

# Notas escépticas sobre el Machine Learning

Lecture 2: Classification Problems

---

# Classification problem in ML

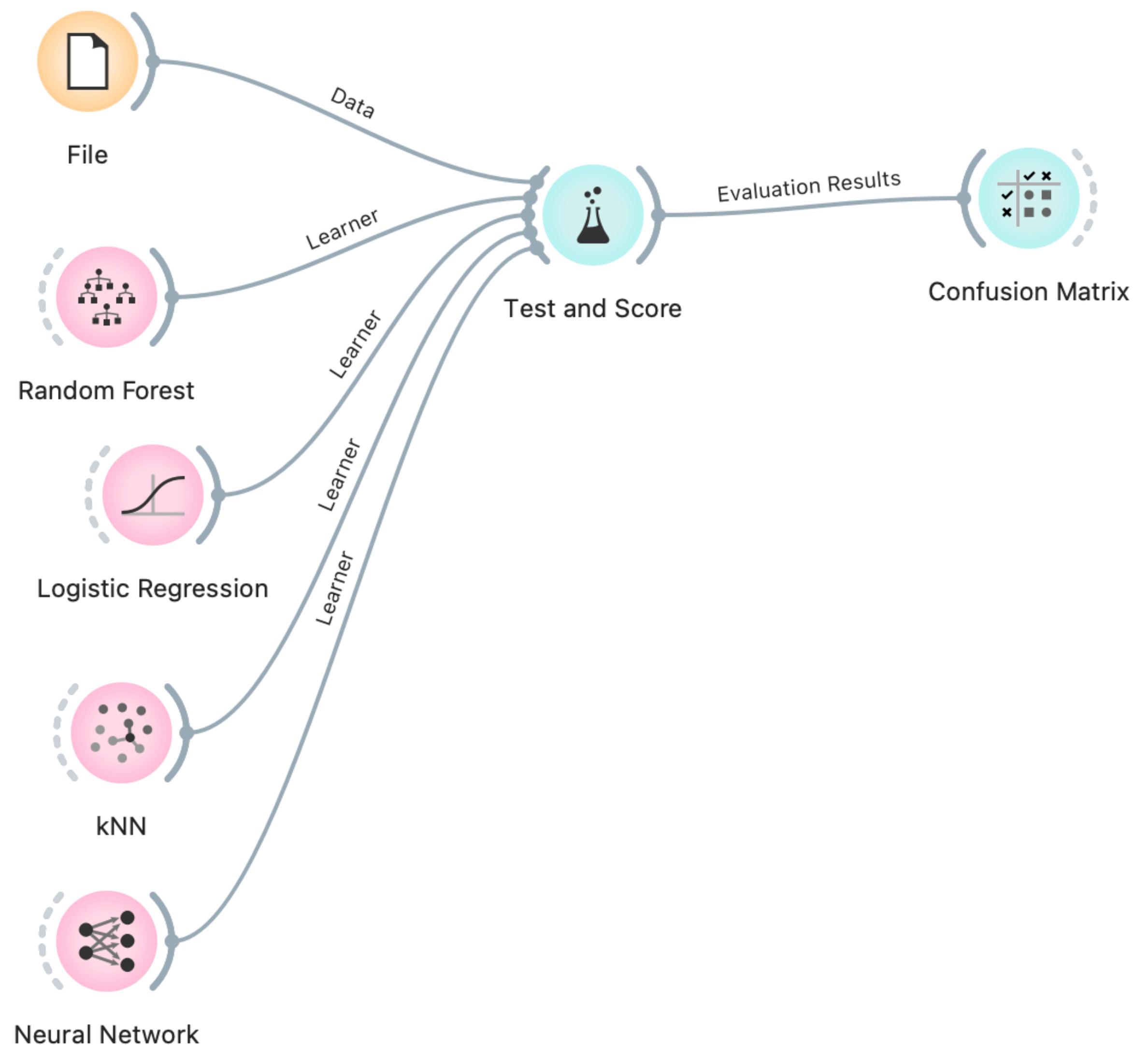
ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000025	5	1	1	1	2	1	3	1	1	benign
1002945	5	4	4	5	7	10	3	2	1	benign
1015425	3	1	1	1	2	2	3	1	1	malignant
1016277	6	8	8	1	3	4	3	7	1	benign
1017023	4	1	1	3	2	1	3	1	1	benign
1017122	8	10	10	8	7	10		7	1	malignant
1018099	1	1	1	1	2	10	3	1	1	benign
1018561	2	1	2	H	2	1	3	1	1	benign
1033078	2	1	1	1	2	1	1	1	5	benign
1033078	4	2	1	1	2	1	2	1	1	benign

9/03/2022

# Classification

In classification, the goal is to predict a *class label*, which is a choice from a predefined list of possibilities.

- Binary Classification
- Multiclass Classification



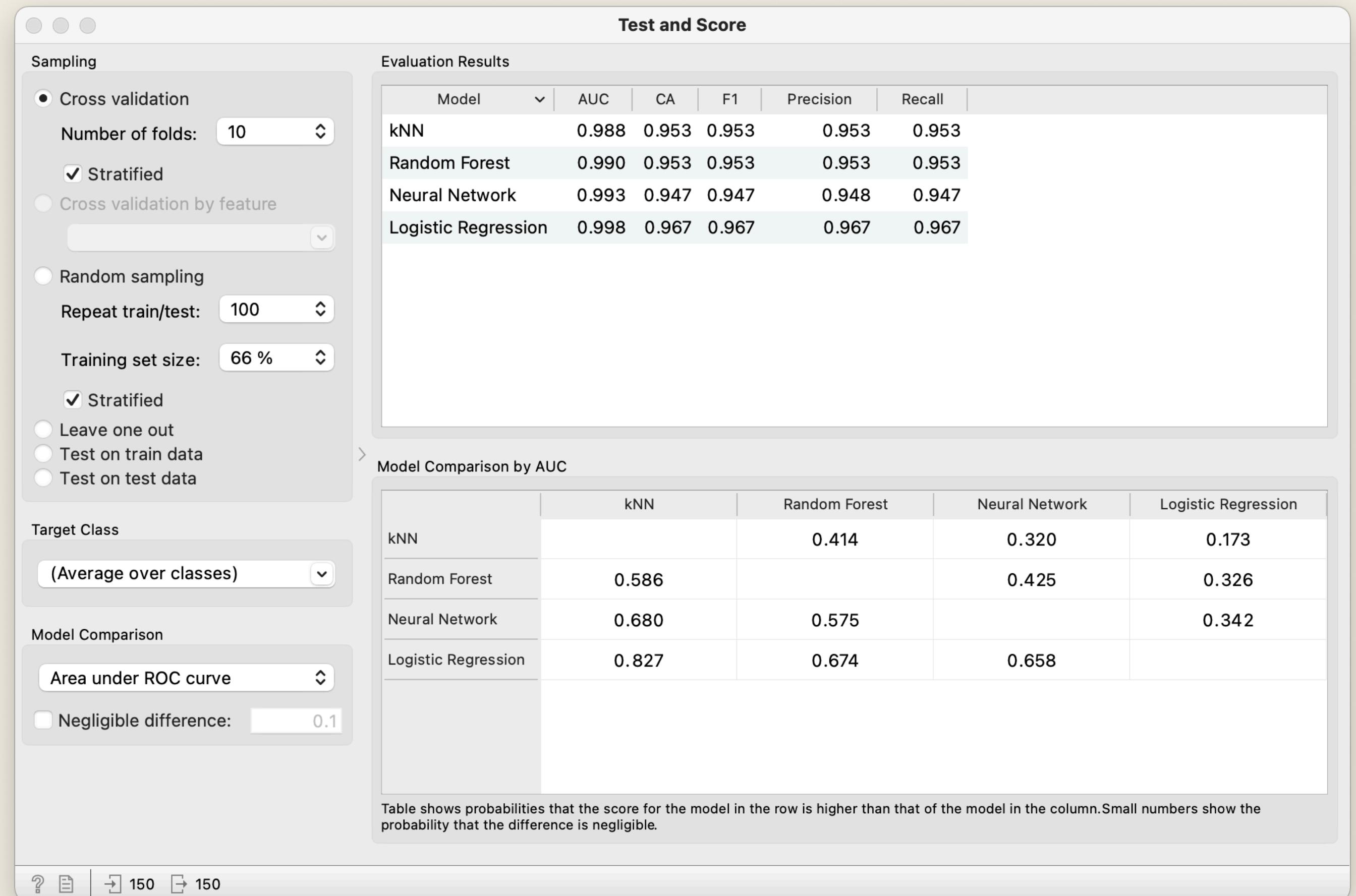
9/MARZO/2022

# Scoring

- K-cross validation
- Test & Score
- Confusion Matrix

# Evaluation metrics in classification

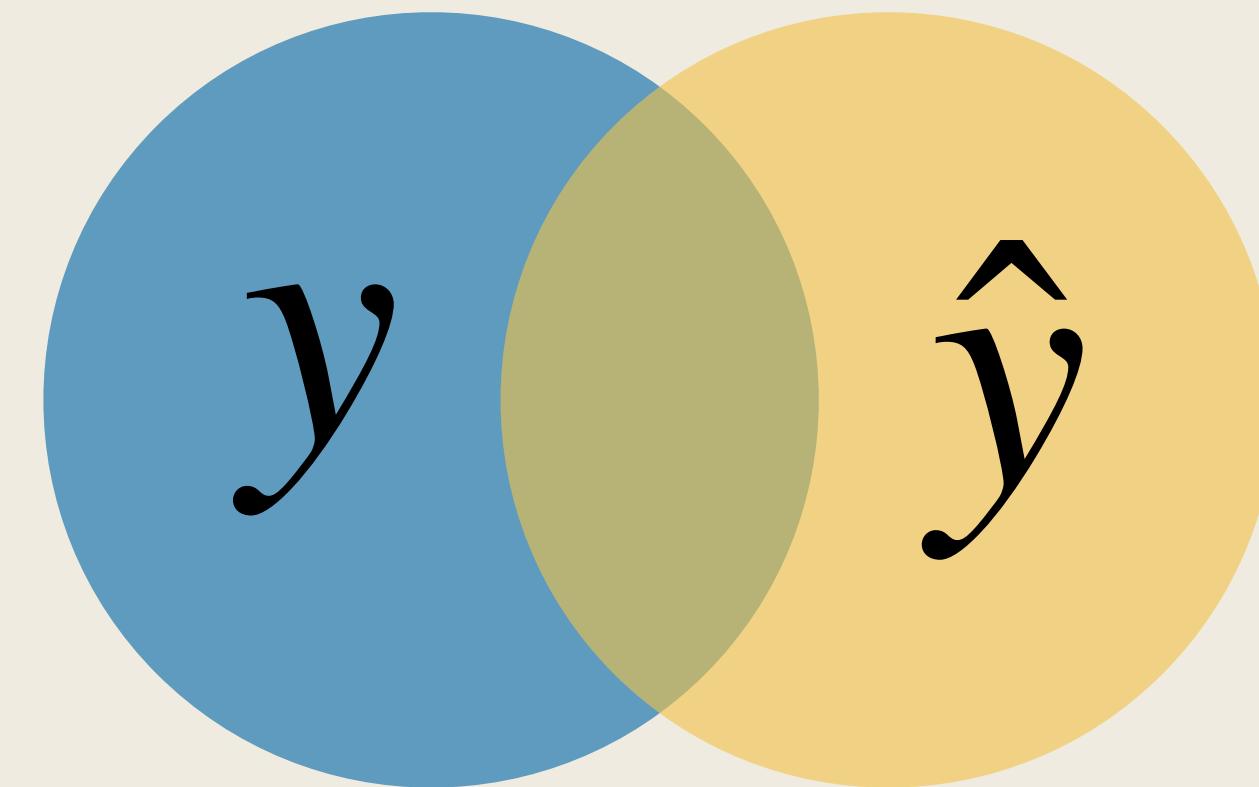
- We are talking about some metrics:
- Jaccard index,
- Confusion matrix



# Jaccard index

$y$  = True values (actual values)

$\hat{y}$  = Predicted values of our model



Jaccard index [0,1]

$$J(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|}$$

Ejemplo

$$y = [0,0,0,0,0,0,1,1,1,1,1] \quad \hat{y} = [1,1,0,0,0,0,1,1,1,1,1]$$

$$J(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|} = \frac{8}{10 + 10 - 8} = 0.66$$

# Confusion Matrix

TP = True Positives

FN = False Negatives

FP = False Positives

TN = True Negatives

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

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

$$F1\text{-score } [0,1] = \frac{2 * Precision * Recall}{Precision + Recall}$$

		Confusion Matrix			$\Sigma$	
		Predicted				
Actual		Iris-setosa	Iris-versicolor	Iris-virginica	$\Sigma$	
		Iris-setosa	50	0	0	50
		Iris-versicolor	0	47	3	50
		Iris-virginica	0	2	48	50
		$\Sigma$	50	49	51	150

**Classification Accuracy** is the proportion of correctly classified examples

$$CA = \frac{TP}{Total}$$

---

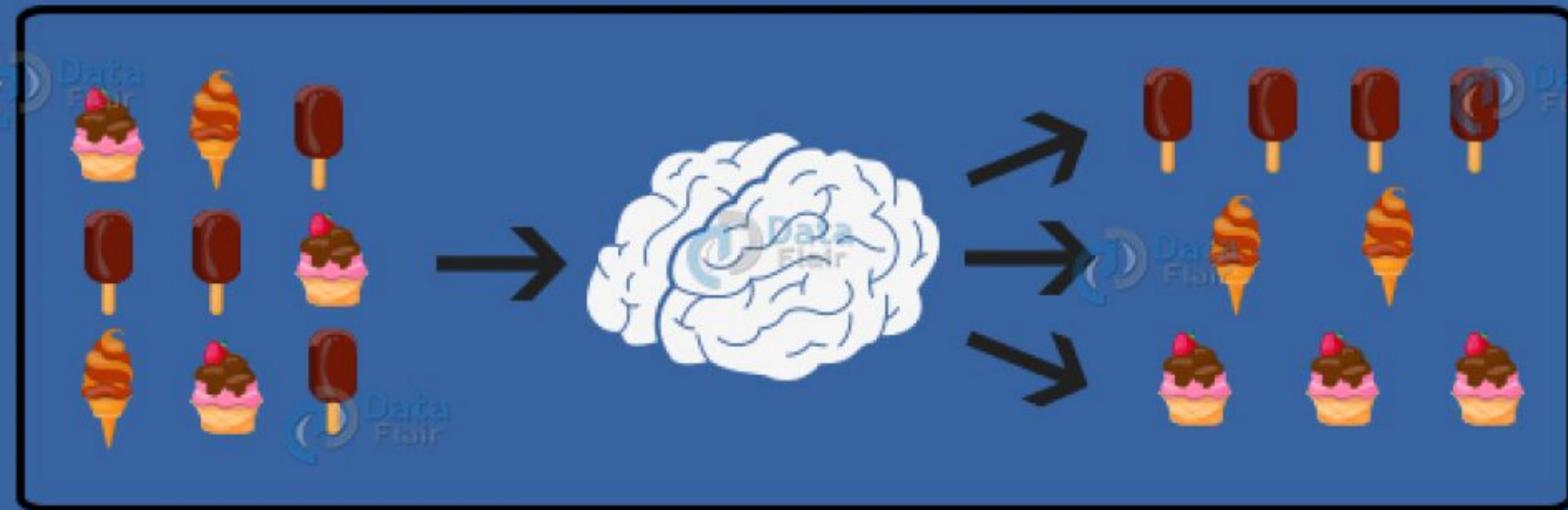
# Classification Algorithms

# Machine Learning Classification Algorithms

Logistic Regression

Naive Bayes

Decision Tree



Support Vector Machines

Random Forest

K-Nearest Neighbour

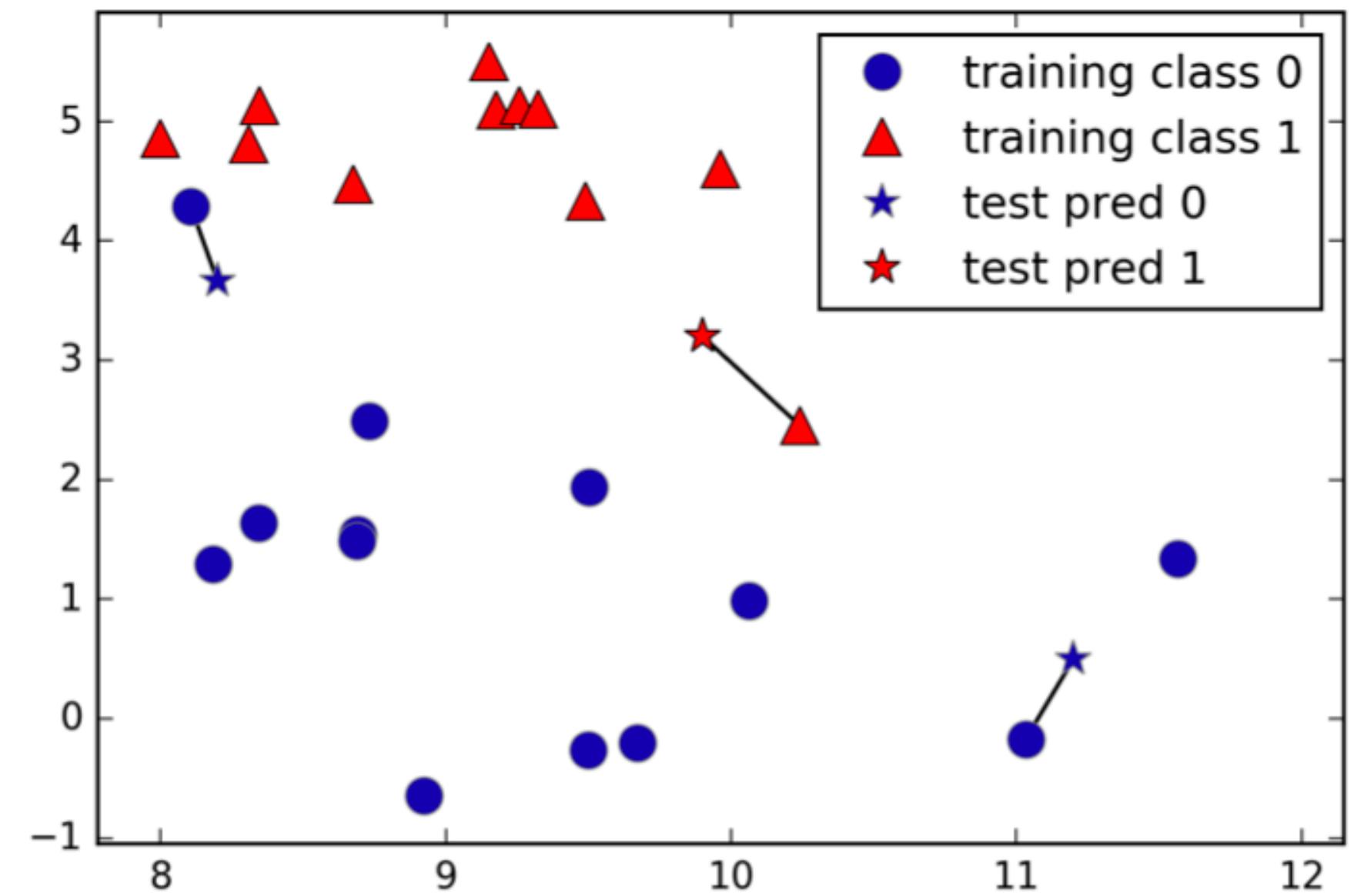


Figure 2-4. Predictions made by the one-nearest-neighbor model on the forge dataset

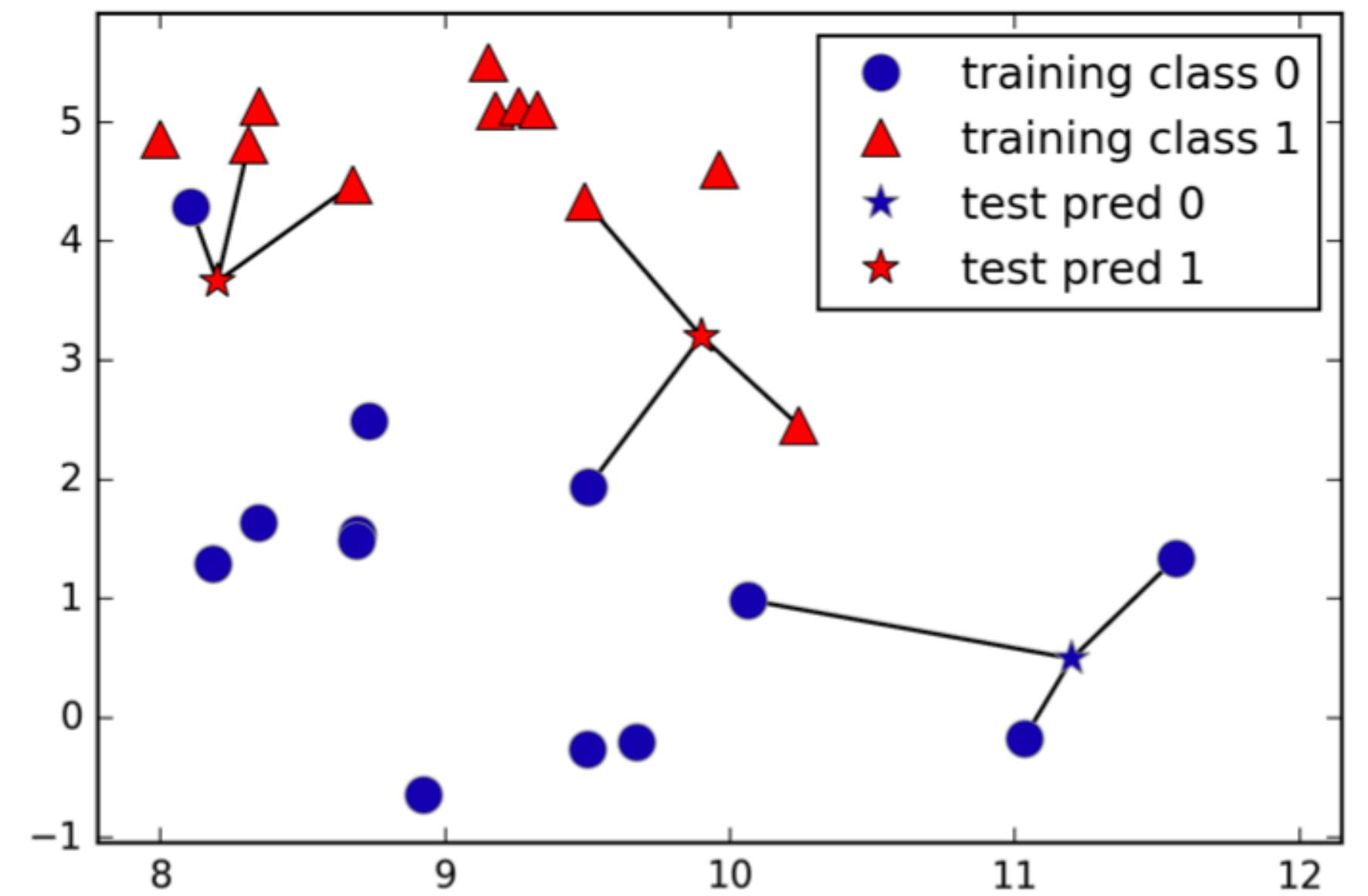
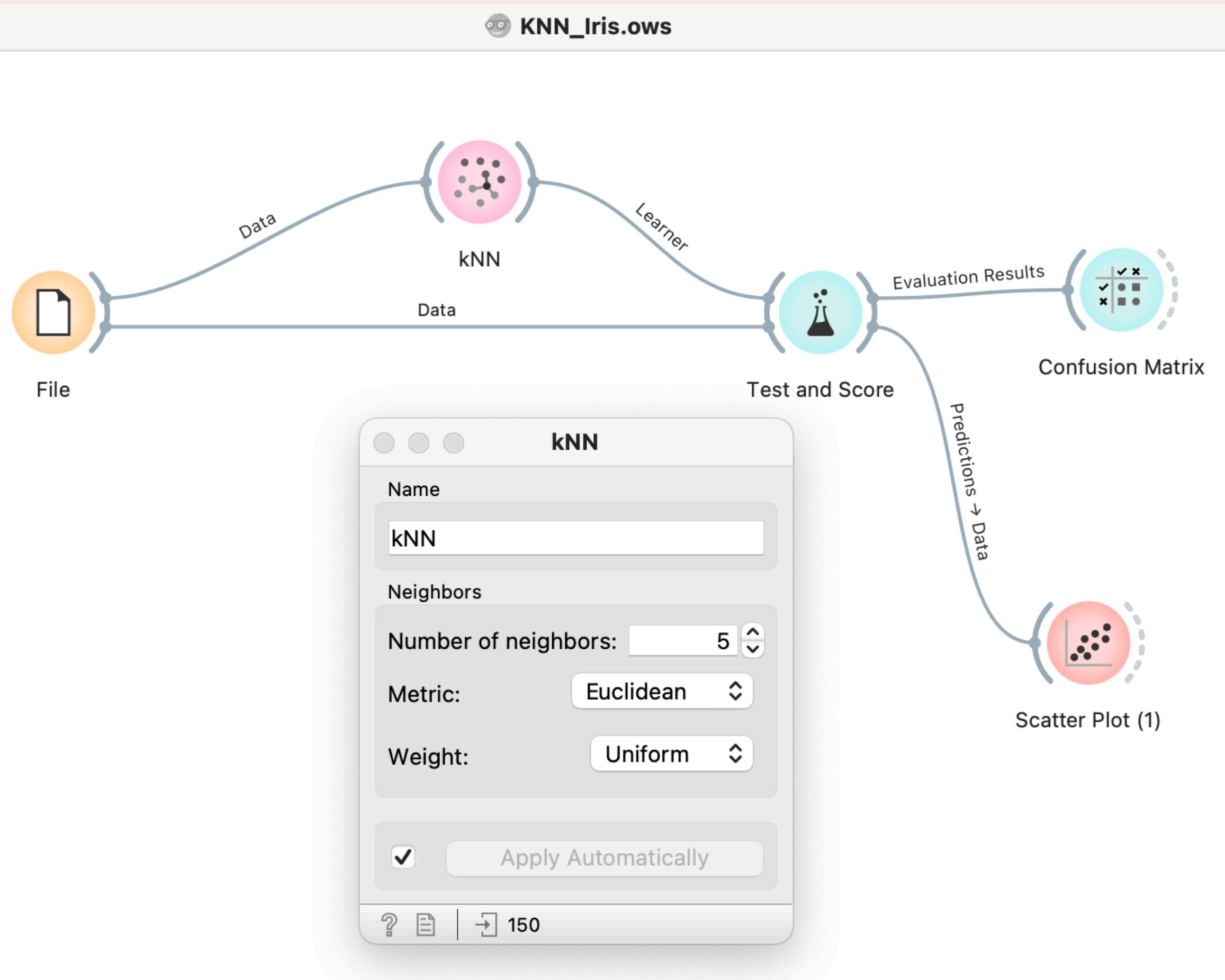


Figure 2-5. Predictions made by the three-nearest-neighbors model on the forge dataset

16/03/2022

# K-Nearest Neighbors

- Building the k-NN algorithm consists only of storing the training dataset.
- To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset—its “nearest neighbors.”



16/MARZO/2022

# K-nearest Neighbors

- Iris
- KNN → Test Score → Confusion Matrix
- Constant ->Test Score -> Confusion Matrix
- Scatter Plot (KNN)

Jupyter Knn\_proofs Last Checkpoint: hace 7 horas (autosaved)

Logout

Edit View Insert Cell Kernel Widgets Help Trusted Python [conda env:root]\*

Modelos Knn

Vamos a realizar nuestros primeros modelos Knn. La librería más utilizada para ML en python es `sklearn`. De aquí vamos a importar las bases de datos y las funciones que necesitamos.

1. Vamos a trabajar con la base de datos de Iris, para comparar con lo que hemos hecho en Orange3
2. Vamos a trabajar con otra base de datos, una sobre cancer para ver cuando se nos cruzan la precisión en el train y test.

El primer paso por tanto es importar la base de datos de iris, `load_iris` y la función que me va a dividir mi base de datos para entrenar y datos para hacer el test.

```
In [1]: from sklearn.datasets import load_iris  
from sklearn.model_selection import train_test_split
```

```
In [2]: iris=load_iris()
```

```
In [3]: iris.keys()
```

```
Out[3]: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

`iris` es un diccionario de python con las siguientes claves. Nosotros estamos interesados en `target` como clase a predecir y `data` como conjunto de características (features).

16/MARZO/2022

# K-nearest Neighbors in Python

- Iris
- KNN → Test Score → Confusion Matrix

The screenshot shows the RStudio interface. In the top-left, there are two tabs: 'First\_program.r' and 'Knn.R\*'. The 'Knn.R\*' tab contains the following R code:

```

1 library(ggplot2)
2 library(gridExtra)
3 library(caret)
4 library(class)
5 library(datasets)
6
7 data(iris)
8 summary(iris)
9
10 p1 <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, col=Species)) +
+   geom_point(alpha=0.8)
11
12 p2 <- ggplot(iris, aes(x=Petal.Length, y=Petal.Width, col=Species)) +
+   geom_point(alpha=0.8)
13
14 grid.arrange(p1, p2, ncol=2)
15
16 iris_tr_feat <- iris[,1:4]
17
18 #Creating Training and Test data set. Training data will be used to build model wh
19
20 #set.seed(123) # To get the same random sample
21
22 set.seed(123) # To get the same random sample

```

In the top-right, the 'Environment' tab shows variables: moddf, p1, p2, test.iris, train.iris, ACC.6, dat.d, knn.6, and test.gc\_labels.

The bottom section is the 'Console' tab, which displays:

```

R 4.1.2 · ~/Dropbox/Research/Burocracia_Research/talks/Curso de ML_Biomedicina/
Mcnemar's Test P-Value : NA

Statistics by Class:

      Class: setosa Class: versicolor Class: virginica
Sensitivity       1.0000      0.9444        1.0000
Specificity       1.0000      1.0000        0.9688
Pos Pred Value    1.0000      1.0000        0.9286
Neg Pred Value    1.0000      0.9643        1.0000
Prevalence        0.3111      0.4000        0.2889
Detection Rate    0.3111      0.3778        0.2889
Detection Prevalence 0.3111      0.3778        0.3111
Balanced Accuracy 1.0000      0.9722        0.9844

```

16/MARZO/2022

# K-nearest Neighbors in R

- Iris
- KNN → Test Score → Confusion Matrix

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	Hiigh	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A
p15	Middle-age	F	Low	Normal	?

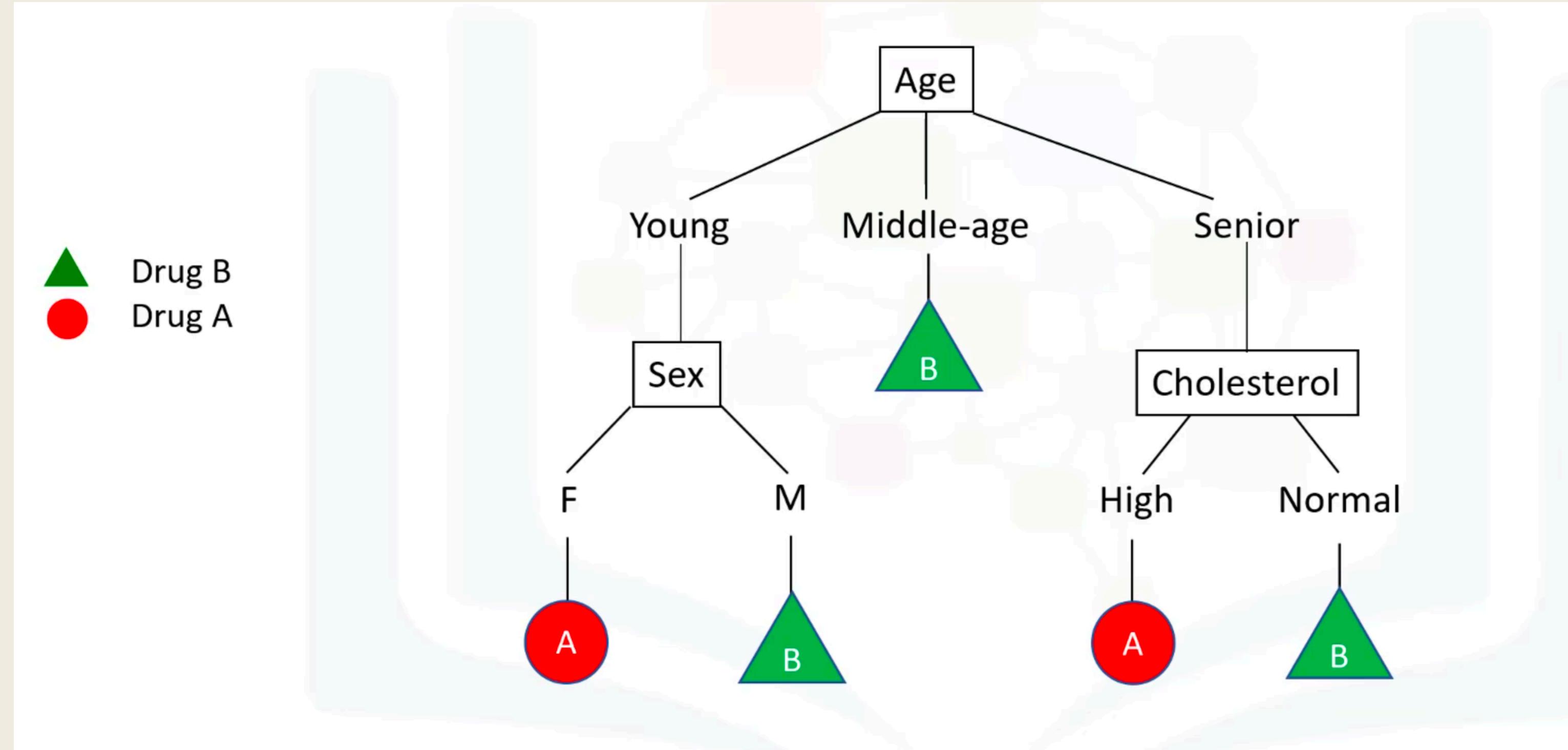
16/MARZO/2022

# Decision Tree

- What exactly is a decision tree?

# How to build a decision tree?

1. Choose an attribute from your dataset.
2. Calculate the significance of attribute in splitting of data.
3. Split data based on the value of the best attribute.
4. Go to step 1.



# How to calculate the significance?

**Information gain** is the information that can increase the level of certainty after splitting.

Information Gain = (Entropy before split) – (weighted entropy after split)

In every step, we want to dismiss the impurity of the nodes. Impurity is calculated by Entropy of data in the node

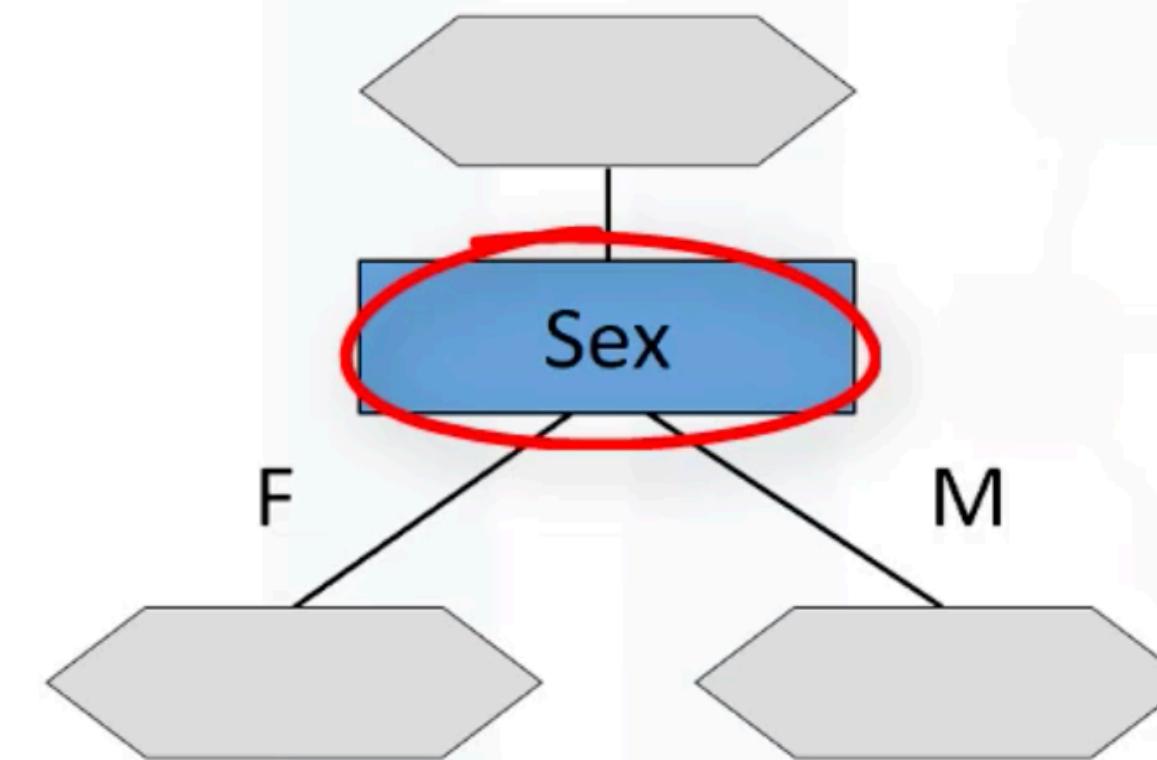
$$E = - \sum_i p_i \log_2(p_i)$$

Entropy is the amount of information disorder.  
If the samples are completely homogeneous the entropy is zero.

# Which attributes is the best?

S: [9 B, 5 A]

E = 0.940



S: [3 B, 4 A]

E = 0.985

S: [6 B, 1 A]

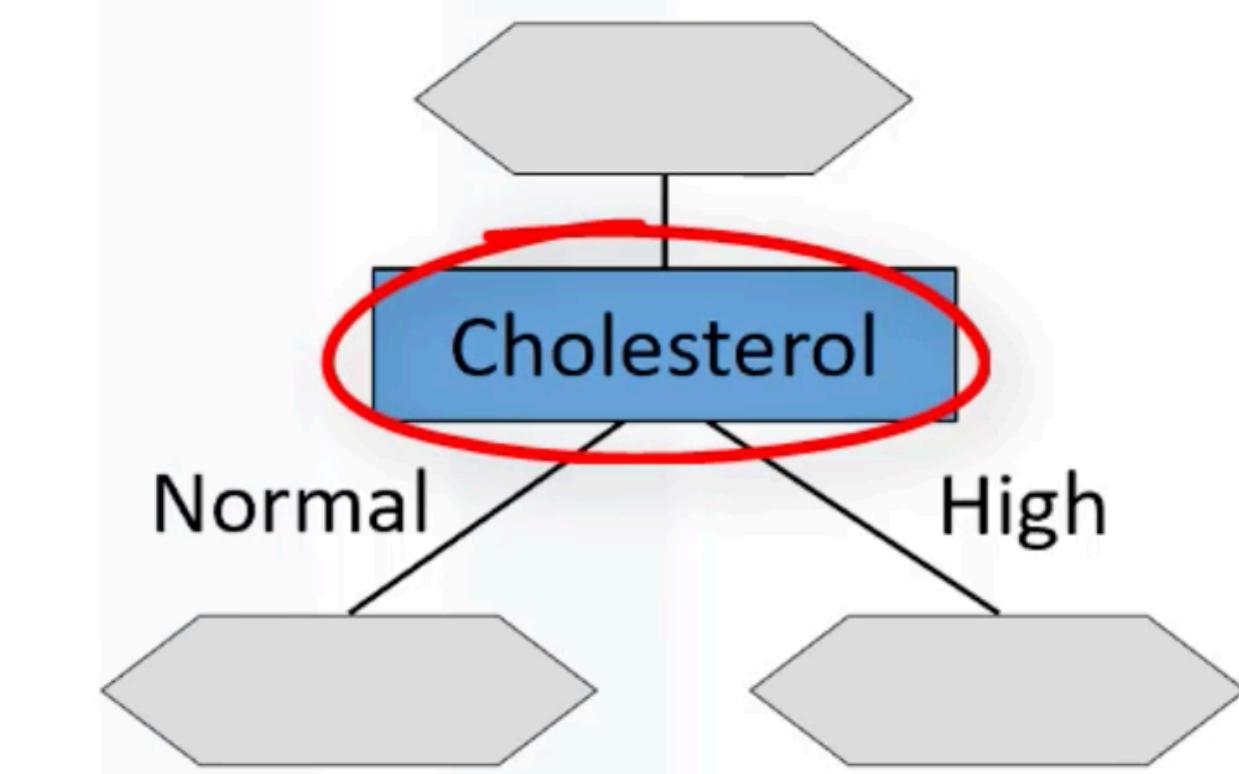
E = 0.592

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	Hiigh	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A

?

S: [9 B, 5 A]

E = 0.940



S: [6 B, 2 A]

E = 0.811

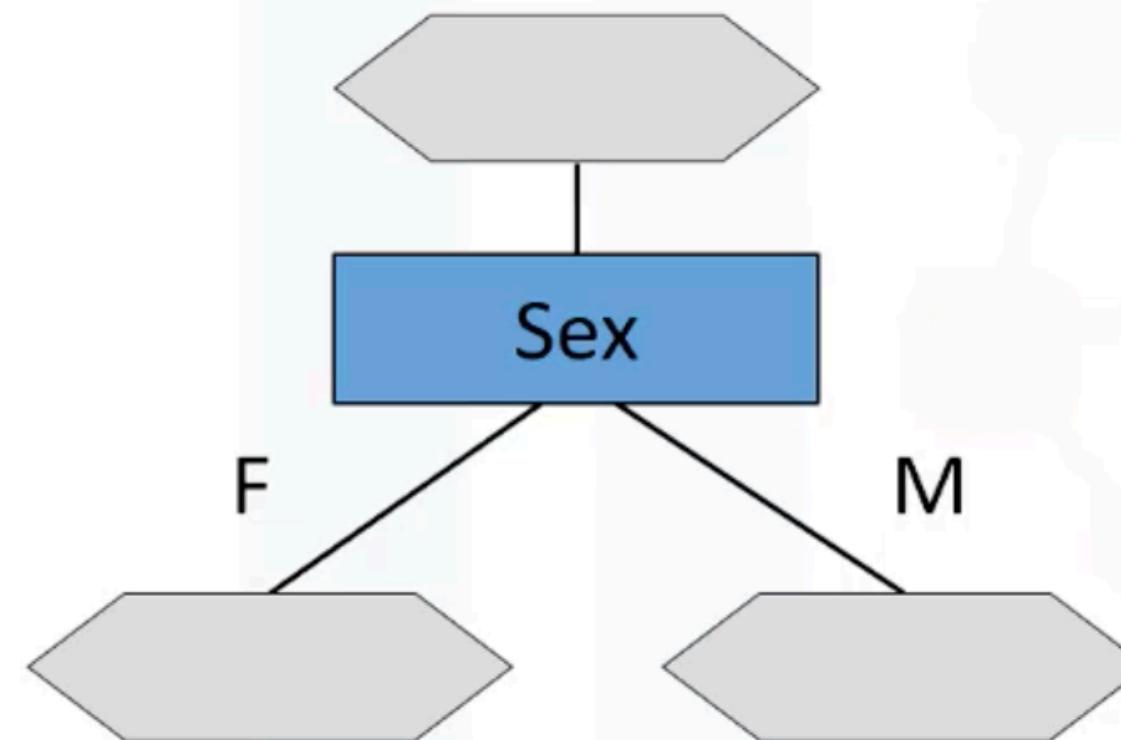
S: [3 B, 3 A]

E = 1.00

$$E = - \sum_i p_i \log_2(p_i)$$

S: [9 B, 5 A]

E = 0.940



S: [3 B, 4 A]

E = 0.985

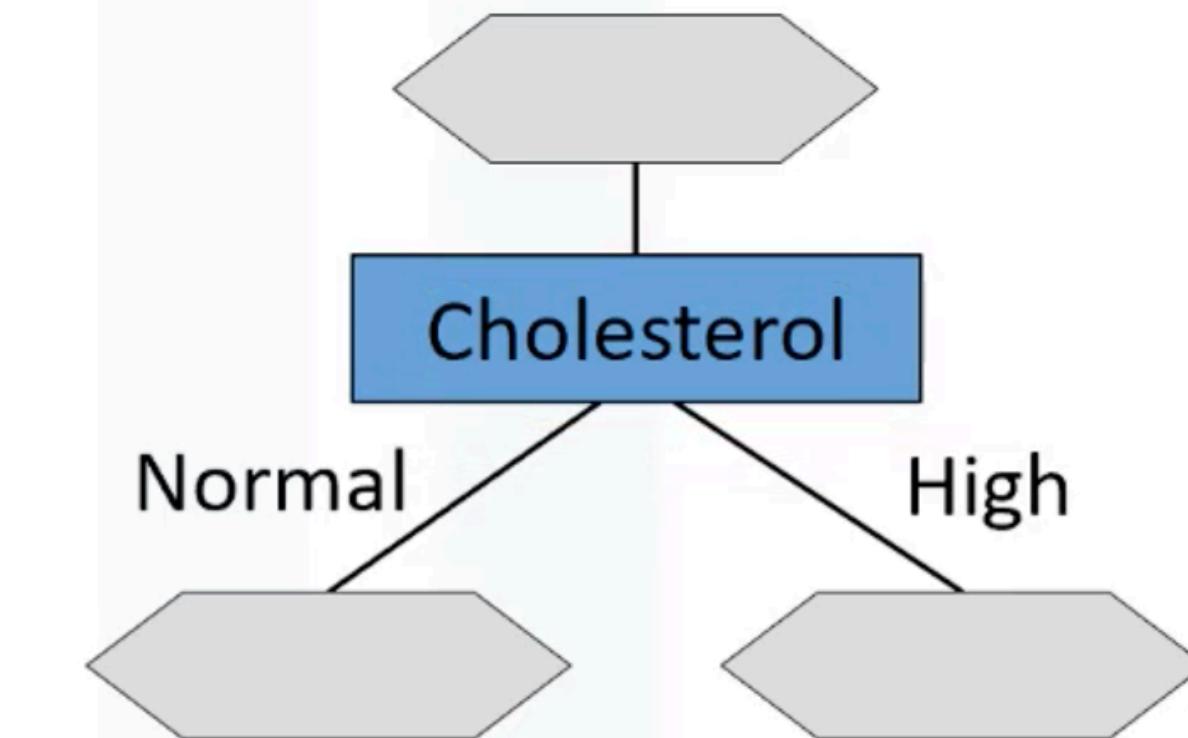
S: [6 B, 1 A]

E = 0.592

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	Hiigh	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A

S: [9 B, 5 A]

E = 0.940



S: [6 B, 2 A]

E = 0.811

S: [3 B, 3 A]

E = 1.00

Gain (s, Sex)

$$= 0.940 - [(7/14)0.985 + (7/14)0.592]$$

$$= 0.151$$

Gain (s, Cholesterol)

$$= 0.940 - [(8/14).811 + (6/14)1.0]$$

$$= 0.048$$

The attribute more suitable give us the higher information gain .  
In this case "SEX"

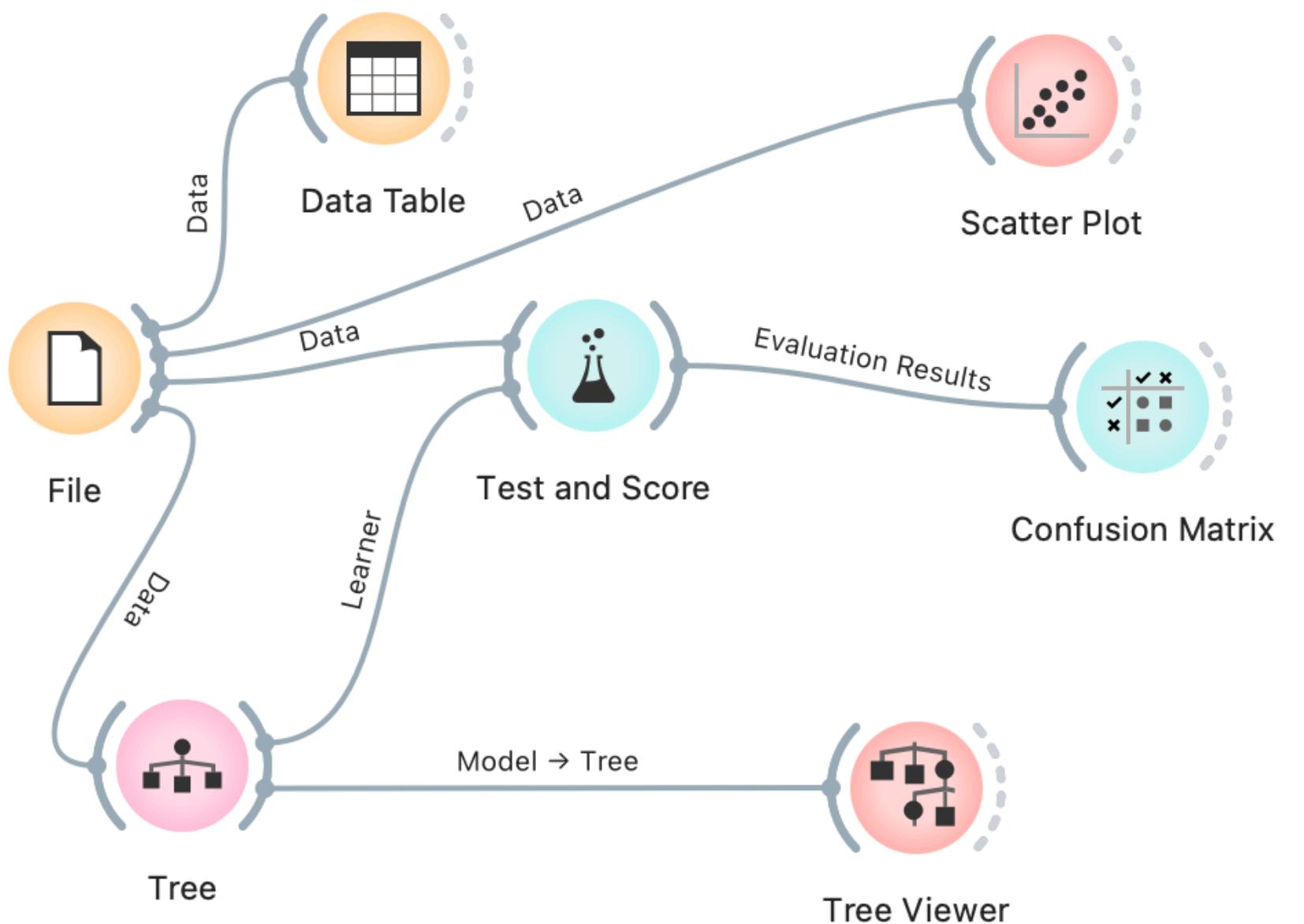
**Information gain** is the information that can increase the level of certainty after splitting.

Information Gain = (Entropy before split) – (weighted entropy after split)

16/MARZO/2022

# Decision Tree

- Datos de Iris
- Test and Score → confusion Matrix
- Tree → Tree Viewer



---

# Linear Models For classification

- Linear models are a class of models that are widely used in practice.
  - Linear models make a prediction using a linear function of the input features
-

# Linear Binary classification

In this case, the prediction is made using the formula:

$$\hat{y} = w[0] * x[0] + w[1]*x[1] + \dots + w[p]*x[p] + b > 0$$

Where:

X [0,..p] are the features (columns)

W[0,..,p] and b are the parameters of the model

Y is the prediction of the model

---

# Linear Models For classification

A linear classifier is a classifier that separates two classes using a line, a plane or a hyperplane

The most common models:

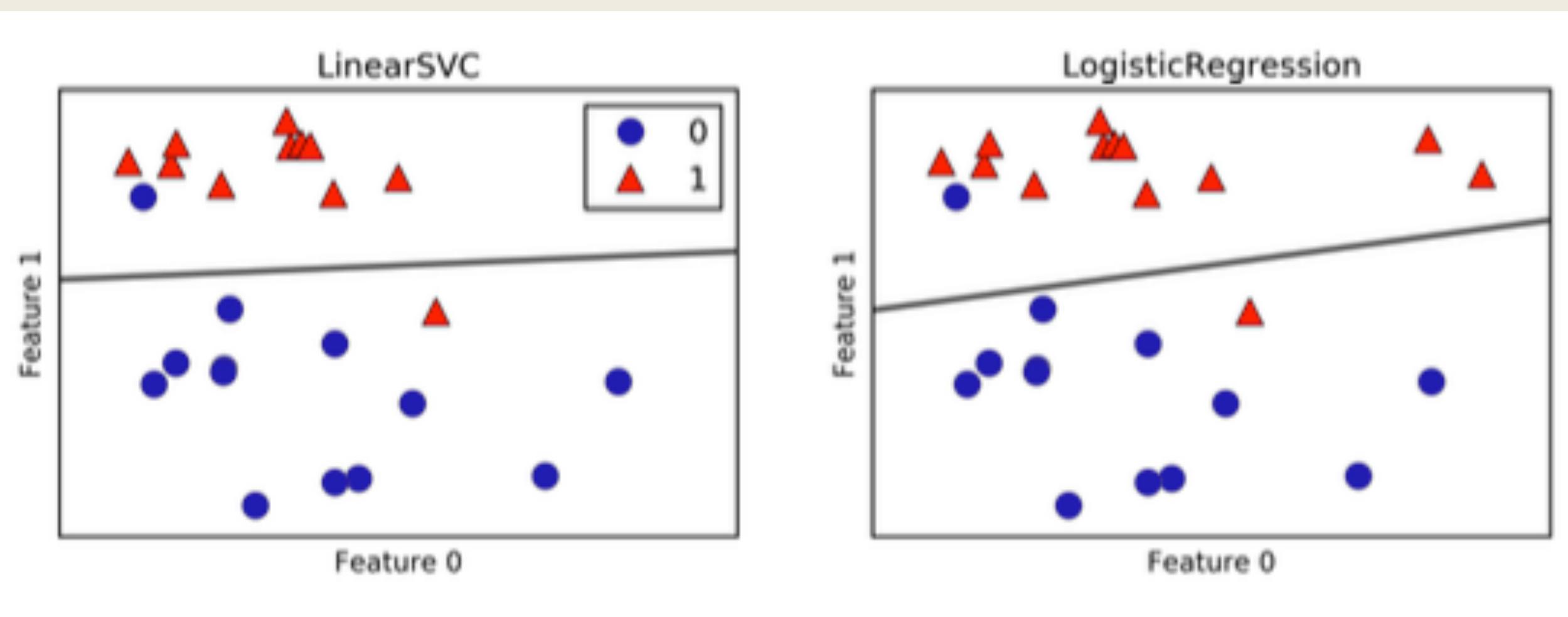
- Logistic Regression
  - Support Vector Machine
-

---

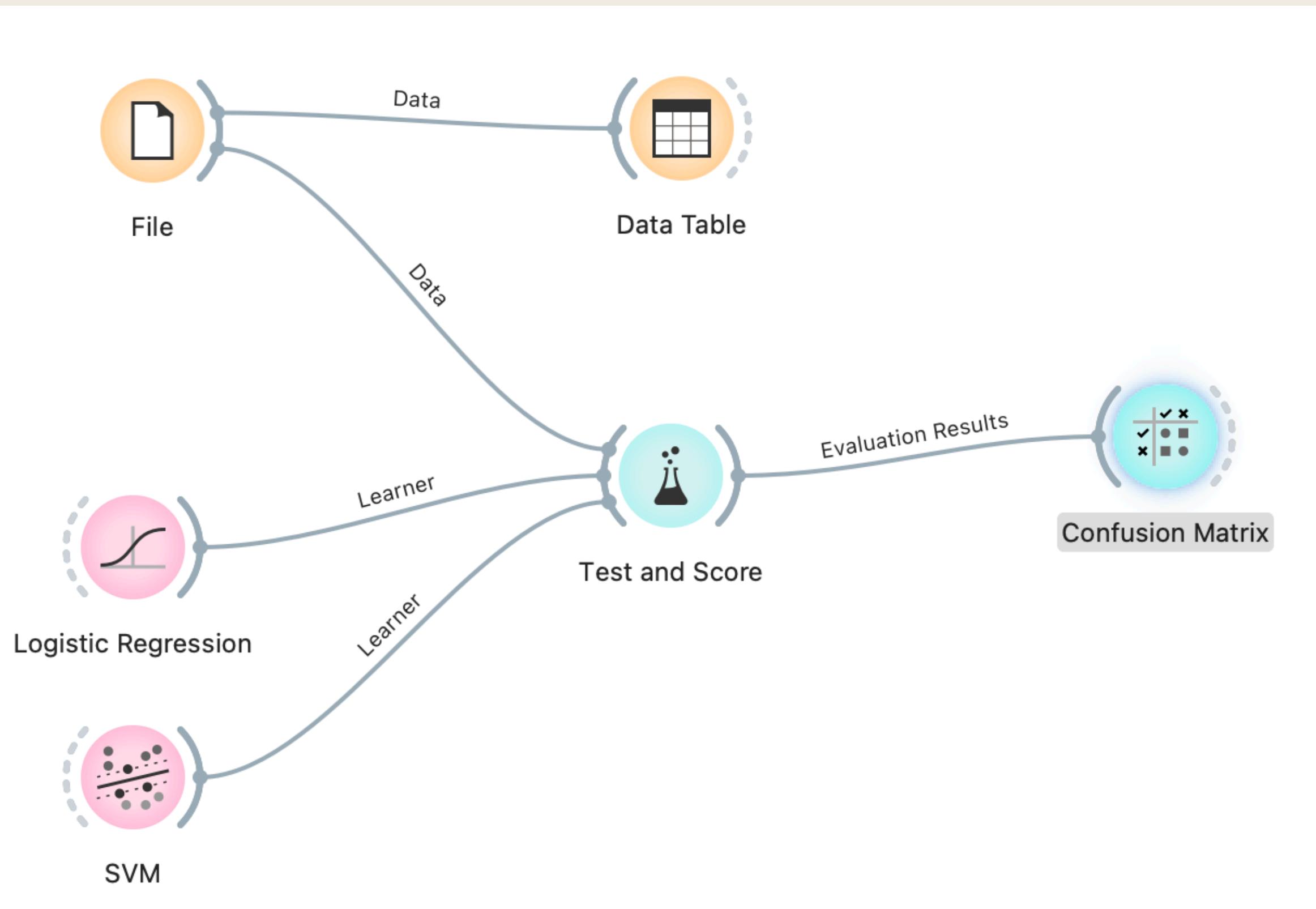
16/MARZO/2022

# Linear models

---



- Linear SVC
- Logistic Regression.



16/MARZO/2022

# Linear Models

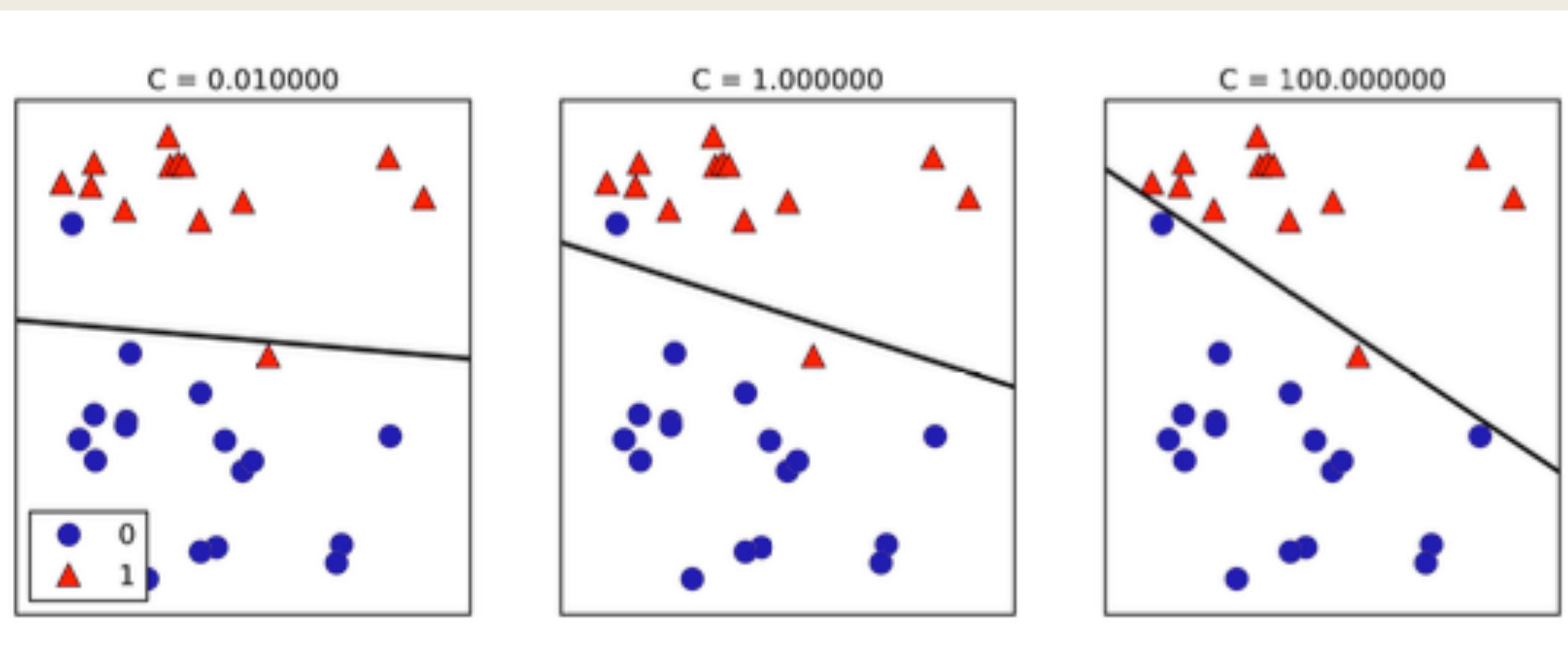
- Datos de Iris
- SVM -> Test and Score -> Confusion M.
- LR -> Test and Score -> Confusion M.
- C values
- Regularization type: L1, L2

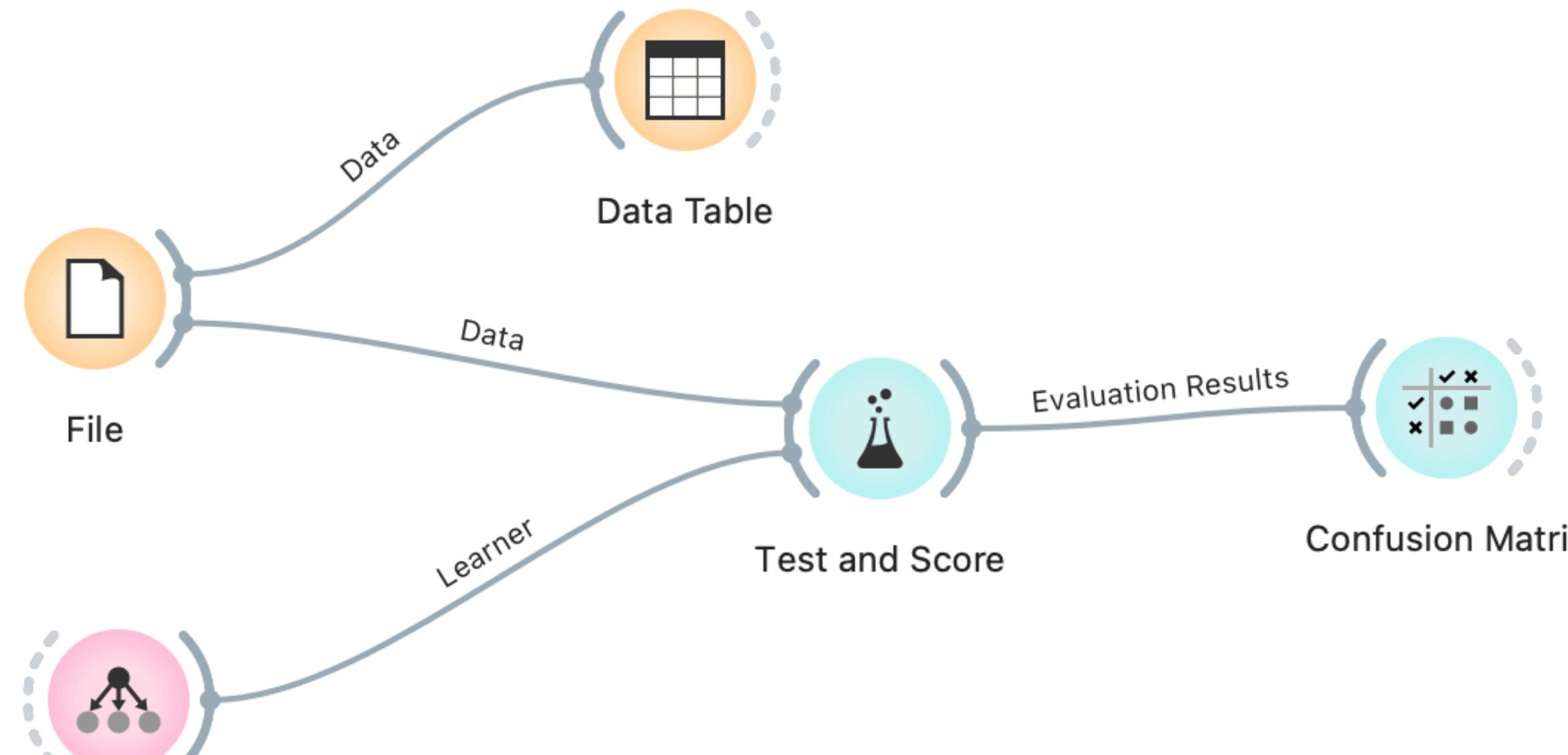
---

16/MARZO/2022

# Linear models

- Studying different C values
- Studying different regularizations.

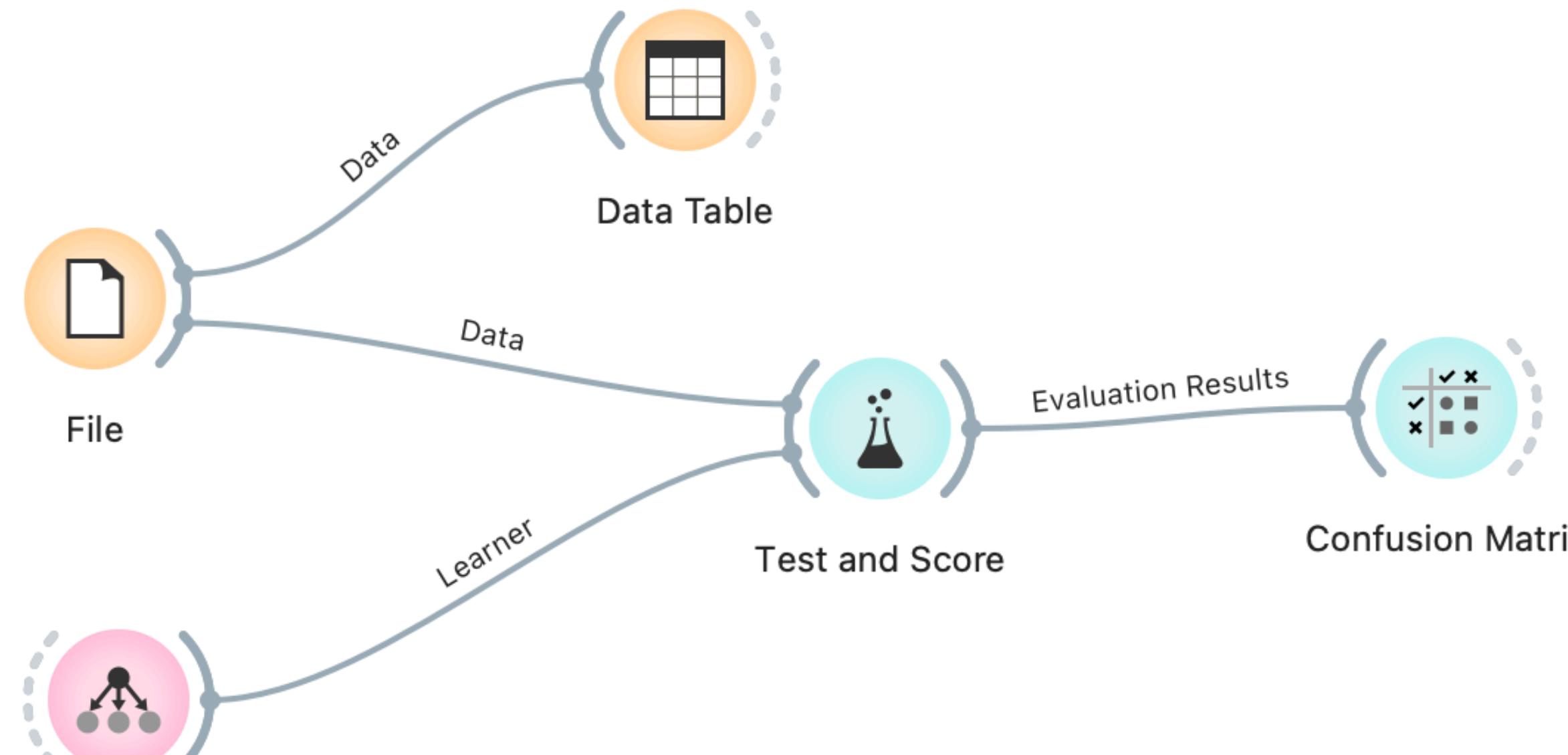




16/MARZO/2022

# Naive Bayes models

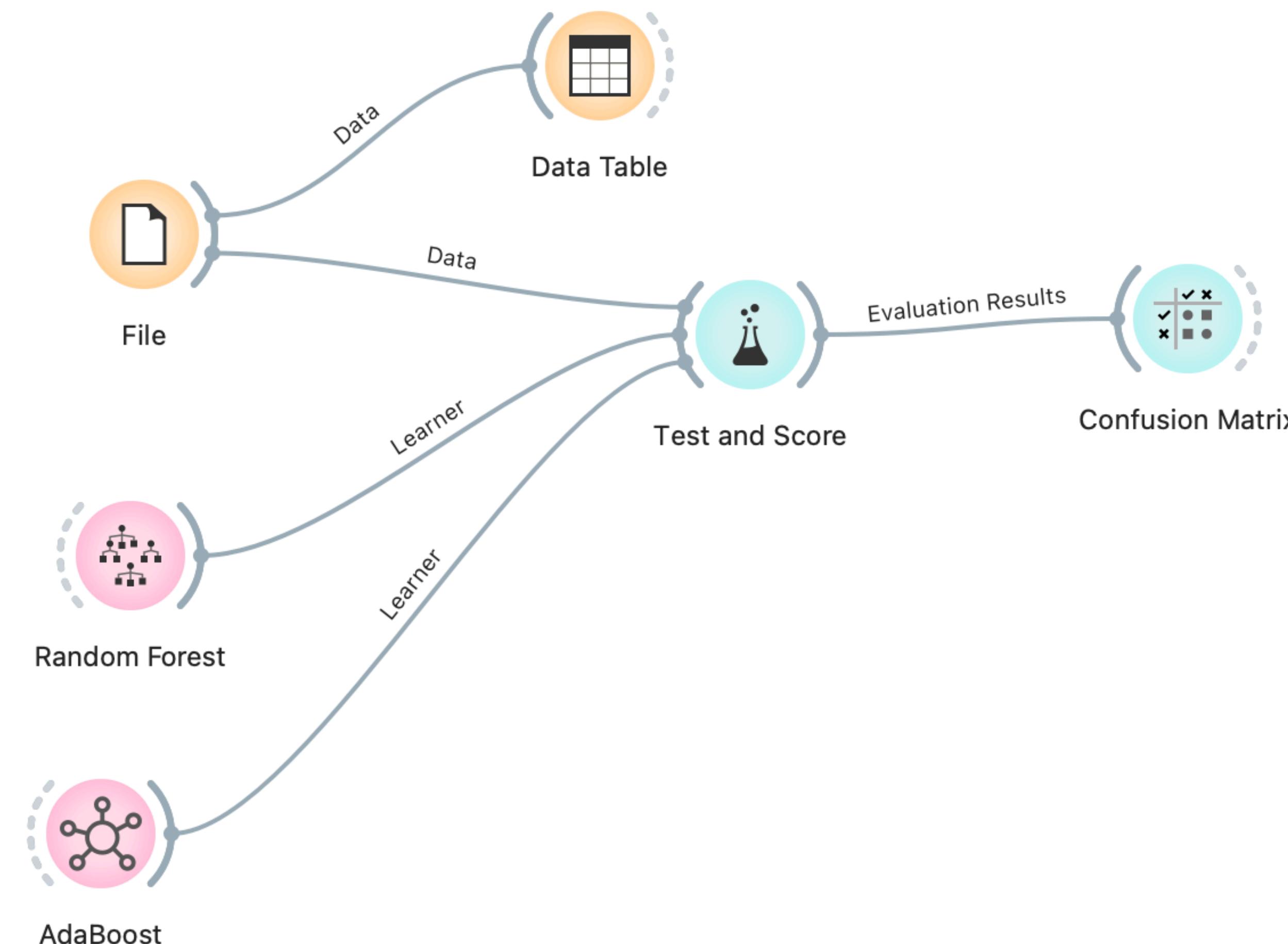
- Naive Bayes classifiers are a family of classifiers that are quite similar to the linear models.
- However, they tend to be even faster in training.
- The price paid for this efficiency is that naive Bayes models often provide generalization performance that is slightly worse than that of linear classifiers



16/MARZO/2022

# Naive Bayes models

- There are three kinds of naive Bayes classifiers: **GaussianNB**, **BernoulliNB**, and **MultinomialNB**.
- GaussianNB can be applied to any continuous data, while BernoulliNB assumes binary data and MultinomialNB assumes count data
- Naive Bayes models are great baseline models and are often used on very large datasets, where training even a linear model might take too long.



16/MARZO/2022

# Ensembles of Decision Trees

- Ensembles are methods that combine multiple machine learning models to create more powerful models.
- There are two ensemble models that have proven to be effective, both of which use decision trees as their building blocks: **random forests** and **gradient boosted decision trees**.

---

# Random Forest

- The main drawback of decision trees is that they **tend to overfit** the training data. Random forests are one way to address this problem.
  - A RF is essentially a **collection of decision trees**, where each tree is slightly different from the others. The idea behind random forests is that each tree might do a relatively good job of predicting, but will likely overfit on part of the data. If we build many trees, all of which work well and overfit in different ways, **we can reduce the amount of overfitting by averaging their results.**
-