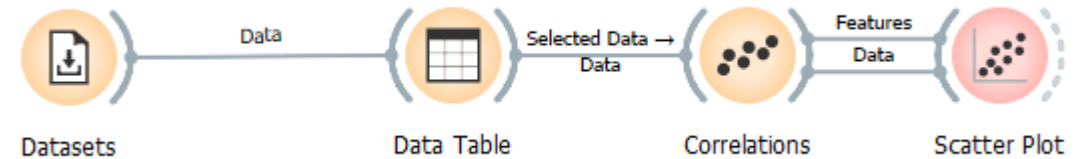


# Correlations



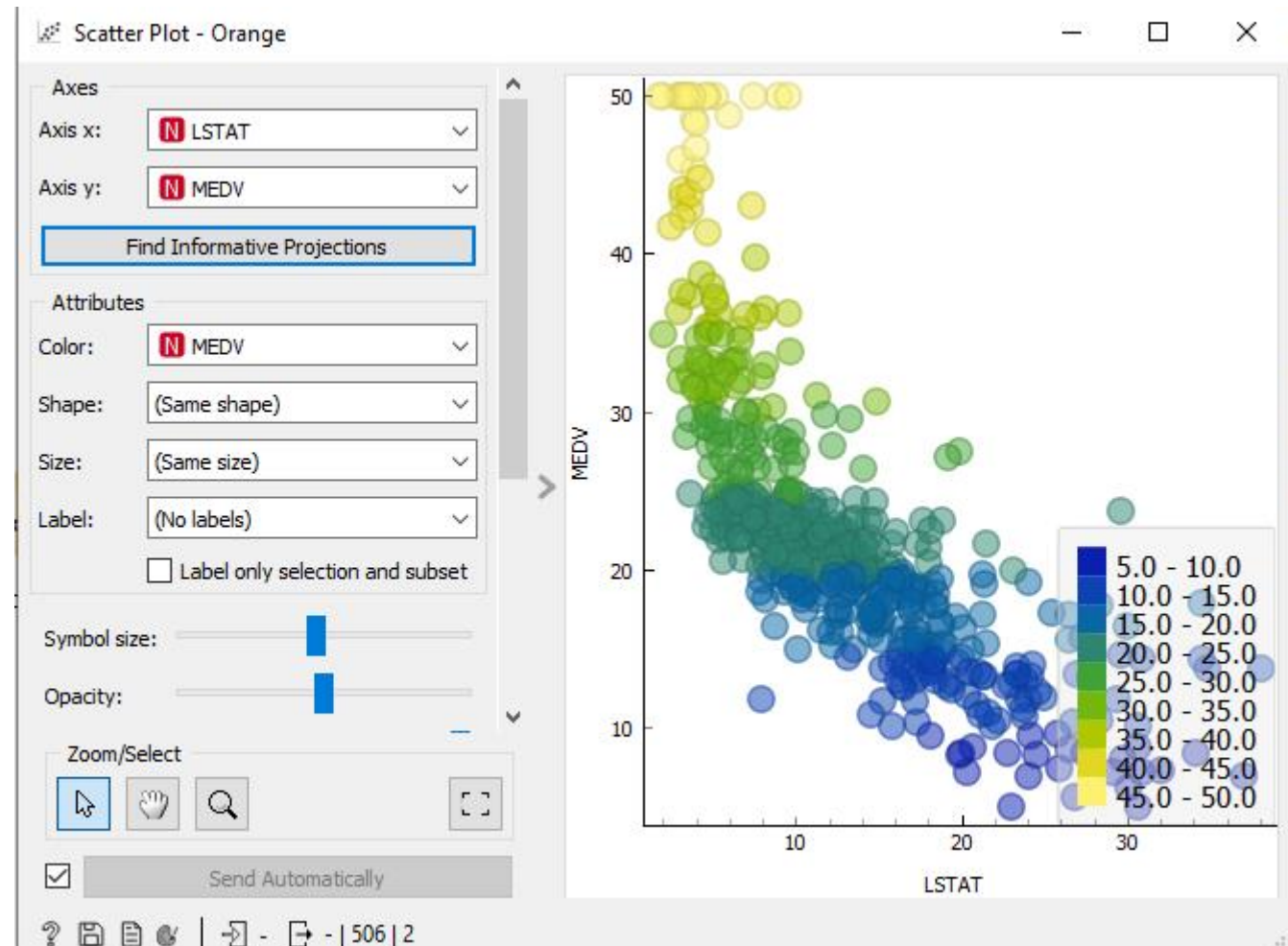
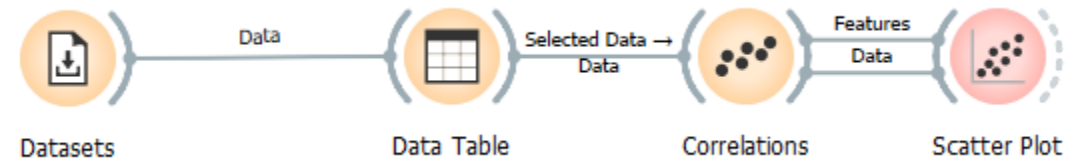
Correlations

- Correlations computes Pearson or Spearman correlation scores for all pairs of features in a dataset. These methods can only detect monotonic relationship.
- Correlations can be computed only for numeric (continuous) features, so we will use housing as an example data set. Load it in the File widget and connect it to Correlations. Positively correlated feature pairs will be at the top of the list and negatively correlated will be at the bottom.



# Scatter Plot

- Scatter plot visualization with exploratory analysis and intelligent data visualization enhancements.
- Selected Data: instances selected from the plot
- Data: data with an additional column showing whether a point is selected.
- The Scatter Plot widget provides a 2-dimensional scatter plot visualization. The data is displayed as a collection of points, each having the value of the x-axis attribute determining the position on the horizontal axis and the value of the y-axis attribute determining the position on the vertical axis

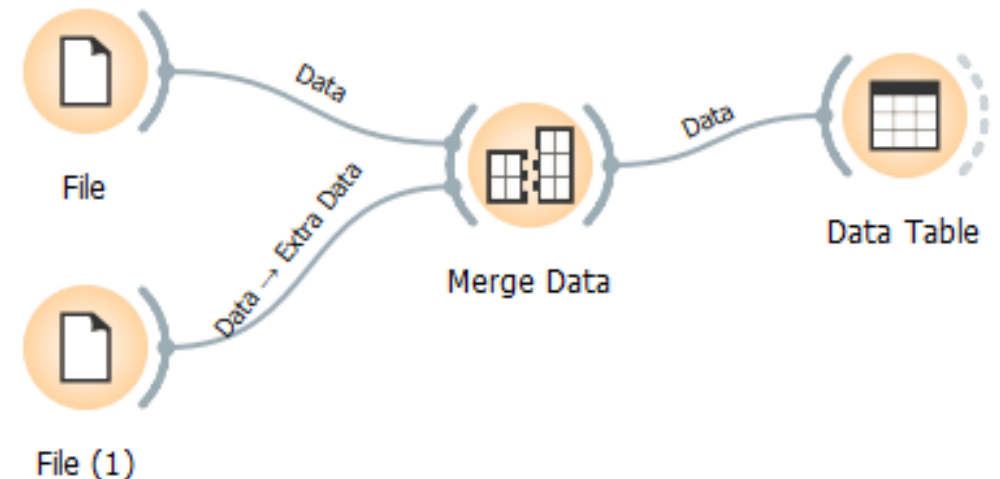


# Merge Data



Merge Data (1)

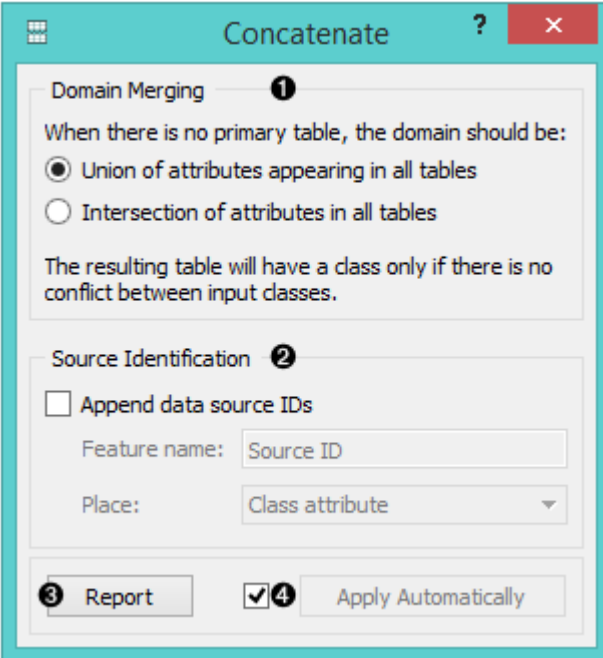
- The Merge Data widget is used to horizontally merge two datasets, based on the values of selected attributes (columns). In the input, two datasets are required, data and extra data. Rows from the two data sets are matched by the values of pairs of attributes, chosen by the user. The widget produces one output. It corresponds to the instances from the input data to which attributes (columns) from input extra data are appended.
- If the selected attribute pair does not contain unique values (in other words, the attributes have duplicate values), the widget will give a warning.





# Concatenate Data

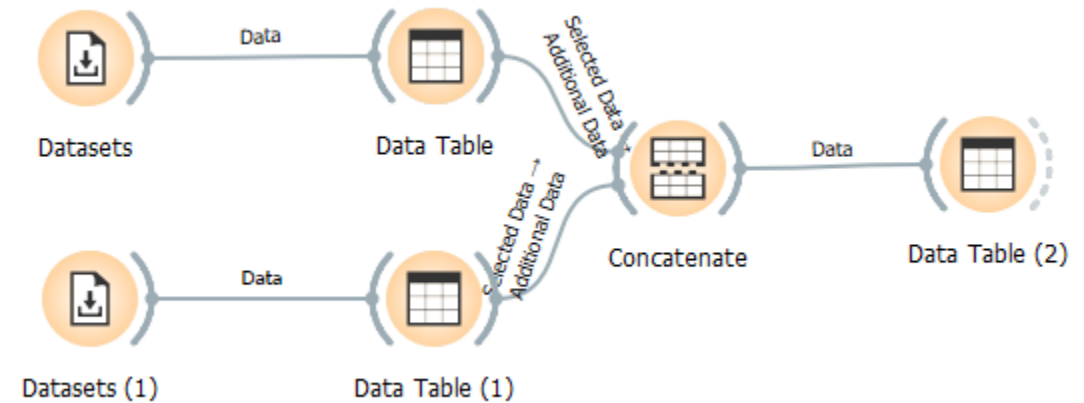
- The widget concatenates multiple sets of instances (data sets). The merge is “vertical”, in a sense that two sets of 10 and 5 instances yield a new set of 15 instances.
- Set the attribute merging method.
- Add the identification of source data sets to the output data set.
- Produce a report.
- If Apply automatically is ticked, changes are communicated automatically. Otherwise, click Apply.



The screenshot shows the 'Concatenate' widget configuration window. It has a title bar with a question mark and a close button. The window is divided into two main sections: 'Domain Merging' and 'Source Identification'. In the 'Domain Merging' section, there is a text label 'When there is no primary table, the domain should be:' followed by two radio buttons: 'Union of attributes appearing in all tables' (which is selected) and 'Intersection of attributes in all tables'. Below this, a text label states 'The resulting table will have a class only if there is no conflict between input classes.' The 'Source Identification' section contains a checkbox 'Append data source IDs' which is unchecked. Below it, there are two input fields: 'Feature name:' with the text 'Source ID' and 'Place:' with a dropdown menu showing 'Class attribute'. At the bottom of the window, there are two buttons: 'Report' and 'Apply Automatically'. The 'Apply Automatically' button has a checkmark icon and a circled number '4' next to it. There are also circled numbers '1' and '2' next to the 'Domain Merging' and 'Source Identification' section headers respectively, and a circled number '3' next to the 'Report' button.

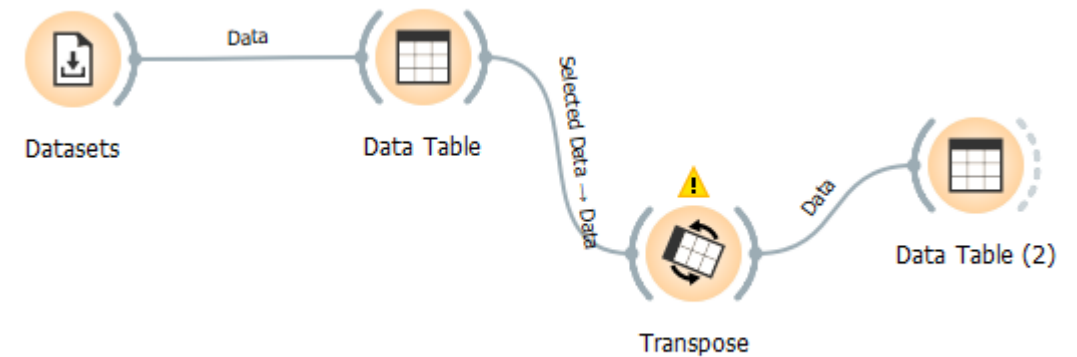
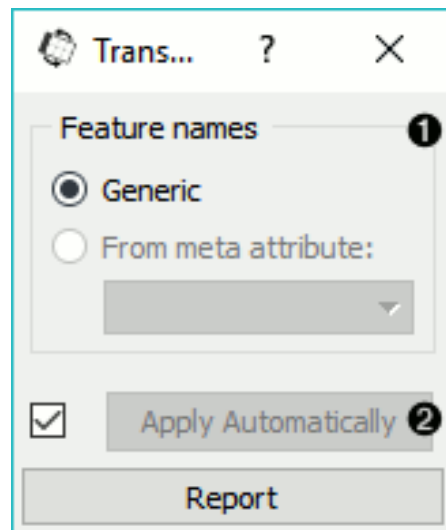
# Concatenate Data

- The widget concatenates multiple sets of instances (data sets). The merge is “vertical”, in a sense that two sets of 10 and 5 instances yield a new set of 15 instances.
- Set the attribute merging method.
- Add the identification of source data sets to the output data set.
- Produce a report.
- If Apply automatically is ticked, changes are communicated automatically. Otherwise, click Apply.



# Transpose

- The output of Transpose is a transposed data table with rows as columns and columns as rows



# Regression

- Regression models describe the relationship between variables by fitting a line to the observed data. Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line.
- Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.
- Simple linear regression is used to estimate the relationship between two quantitative variables.
- There are two types of **quantitative** variables: **discrete** (Number of students in CSIT) and **continuous** (Age, Distance, Height, Volume, etc).
- There are three types of categorical variables: **binary** (*Head/Tail*) **nominal** (*Colors, Brands*), and **ordinal** (*Finishing place in MotoGP race, Rating scale response in survey*) variables.

# Example

This example sheet is color-coded according to the type of variable: **nominal**, **continuous**, **ordinal**, and **binary**.

Data Sheet// Salt Tolerance// Date: \_\_\_\_\_

Sample #	Plant Species	Salt Added (mg/L water)	Starting Height (cm)	Growth (cm) (current height – starting height)	Wilting (rank 0- 10)	Survival (1=survived, 0=died)
1	A	0	12			
2	A	100	13			
3	A	250	11			
4	B	0	25			
5	B	100	26			
6	B	250	25			

# Linear Regression

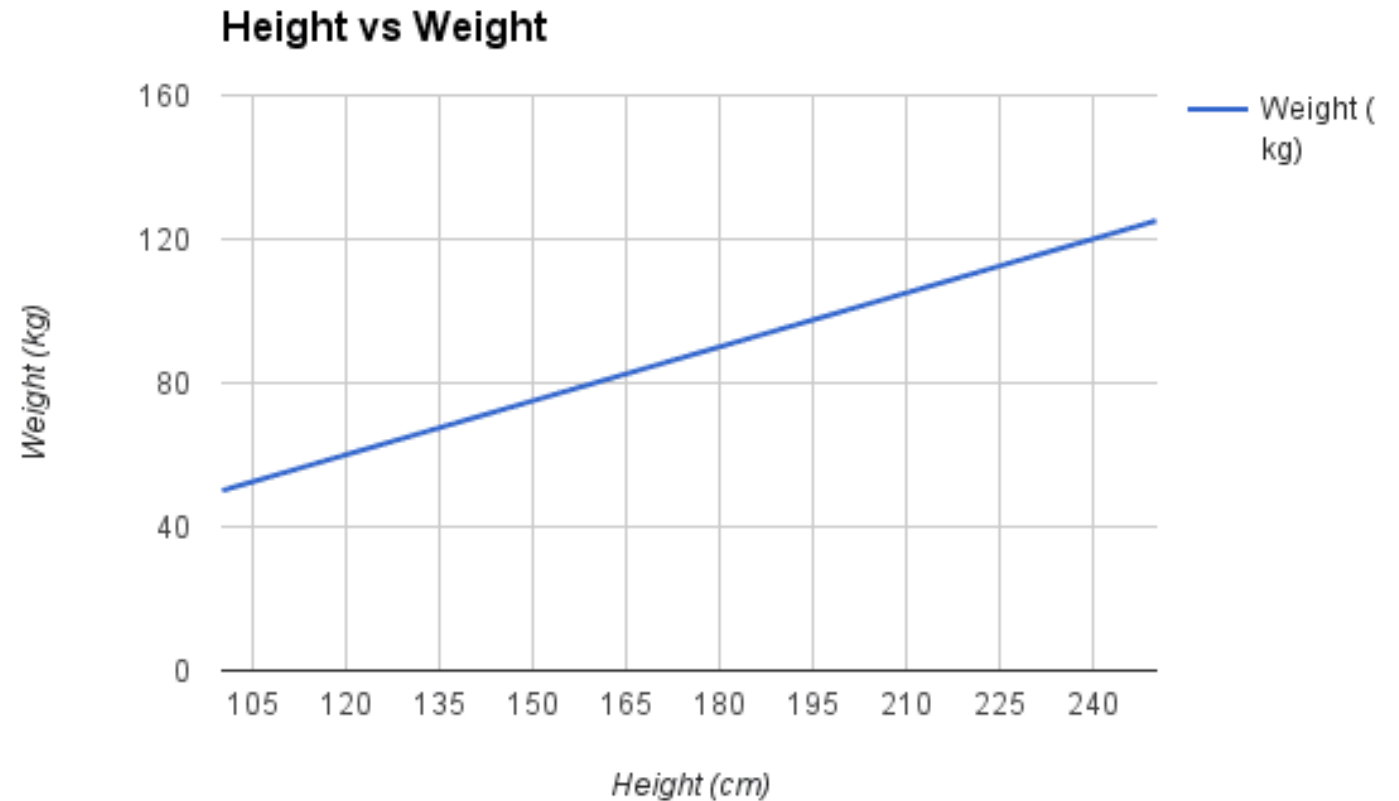
- Simple linear regression is used to estimate the relationship between two quantitative variables
- One variable, denoted  $x$ , is regarded as the **predictor**, **explanatory**, or **independent** variable.
- The other variable, denoted  $y$ , is regarded as the **response**, **outcome**, or **dependent** variable
- You can use simple **linear regression** when you want to know:
  - 1. How strong the relationship is between two variables (e.g. the relationship between rainfall and soil erosion).
  - 2. The value of the dependent variable at a certain value of the independent variable (e.g. the amount of soil erosion at a certain level of rainfall).

We manipulate the **Independent variable** (the one you think might be the **cause**) and then measure the **dependent variable** (the one you think might be the **effect**) to find out what this effect might be.

We probably also have variables that you hold constant (**control variables**) in order to focus on our experiment.

# Example

**For example**, someone's height and weight usually have a relationship. Generally, taller people tend to weigh more. We could use regression analysis to help predict the weight of an individual, given their height.



Here we have one **Independent**, and three **Dependent** variables.

The other variables in the sheet can't be classified as independent or dependent, but they do contain data that you will need in order to interpret your dependent and independent variables.

Data Sheet// Salt Tolerance// Date: \_\_\_\_\_

Sample #	Plant Species	Salt Added (mg/L water)	Starting Height (cm)	Growth (cm) (current height – starting height)	Wilting (rank 0- 10)	Survival (1=survived, 0=died)
1	A	0	12			
2	A	100	13			
3	A	250	11			
4	B	0	25			
5	B	100	26			
6	B	250	25			

**Note:** the terms “dependent” and “independent” don’t apply in Correlations, because we are not trying to establish a cause and effect relationship



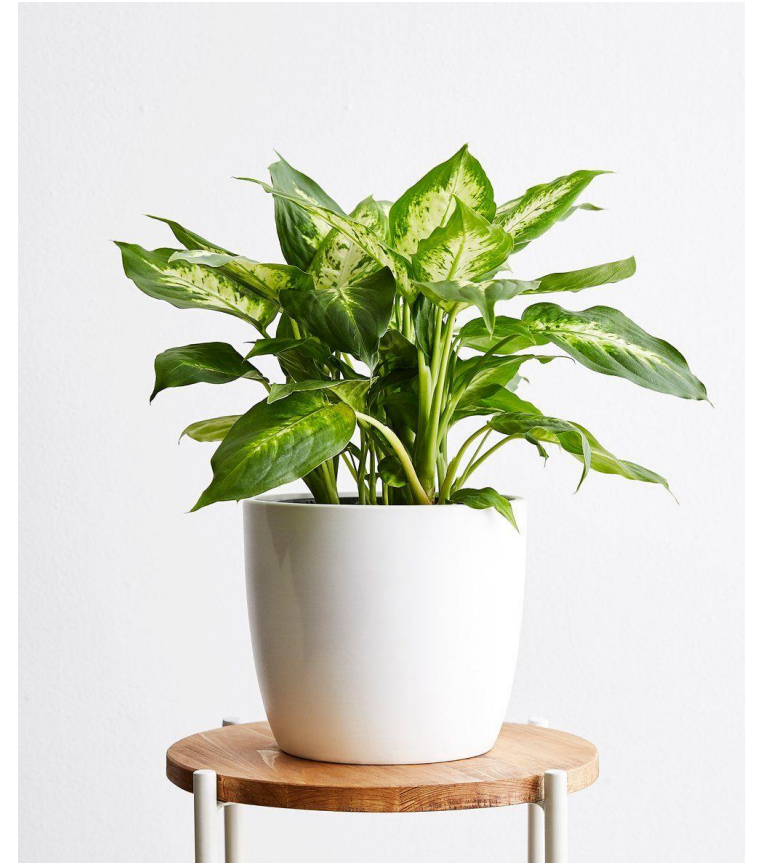
# Variables

- **Independent Variable:** The amount of salt added to each plant's water.
- **Dependent Variable:** Any measurement of plant health and growth: plant height.
- **Other Variable:** The temperature and light in the room the plants are kept in, and the volume of water given to each plant.



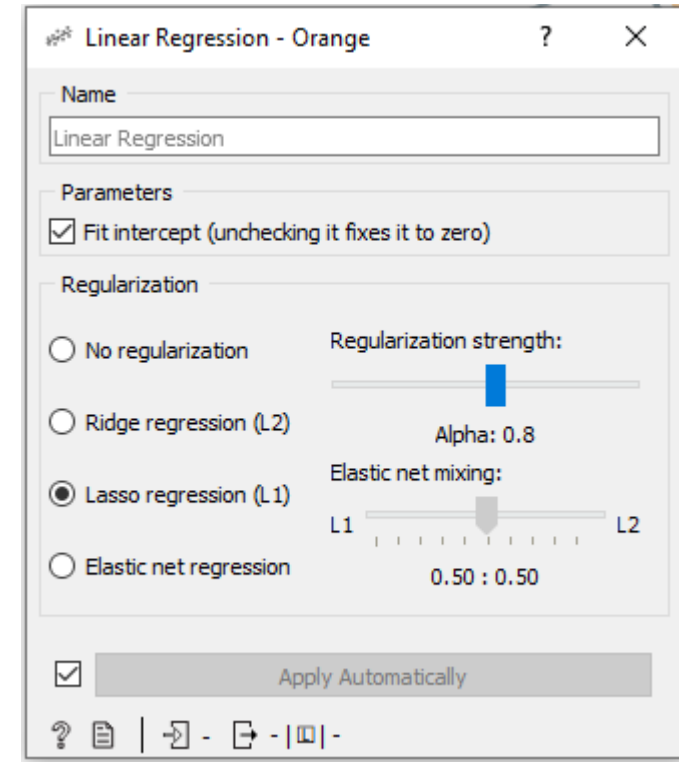
# Other Variables

- **Confounding Variable:** A variable that hides the true effect of another variable in your experiment. *Ex: Pot size and soil type might affect plant survival as much or more than salt additions*
- **Latent Variable:** A variable that can't be directly measured, but that you represent via a proxy. *Ex: Salt tolerance in plants cannot be measured directly, but can be inferred from measurements of plant health.*
- **Composite Variable:** A variable that is made by combining multiple variables. *Ex: Plant health score depends on multiple variables.*



# Linear Regression

- A linear regression algorithm with optional L1 (LASSO), L2 (ridge) or L1L2 (elastic net) regularization.
- **Inputs**
  - Data: input dataset
  - Preprocessor: pre-processing method(s)
- **Outputs**
  - Learner: linear regression learning algorithm
  - Model: trained model
  - Coefficients: linear regression coefficients



# Lasso bound

- In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

# Ridge Regularization

- Regularization is one of the ways to improve our model to work on unseen data by ignoring the less important features.
- Regularization minimizes the validation loss and tries to improve the accuracy of the model.
- It avoids overfitting by adding a penalty to the model with high variance, thereby shrinking the beta coefficients to zero.

# Preprocessing in Linear Regression

Linear Regression uses default preprocessing when no other preprocessors are given. It executes them in the following order:

- removes instances with unknown target values
- continues categorical variables (with one-hot-encoding)
- removes empty columns
- imputes missing values with mean values

To remove default preprocessing, connect an empty Preprocess widget to the learner.

# Parameters used in Linear Regression

- **Mean Squared Error (MSE):** The mean squared error (MSE) tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences. It’s called the mean squared error as you’re finding the average of a set of errors. The lower the MSE, the better the forecast.
- **Root mean squared error (RMSE)**
- The most common metric for evaluating linear regression model performance is called root mean squared error, or RMSE. The basic idea is to measure how bad/erroneous the model’s predictions are when compared to actual observed values. So a high RMSE is “**bad**” and a low RMSE is “**good**”.

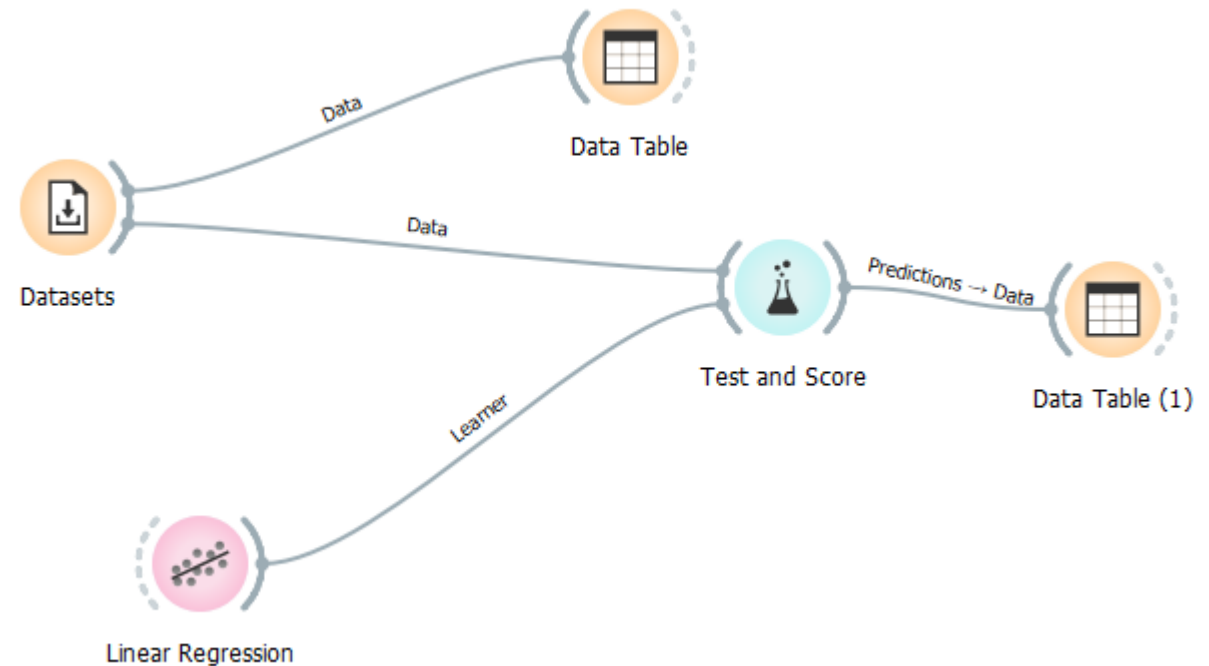
# Parameters used in Linear Regression

- **Mean Absolute Error (MAE):** The Mean absolute error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.
- **R-Squared:** The coefficient of determination or R-squared represents the proportion of the variance in the dependent variable which is explained by the linear regression model. It is a scale-free score i.e. irrespective of the values being small or large, the value of R square will be less than one.



# Linear Regression

The Linear Regression widget constructs a learner/predictor that learns a linear function from its input data. The model can identify the relationship between a predictor  $x$  and the response variable  $y$ .



# Assumption in Linear Regression

- Linearity
- Normality
- Multiple Linear Regression
- Homoscedasticity
- No Autocorrelation

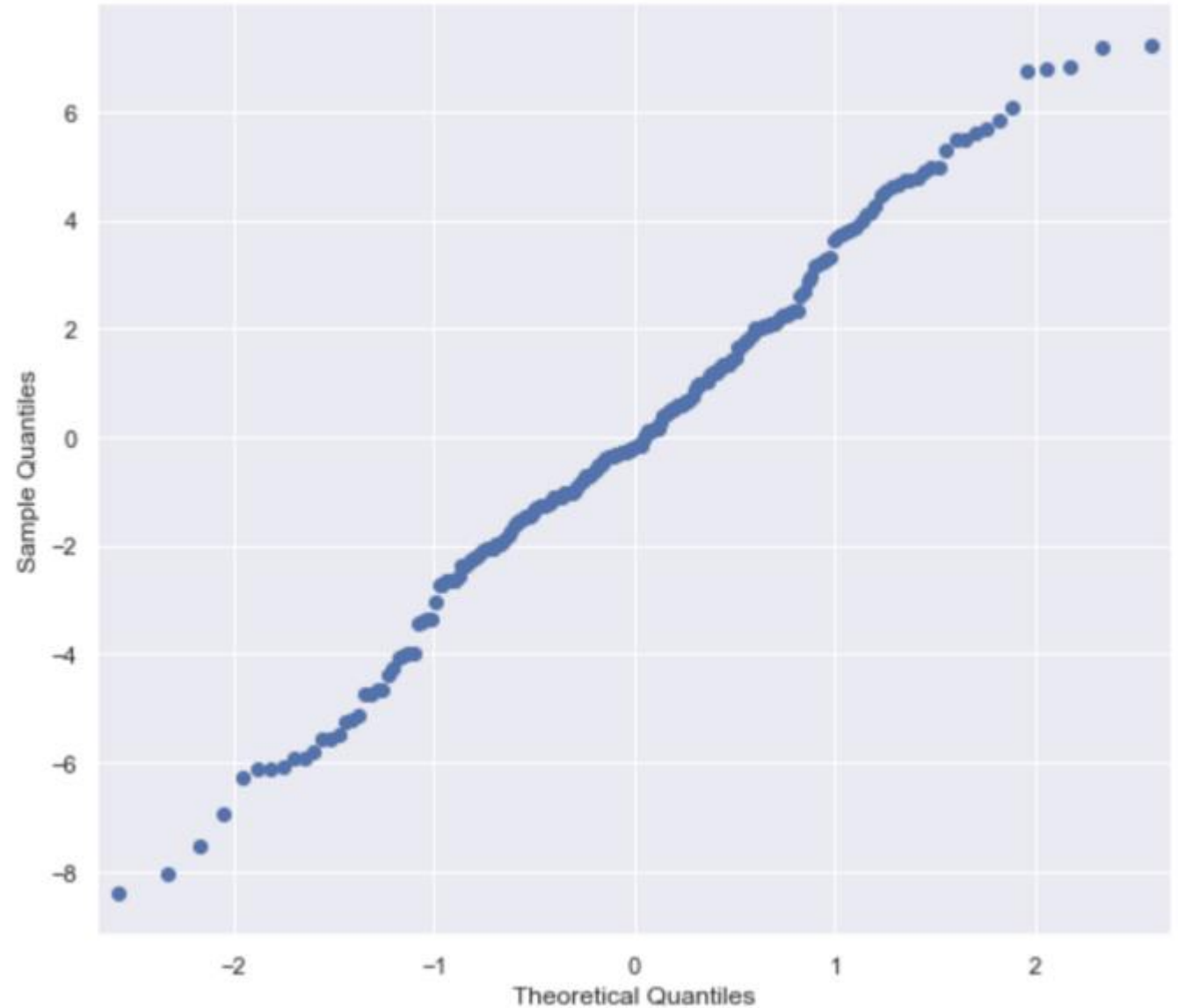
# Linearity

- There is a linear relationship between the dependent variable ( $y$ ) and the independent variable ( $x$ ). We can check for linearity by creating a scatter plot.



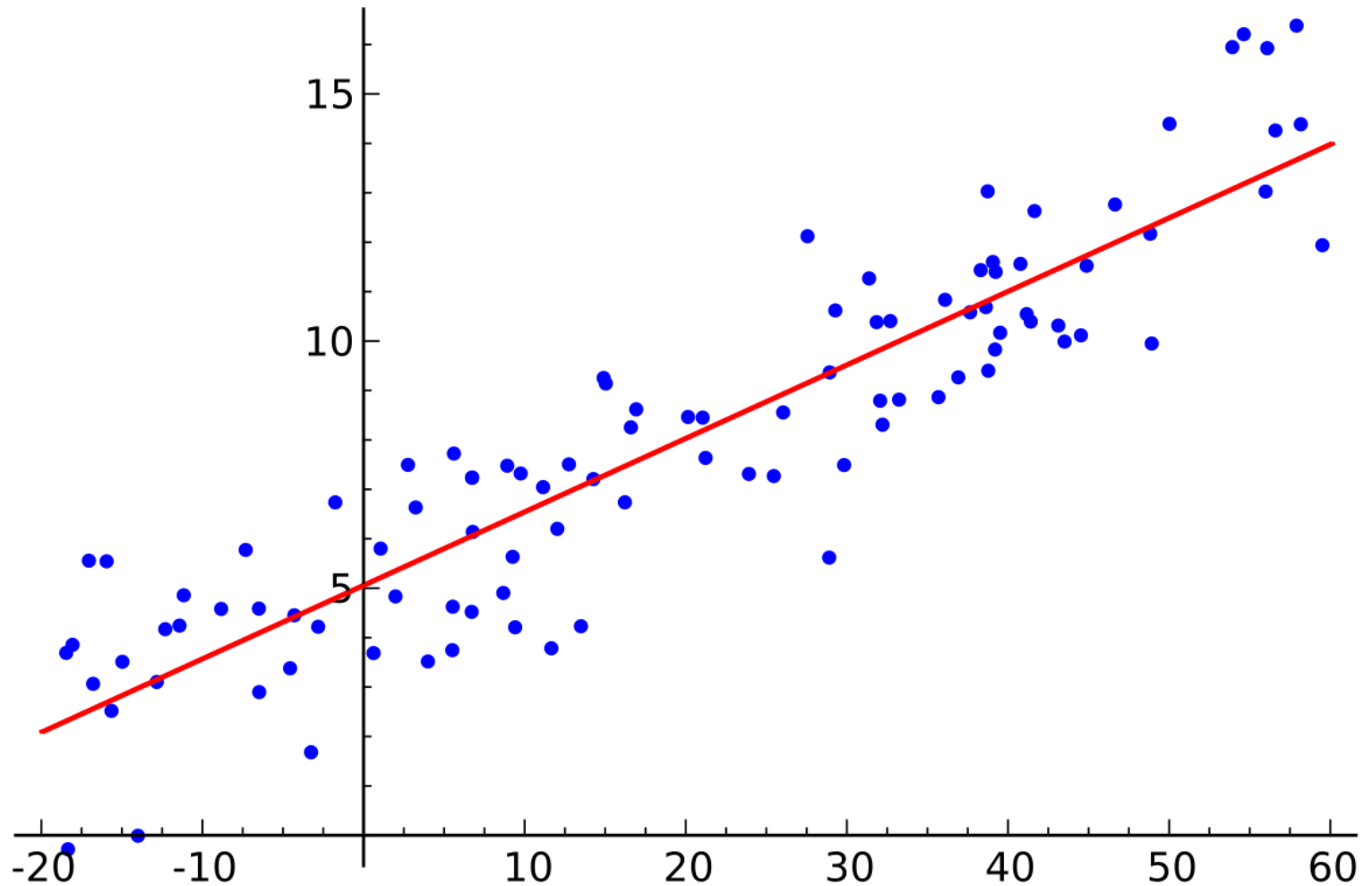
# Normality

- This assumption states that the residuals (difference between  $\text{actual}_y$  and  $\text{predicted}_y$ ) of a model are normally distributed. This assumption can be checked by created histograms or Q-Q-Plots (quantile-quantile-plots are scatterplots of two sets of quantiles plotted against each other.)



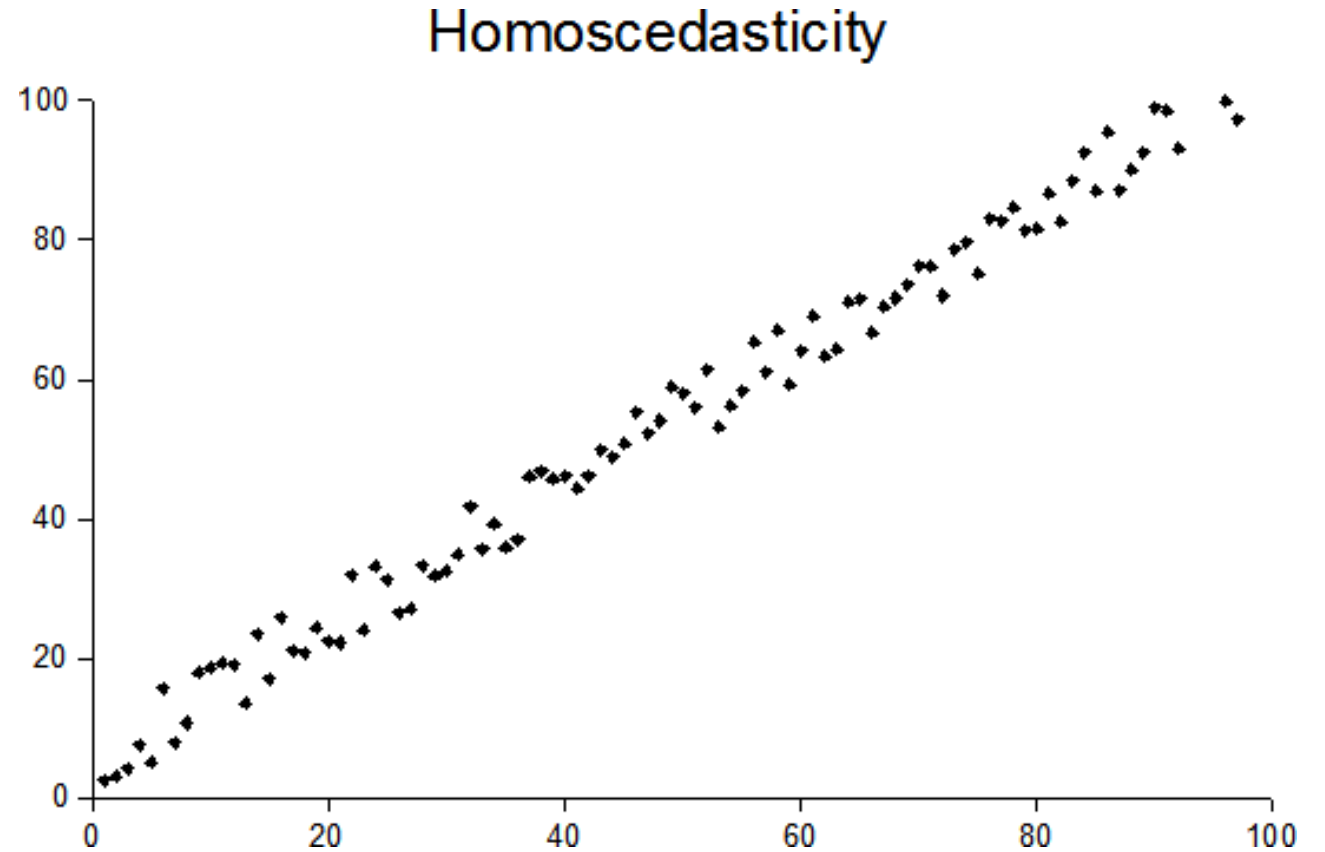
# Multiple Linear Regression

- is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. Multiple regression is an extension of linear regression that uses just one explanatory variable.
- *Ex: multiple factors predict the outcome of Stock Price*



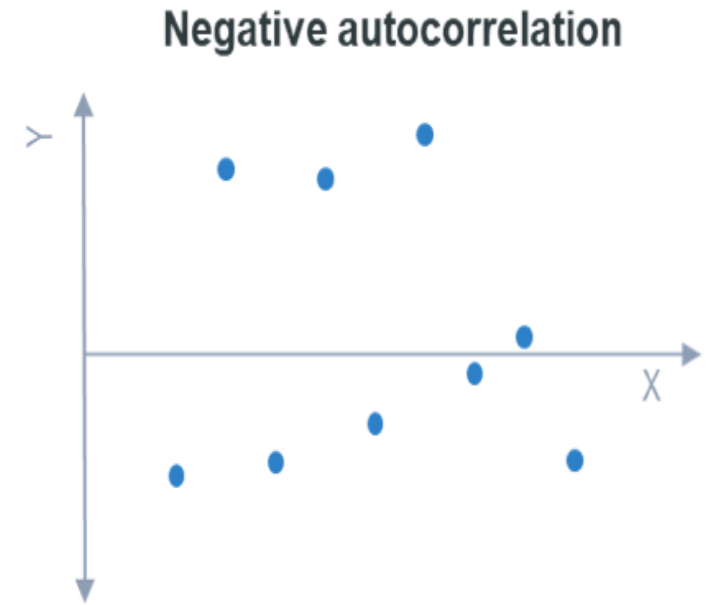
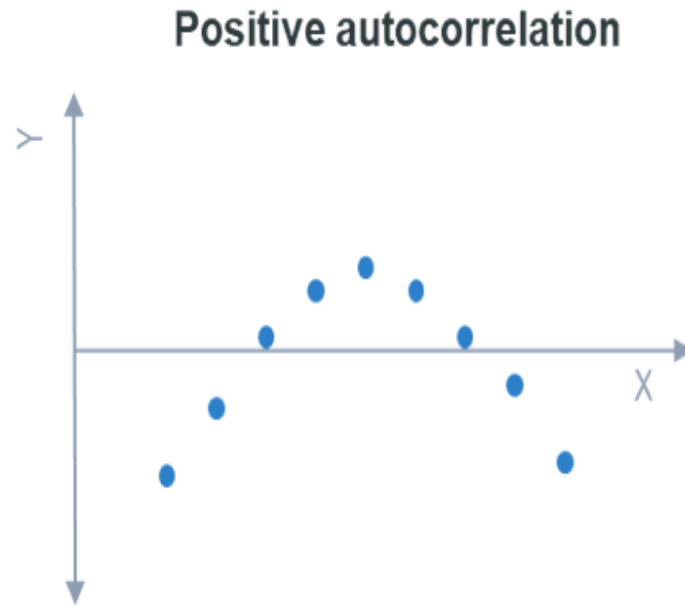
# Homoscedasticity

- Homoscedasticity refers to the variability of the dependent variable being equal across the independent variable values.
- at each value of  $x$ , the  $y$ -value of the dots has about the same variance.



# No Autocorrelation

- No autocorrelation refers to a situation in which no identifiable relationship exists between the values of the error term.
- *there is some information left over which should be accounted for in the model in order to obtain better forecasts.*



# Logistic Regression

- Linear Regression is always used for representing the relationship between some continuous values. However, contrary to this Logistic Regression works on discrete values.
- Logistic regression finds the most common application in solving binary classification problems, that is, when there are only two possibilities of an event, either the event will occur or it will not occur (0 or 1).

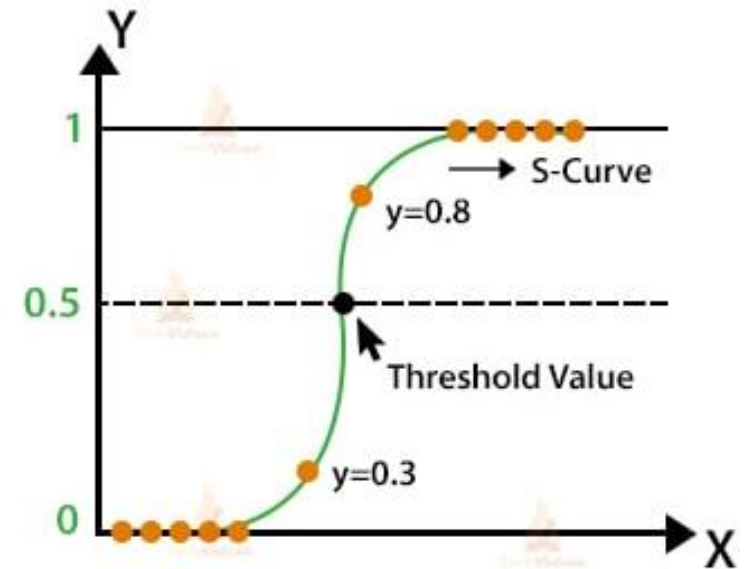


# Logistic Regression

- Thus, in Logistic Regression, we convert the predicted values into such values that lie in the range of 0 to 1 by using a non-linear transform function which is called a logistic function.
- The logistic function results in an S-shaped curve and is therefore also called a Sigmoid function given by the equation:

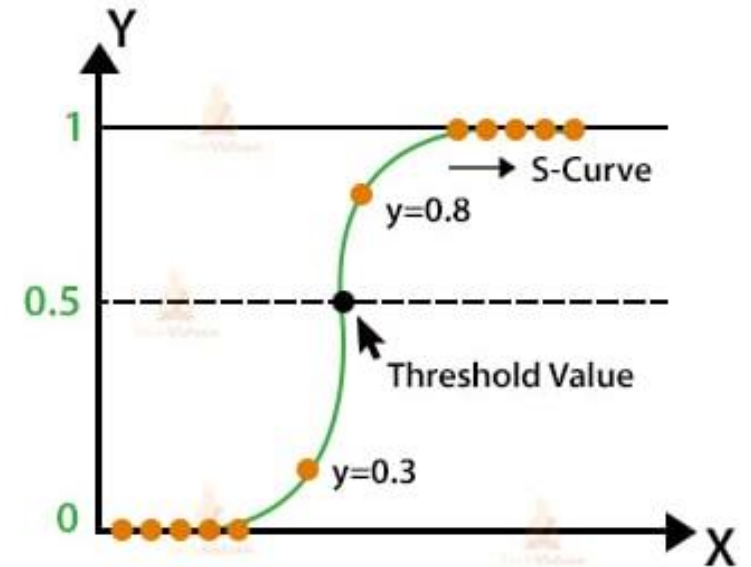
$$f(x) = \frac{1}{1 + e^{-x}}$$

which means as  $f(x)$  approaches infinity, the probability becomes 1, and as  $f(x)$  approaches negative infinity, the probability becomes 0. The model sets a threshold value that decides what range of probability is mapped to which binary variable.



# Logistic Regression

- Suppose we have two possible outcomes, *true* and *false*, and have set the threshold as 0.5. A probability less than 0.5 would be mapped to the outcome *false*, and a probability greater than or equal to 0.5 would be mapped to the outcome *true*.



# Logistic Regression

- Imagine a situation where you might have to decide if a system breached with **DDoS attack(1)** or **not(0)**:
- Imagine a situation where you might have to decide if a hand gesture is **detected(1)** or **not(0)**:
- Imagine a situation where you might have to decide if you've received spam **email (1)** or **not(0)**:
- Imagine a situation where you might have to decide if a student is interested in attending this **session(1)** or **not(0)**:

# Logistic Regression

- Therefore,

$$P(\text{System Corrupted}) = 0.9 \quad \textbf{(90\%)}$$

***Odds*** is the ratio of the probabilities of positive class and negative class.

$$\text{Odds (System)} = P(\text{Corrupted}) / P(\text{Not Corrupted})$$

# Log odds is logarithm of odds

For example, the output can be a probability that if the system is compromised by hacker is 95% or the probability that Btech student will go abroad for masters is 70%. However, in most cases, probabilities are used to classify data points. If the probability is greater than 50%, the prediction is ***positive class (1)***. Otherwise, the prediction is ***negative class (0)***.

Probability	Odds	Log Odds
0,05	0,05	-1,28
0,1	0,11	-0,95
0,2	0,25	-0,60
0,3	0,43	-0,37
0,4	0,67	-0,18
0,5	1,00	0,00
0,6	1,50	0,18
0,7	2,33	0,37
0,8	4,00	0,60
0,9	9,00	0,95
0,95	19,00	1,28

# Logistic Regression

Everything seems ok up until now except for one issue. It is not always desired to choose positive class for all probability values higher than 50%. Regarding the DDoS attack case, we have to be almost sure in order to classify a system is vulnerable of DDoS attack. Since network resources were predominantly used by system to access data, we do not want the user to miss important data. Data in packets are not classified as ping of death unless we are almost sure. So the value that serves as a threshold between positive and negative class is problem-dependent. Good thing is that logistic regression allows us to adjust this threshold value.

# Logistic Regression

If we set a **high threshold (i.e. 90%)**, almost all the predictions we made as positive will be correct. However, we will miss some of the positive class and label them as negative.

If we set a **low threshold (i.e. 35%)**, we will predict almost all the positive classes correctly. However, we will classify some of the negative classes as positive.

Both of these cases will affect the accuracy of our model. The simplest way to measure accuracy is:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Number of All Predictions}}$$

# Why Logistic Regression?

- Logistic regression introduces a decision threshold – the threshold value holds significance because it directly affects the classification problem and regression model. This threshold allows the value to range strictly between 0 and 1.
- Logistic Regression is used when the dependent variable is categorical.
- We cannot use linear regression because no classification based threshold can be made, therefore we use logistic regression.
- Linear regression assumes that the data follows a linear function, while logistic regression models the data using the sigmoid function (S-curve).
- The threshold value is completely dependent on two factors, *precision* and *recall*. we want both precision and recall to have a value of **1**. However, this is not realistically possible, so we settle on trade-off values.







# Why Logistic Regression?

- Now, before getting into precision and recall, I would like to define some terms here in confusion matrix that depicts possible outcomes:

A **true positive (TP)** is an outcome where the model **correctly predicts** the *positive* class. Similarly, a **true negative (TN)** is an outcome where the model **correctly predicts** the *negative* class.

A **false positive (FP)** is an outcome where the model **incorrectly predicts** the *positive* class. And a **false negative (FN)** is an outcome where the model **incorrectly predicts** the *negative* class.

		Actual Labels or Answer or Values	
		Postive (Dog)	Negative (Cat)
Machine Learning Model Answer Predicted Values	Postive (Dog)	✓  1	✗  2
	Negative (Cat)	✗  3	✓  4

# Logistic Regression Scores

- It is desired to make a prediction that is either TP or TN so the models aim to maximize TP and TN values. (but we cannot maximize both due to trade-off)
- **Precision** measures how good our model is when the prediction is positive.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** measures how good our model is at correctly predicting positive classes.

$$Recall = \frac{TP}{TP + FN}$$

# Logistic Regression Scores

- Increasing precision decreases recall and vice versa. You can aim to maximize precision or recall depending on the task. For an system compromised with DDoS attack, we try to maximize precision because we want to be correct when a system is vulnerable against DDoS. We do not want to label a normal packet as vulnerable (i.e. false positive). If false positive is low, then precision is high.
- There is another measure that combines precision and recall into a single number: **F1\_score**. It is the weighted average of precision and recall and calculated as:

$$F1\_score = 2 \frac{Precision * Recall}{Precision + Recall}$$

# Logistic Regression Scores

- **AUC: The Area Under the curve (AUC)** is an aggregated metric that evaluates how well a logistic regression model classifies positive and negative outcomes at all possible cutoffs. It can range from 0.5 to 1, and the larger it is the better.
- **CA – Classification Accuracy Score**: measure of prediction accuracy for that data set, the score obtained is correctly classified by the model.

# Logistic Regression

