DIN Departamento de Informática

UEM Universidade Estadual de Maringá

4086/1 – APRENDIZAGEM DE MÁQUINA

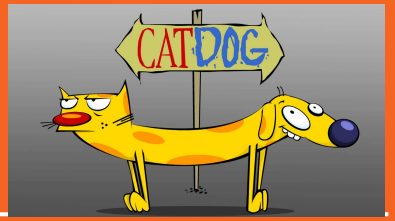
01/2021

Prof. Dra. Valéria Delisandra Feltrim

Doutorado Acadêmicos em Ciência da Computação

Aluno: Jéfer Benedett Dorr

pg54802@uem.br

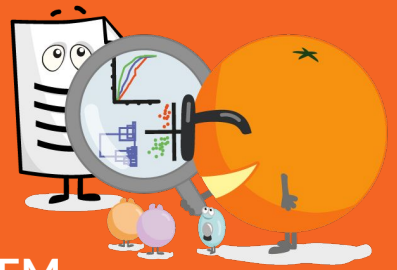


orange

DATA MINING
FRUITFUL&FUN

Ferramenta no-code, produtiva e divertida para ensino lúdico dos conceitos de Aprendizagem de Máquina

Prof. Me. Jéfer Benedett Dörr



- Graduado em Informática pela UNIOESTE
- Especialista em Software Livre pela Unipan
- Mestre em Informática pela UFPR
- Doutorando em Ciência da Computação - UEM

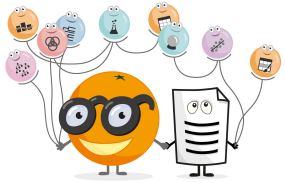
ORANGE is the new BLACK

manna team

UFPR Setor PALOTINA

Docente do Curso de Licenciatura em Computação da UFPR/Palotina

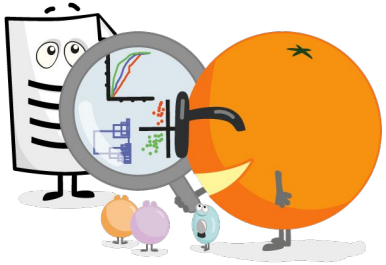
https://orangedatamining.com/



Orange Data Mining - Fruitful & Fun

Atividade prática, mão da massa, de Machine Learning para quem não sabe nada de Inteligência Artificial.
Uso de uma ferramenta gráfica, no coding, onde serão apresentados conceitos e realizados exemplos utilizando alguns dos principais algoritmos de aprendizagem de máquina, como:

- **Naive Bayes**
- **Classificador de Máxima Entropia**
- **Redes neurais**
- **KNN**
- **k-means**
- **SVM**
- **Random Forest**



links para acompanhar

colab - fofo

dataset principal

pasta

modelo teachable machine

colab - teachable machine

colab - cat vs dog

colab - fofo

Machine Learning é uma técnica de Inteligência Artificial que permite que a máquina aprenda através de exemplos

Exemplo: PROBLEMA identificar o que é um gato e por eliminação, se o bichinho não for um gato, vamos identificá-lo como cachorro.

cat	fofo?	orelha pequena?	mia?
cat	1	1	1
cat	1	0	1
cat	0	1	1

características do nosso gato referência:

- É fofo?
- Tem orelhinha pequena?
- Faz miau?

dog	fofo?	orelha pequena?	mia?
dog	1	1	0
dog	0	1	0
dog	0	1	0

#Dataset:

bichinho1 = [1, 1, 1]

bichinho2 = [1, 0, 1]

bichinho3 = [0, 1, 1]

bichinho4 = [1, 1, 0]

bichinho5 = [0, 1, 0]

bichinho6 = [0, 1, 0]

#label

#1 = Gato

#-1 = Cachorro

marcacoes = [1, 1, 1, -1, -1, -1]

dados = [bichinho1, bichinho2, bichinho3, bichinho4, bichinho5, bichinho6]

https://colab.research.google.com/drive/1-SHLlksDt2E0j59_UUGWIFTFsJKbuukH?usp=sharing

Create model - Naive Bayes

```
!pip install scikit-learn
from sklearn.naive_bayes import MultinomialNB


modelo = MultinomialNB()
modelo.fit(dados,marcacoes)
```

Naive-Bayes: calcular a probabilidade que uma amostra desconhecida pertença a cada uma das classes possíveis, ou seja, predizer (ou adivinhar) a classe mais provável.

Predict

```
bicho_misterioso1 = [1, 1, 1]
bicho_misterioso2 = [1, 0, 0]
bicho_misterioso3 = [0, 0, 1]
```

-	fofo?	orelha pequena?	mia?
?	1	1	1
?	1	0	0
?	0	0	1

Resultados

```
teste = [bicho_misterioso1, bicho_misterioso2,
bicho_misterioso3]
resultado = modelo.predict(teste)
print(resultado)
```

Correto?

```
marcacoes_teste = [1,-1, 1]
print('Resultado: ')
print(resultado)
print ('Marcacoes: ')
print(marcacoes_teste)
```

Resultado[Gato, Gato, Gato]

e a resposta correta deveria ser

Marcacoes [Gato,Cachorro,Gato]

algoritmo tem a acurácia de 66,66%, ou seja,

ele acertou 2 bichos misteriosos do nosso total de 3.

curiosidade

O algoritmo “**Naive Bayes**” é um classificador probabilístico baseado no “**Teorema de Bayes**”, o qual foi criado por **Thomas Bayes** (1701 - 1761) para tentar provar a existência de Deus.

Hoje, usado para identificar se determinado e-mail é um **SPAM!**

```
# Criar um classificador
nbayes = NBayes::Base.new
# Treinar o classificador com exemplos - as palavras da String são
#divididas em um array
nbayes.train( "You need to buy some Viagra".split(/\s+/), 'SPAM' )
nbayes.train( "This is not spam, just a letter to Bob.".split(/\s+/), 'HAM' )
nbayes.train( "Hey Oasic, Do you offer consulting?".split(/\s+/), 'HAM' )
nbayes.train( "You should buy this stock".split(/\s+/), 'SPAM' )

# Dividir mensagem que precisa ser classificada
tokens = "Now is the time to buy Viagra cheaply and
discreetly".split(/\s+/)
result = @nbayes.classify(tokens)
# Imprime a classe em que o texto foi classificado. (SPAM ou HAM)
p result.max_class
# Imprime a probabilidade da mensagem ser SPAM
p result['SPAM']
# Imprime a probabilidade da mensagem ser HAM
p result['HAM']
```

exemplo

- 100 pessoas realizaram o teste.
- 20% das pessoas que realizaram o teste possuíam a doença.
- 90% das pessoas que possuíam a doença, receberam positivo no teste.
- 30% das pessoas que não possuíam a doença, receberam positivo no teste.

Se uma nova pessoa realizar o teste e receber um resultado positivo, qual a probabilidade de ela possuir a doença?

O algoritmo de **Naive Bayes** consiste em encontrar uma probabilidade a posteriori (parte do efeito para a causa, ou seja, possuir a doença, dado que recebeu um resultado positivo), multiplicando a probabilidade a priori (da causa para o efeito, ou, possuir a doença) pela probabilidade de "receber um resultado positivo, dado que tem a doença".

Devemos também computar a probabilidade a posteriori da negação (Não possuir a doença, dado que recebeu um resultado Positivo).

P(doença|positivo) = 20% * 90% = 0,2*0,9=0,18
P(doença|negativo) = 80% * 30% = 0,8*0,3=0,24
normalizando:

P(doença|positivo) = 0,18/(0,18+0,24) = 0,4285

P(não doença|positivo) = 0,24/(0,18+0,24) = 0,5714

se o resultado do teste da nova pessoa for positivo, ela possui aproximadamente 43% (0,4285) de chance de estar doente.

Naive Bayes - titanic disaster (base disponivel no Orange)

- Orange is a component-based visual programming software package for data visualization, machine learning, data mining, and data analysis.

- open-source software GPL

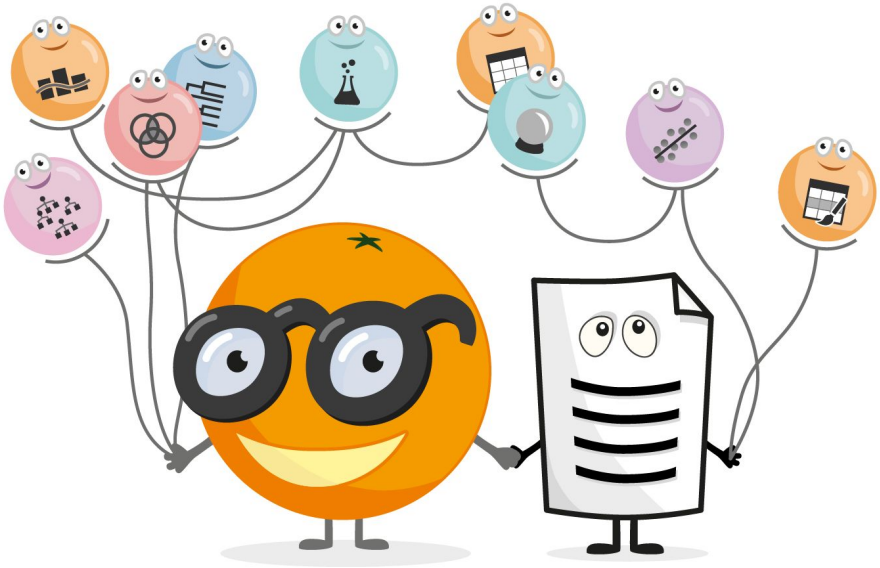
- Python
- scientific computing: numpy, scipy and scikit-learn
- interface: Qt framework

- *Slovenia*

- Initial release: 1996
- Last release: 2020

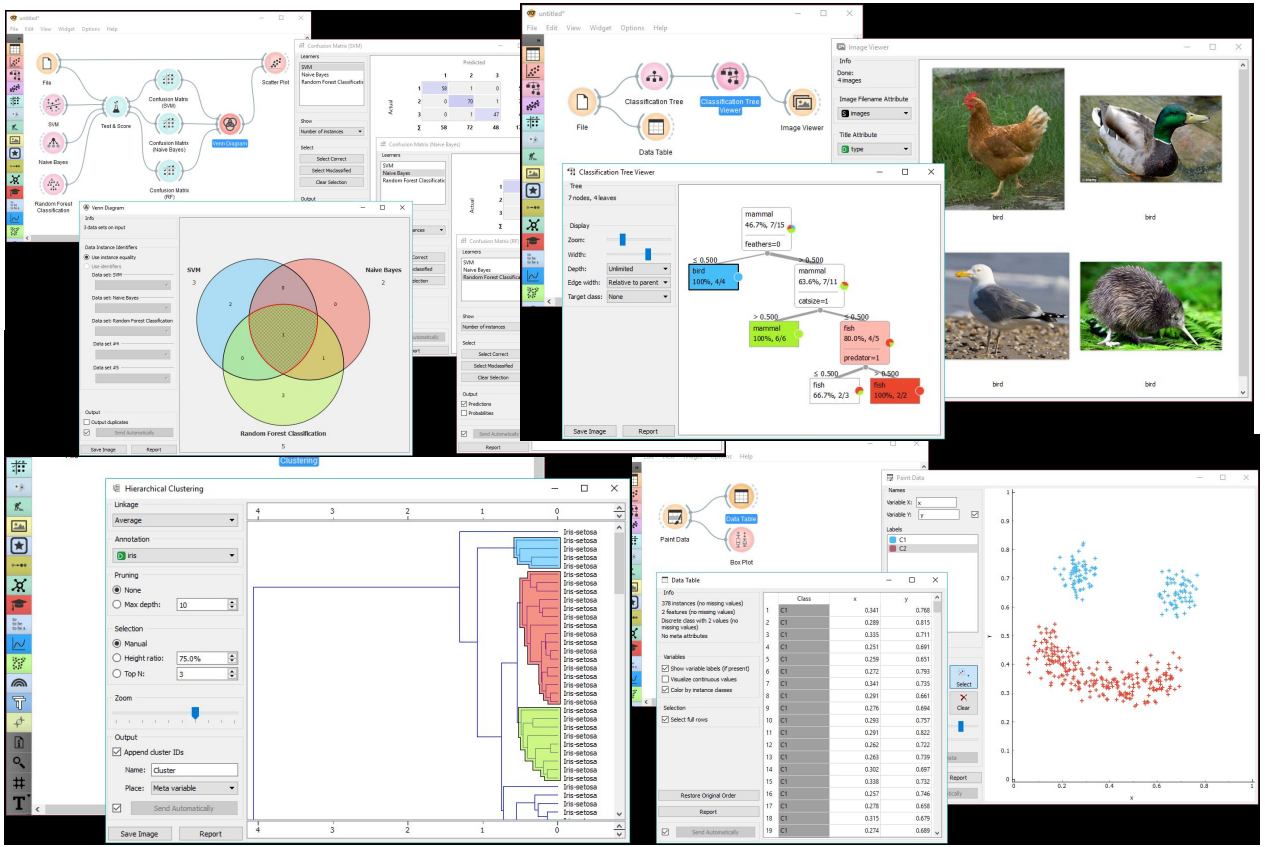
- drag and drop

ORANGE is the new BLACK



https://orangedatamining.com/

images



https://orangedatamining.com/screenshots/

blog

It's Sailing Time (Again)

By: BLAZ, Aug 11, 2017

Every fall I teach a course on Introduction to Data Mining. And while the course is really on statistical learning and its applications, I also venture into classification trees. For several reasons. First, I can introduce information gain and with it feature scoring and ranking. Second, classificatio trees are one of the first machine learning approaches co-invented by engineers (Ross Quinlan) an statisticians (Leo Breiman, Jerome Friedman, Charles J. Stone, Richard A. Olshen). And finally, because they make the base of random forests, one of the most accurate machine learning models for smaller and mid-size data sets.

Related: Introduction to Data Mining Course in Houston

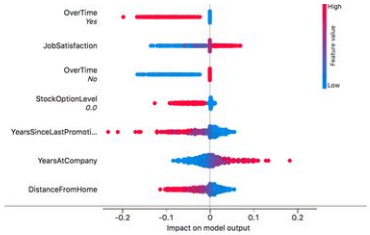
Lecture on classification trees has to start with the data. Years back I have crafted a data set on sailing. Every data set has to have a story. Here is one:

> Sara likes weekend sailing. Though, not under any condition. Past twenty Wednesdays I have asked her if she will have any company, what kind of boat she can rent, and I have checked the weather forecast. Then, on Saturday, I wrote down if she actually went to the Sea.

Data on Sara's sailing contains three attributes (Outlook, Company, Sailboat) and a class (Sail).

Data Table

	Sail	Outlook	Company	Sailboat
1	yes	rainy	big	big
2	yes	rainy	big	small
3	no	rainy	med	big
4	no	rainy	med	small
5	yes	sunny	big	big
6	yes	sunny	big	small
7	yes	sunny	med	big



By: Ajda Pretnar, Feb 10, 2021

Explaining Predictive Models

New Orange Explain add-on for understanding predictions and predictive models.

Categories: model explanation | explainable AI | explain | predictive modelling

Read More

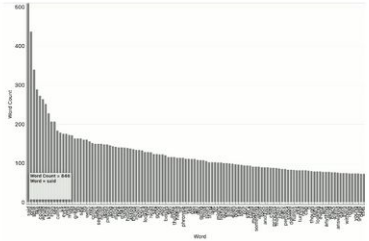
Ajda Pretnar, Jan 27, 2021

bserving Word Distribution

w to inspect word distribution in a corpus with a clever combination of widgets Orange.

tegories: text mining | word distribution | bar plot | word cloud

ad More



By: Ajda Pretnar, Jan 11, 2021

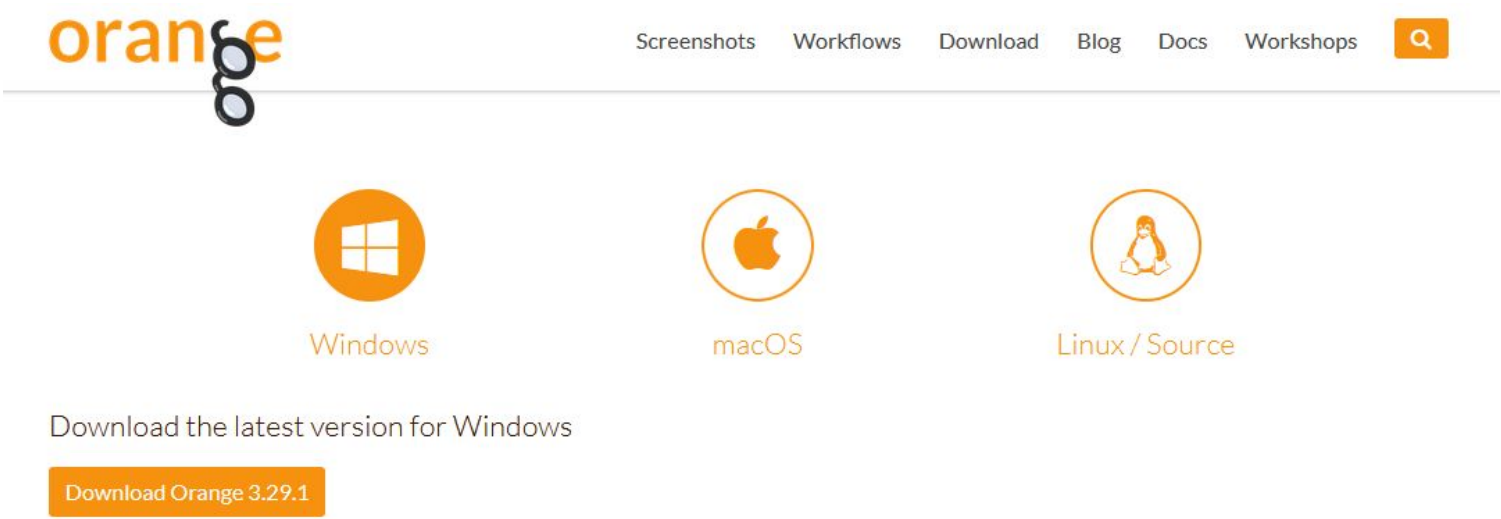
Orange in Classroom

Orange is actively used in classrooms at over two hundred universities from around the world.

Categories: Orange | education | teaching | university

https://orangedatamining.com/blog

download



https://orangedatamining.com/download/

Install

Windows: double click - next - next - ok

Linux:

sudo apt install virtualenv build-essential python3-dev

virtualenv --python=python3 --system-site-packages orange3venv

pip install PyQt5 PyQtWebEngine

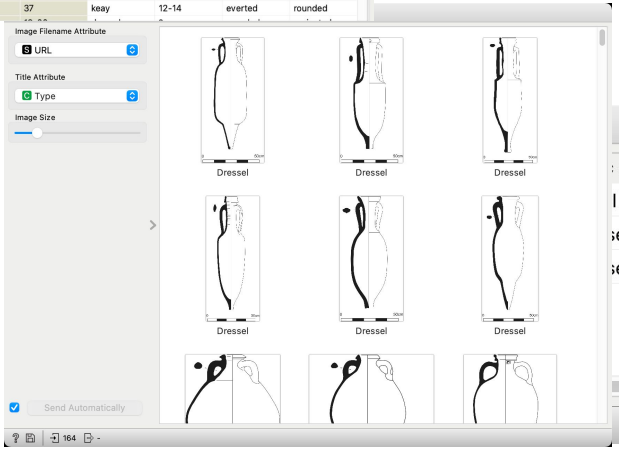
source bin/activate

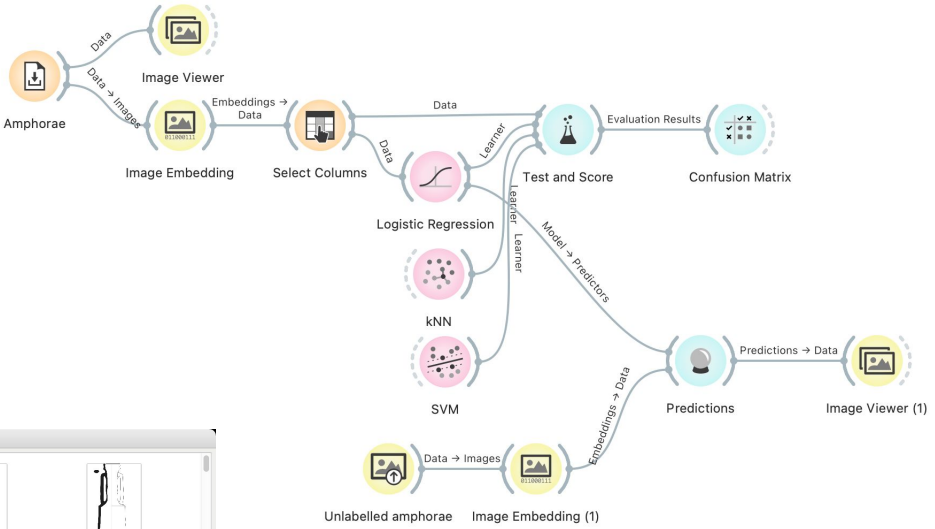
pip install orange3

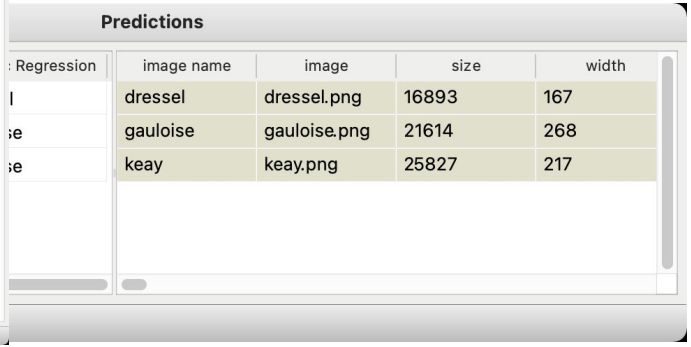
anaconda


ANACONDA NAVIGATOR
Sign in
Home
Environments
Projects (beta)
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
anaconda
Channels
glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Launch
notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis
Launch
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Launch
qtconsole
4.3.0
Launch
rstudio
1.1.383
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Launch
spyder
3.1.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
jupyterlab
0.31.8
Install

examples

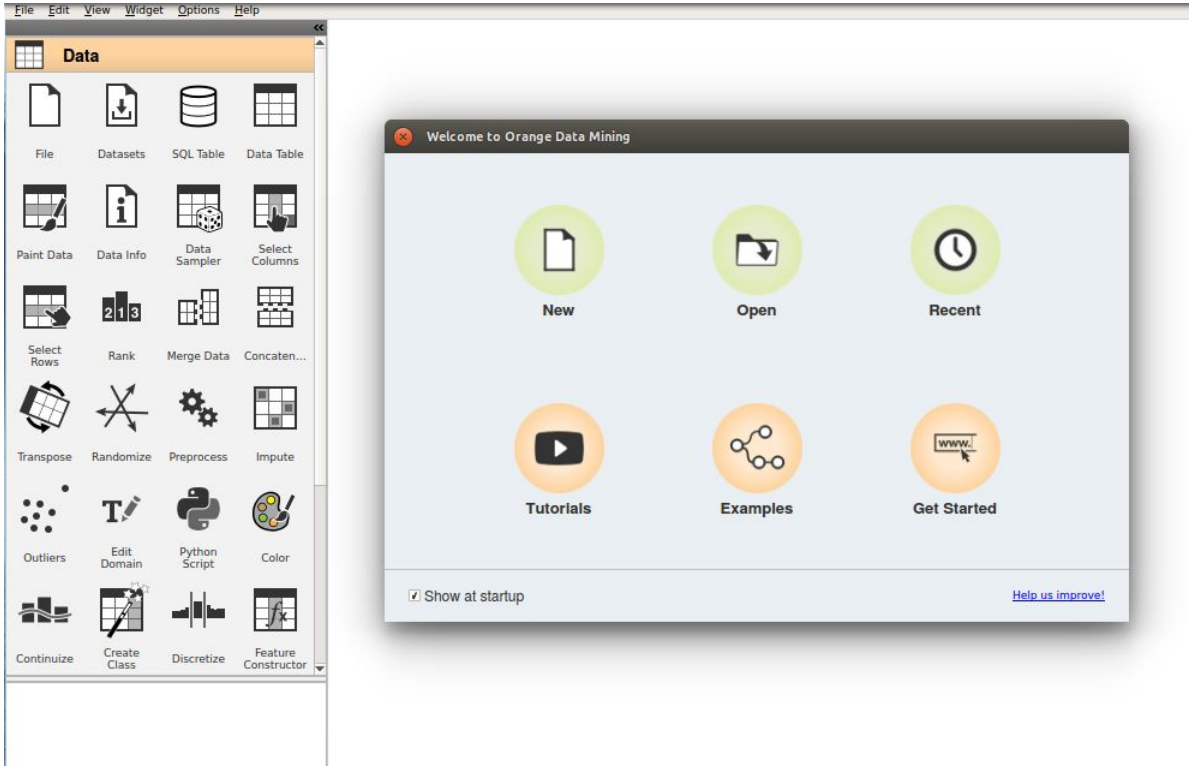




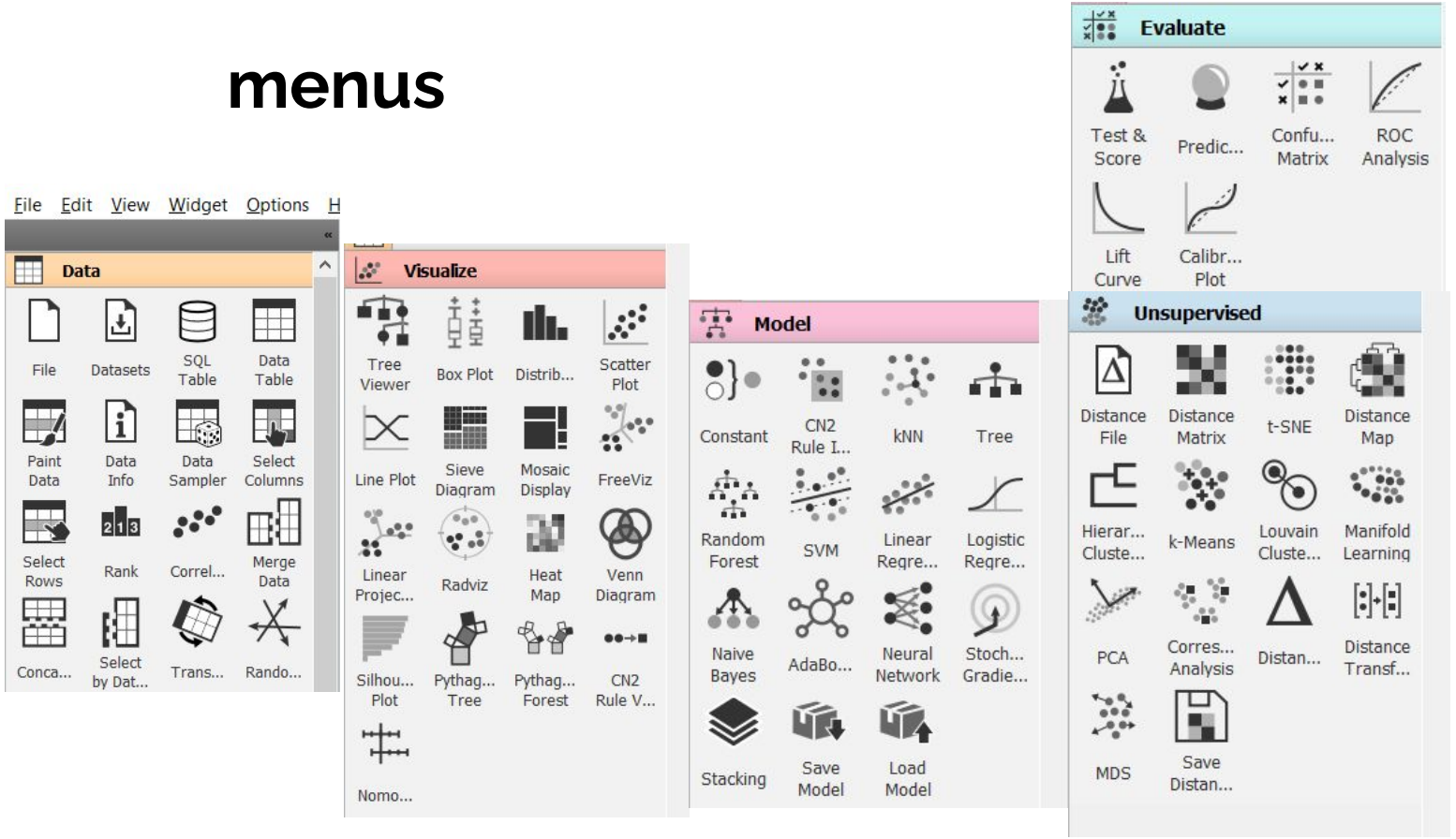


https://orangedatamining.com/blog/2021/2021-04-23-archaeology-workshop/

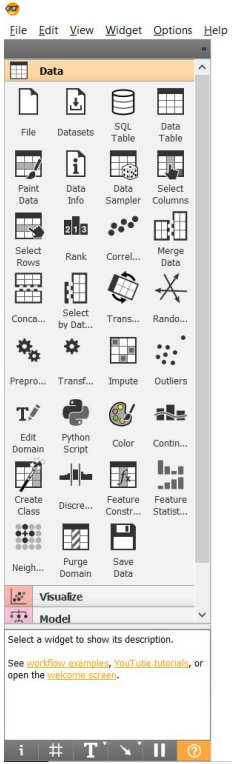
use

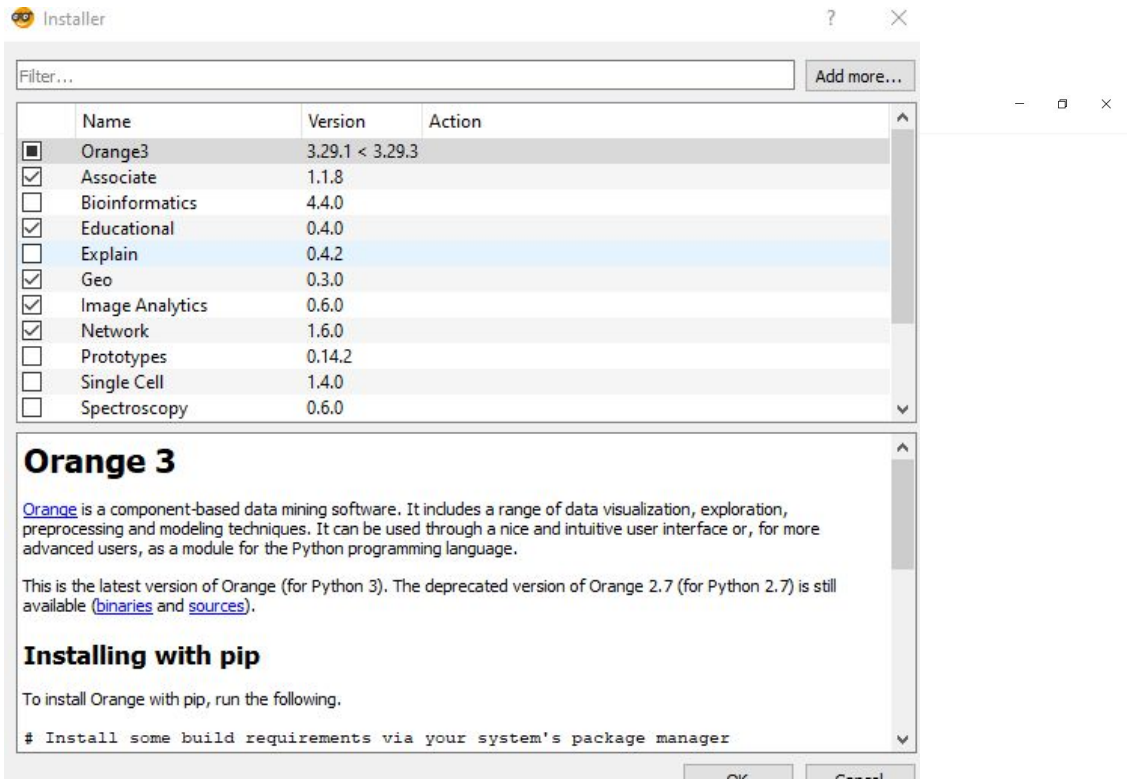

File Edit View Widget Options Help
Data
File
Datasets
SQL Table
Data Table
Paint Data
Data Info
Data Sampler
Select Columns
Select Rows
Rank
Merge Data
Concaten...
Transpose
Randomize
Preprocess
Impute
Outliers
Edit Domain
Python Script
Color
Continuize
Create Class
Discretize
Feature Constructor
Welcome to Orange Data Mining
New
Open
Recent
Tutorials
Examples
Get Started
Show at startup
Help us improve!

menus

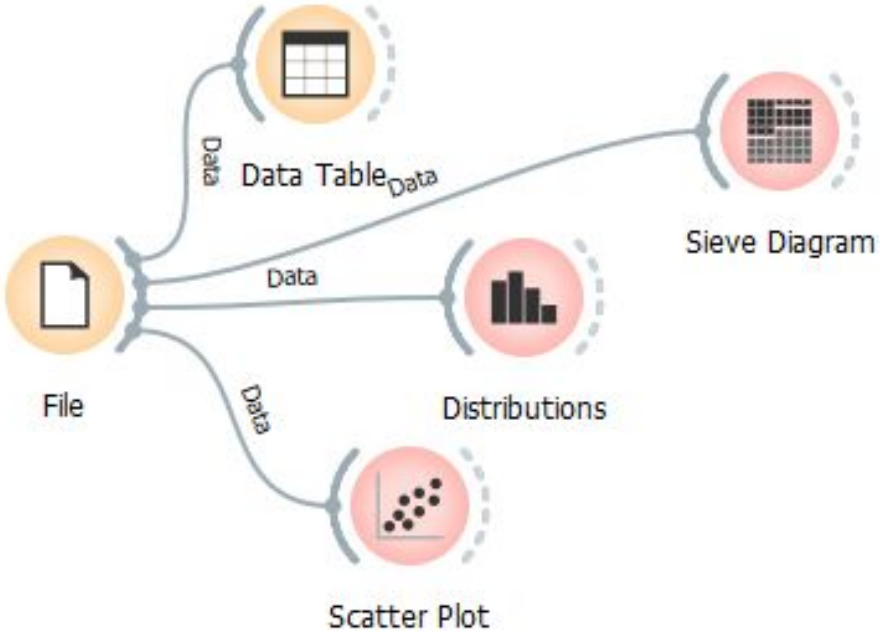

File Edit View Widget Options H
Data
File
Datasets
SQL Table
Data Table
Paint Data
Data Info
Data Sampler
Select Columns
Select Rows
Rank
Correl...
Merge Data
Conca...
Select by Dat...
Trans...
Rando...
Visualize
Tree Viewer
Box Plot
Distrib...
Scatter Plot
Line Plot
Sieve Diagram
Mosaic Display
FreeViz
Linear Projec...
Radviz
Heat Map
Venn Diagram
Silhou... Plot
Pythag... Tree
Pythag... Forest
CN2 Rule V...
Nomo...
Model
Constant
CN2 Rule I...
kNN
Tree
Random Forest
SVM
Linear Regre...
Logistic Regre...
Naive Bayes
AdaBo...
Neural Network
Stoch... Gradie...
Stacking
Save Model
Load Model
Evaluate
Test & Score
Predic...
Confu... Matrix
ROC Analysis
Lift Curve
Calibr... Plot
Unsupervised
Distance File
Distance Matrix
t-SNE
Distance Map
Hierar... Cluste...
k-Means
Louvain Cluste...
Manifold Learning
PCA
Corres... Analysis
Distan...
Distance Transf...
MDS
Save Distan...

addons

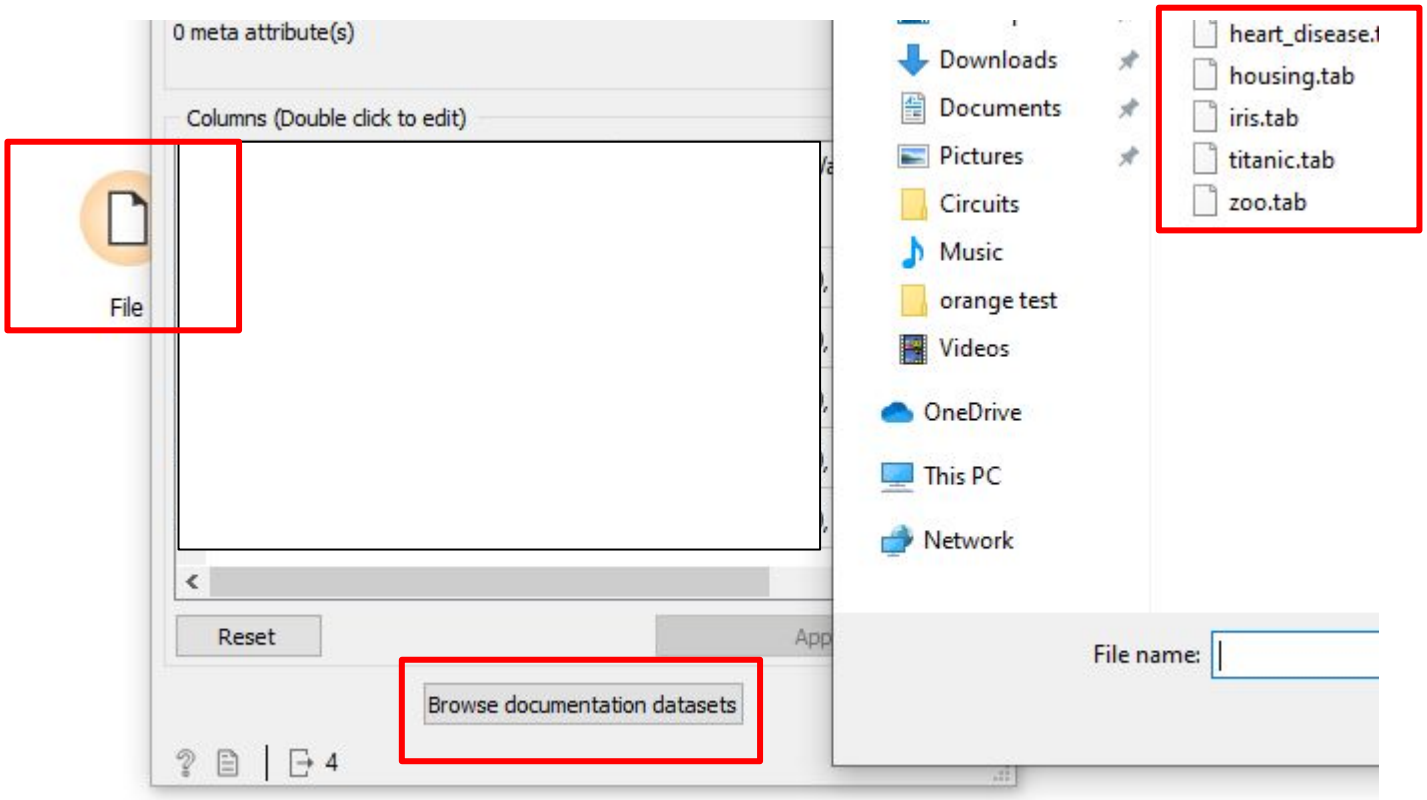

File Edit View Widget Options Help
Data
File
Datasets
SQL Table
Data Table
Paint Data
Data Info
Data Sampler
Select Columns
Select Rows
Rank
Correl...
Merge Data
Conca...
Select by Dat...
Trans...
Rando...
Prepro...
Transf...
Impute
Outliers
Edit Domain
Python Script
Color
Contin...
Create Class
Discre...
Feature Constr...
Feature Statist...
Neigh...
Purge Domain
Save Data
Visualize
Model
Select a widget to show its description.
See workflow examples, YouTube tutorials, or open the welcome screen.


Installer
Filter...
Add more...
Name
Version
Action
Orange3 3.29.1 < 3.29.3
Associate 1.1.8
Bioinformatics 4.4.0
Educational 0.4.0
Explain 0.4.2
Geo 0.3.0
Image Analytics 0.6.0
Network 1.6.0
Prototypes 0.14.2
Single Cell 1.4.0
Spectroscopy 0.6.0
Orange 3
Orange is a component-based data mining software. It includes a range of data visualization, exploration, preprocessing and modeling techniques. It can be used through a nice and intuitive user interface or, for more advanced users, as a module for the Python programming language.
This is the latest version of Orange (for Python 3). The deprecated version of Orange 2.7 (for Python 2.7) is still available (binaries and sources).
Installing with pip
To install Orange with pip, run the following.
Install some build requirements via your system's package manager

basic use


Data
Data Table
Data
Sieve Diagram
Data
File
Distributions
Data
Scatter Plot

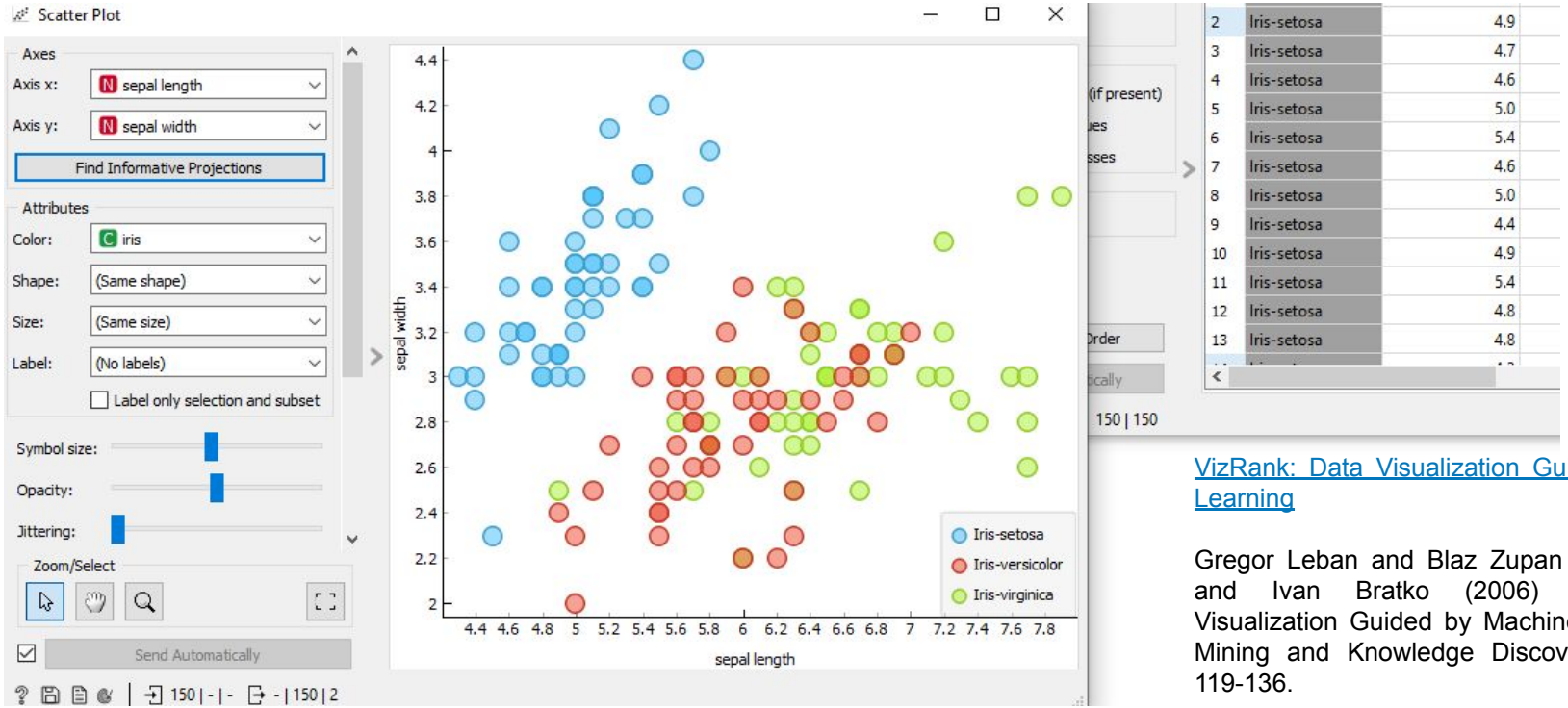
documentation datasets



Scatter plot visualization with explorative analysis and intelligent data visualization enhancements.

Scatter-Plot - Gráfico de dispersão

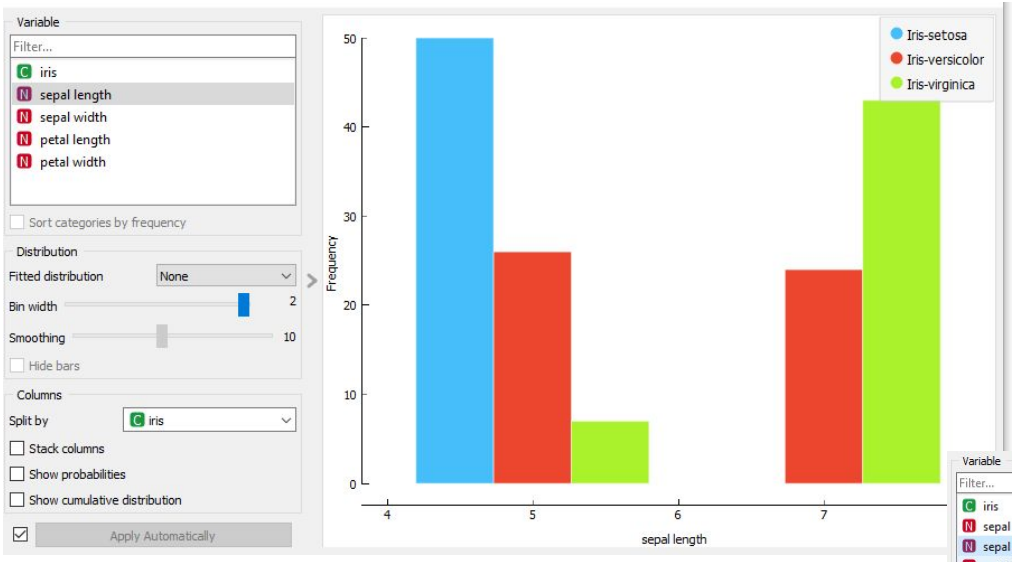
https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/visualize/scatterplot.html

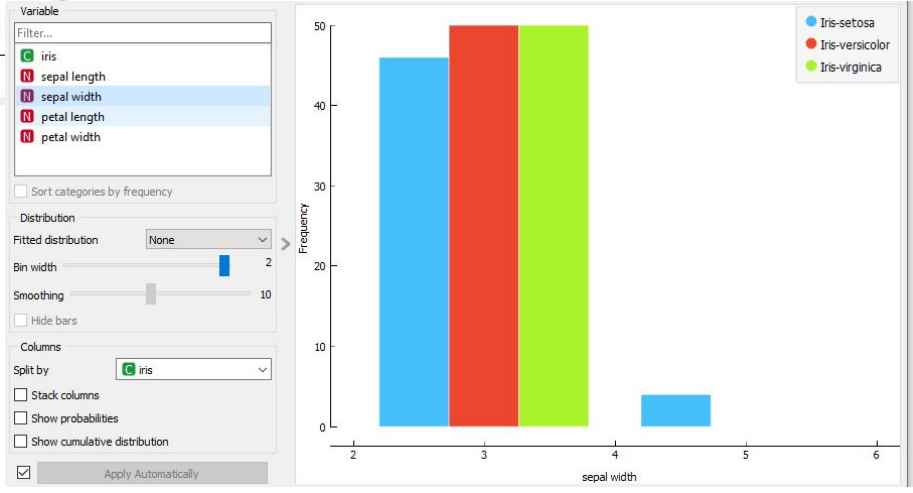


[VizRank: Data Visualization Guided by Machine Learning]

Gregor Leban and Blaz Zupan and Gaj Vidmar and Ivan Bratko (2006) VizRank: Data Visualization Guided by Machine Learning. Data Mining and Knowledge Discovery, 13 (2). pp. 119-136.

Distributions

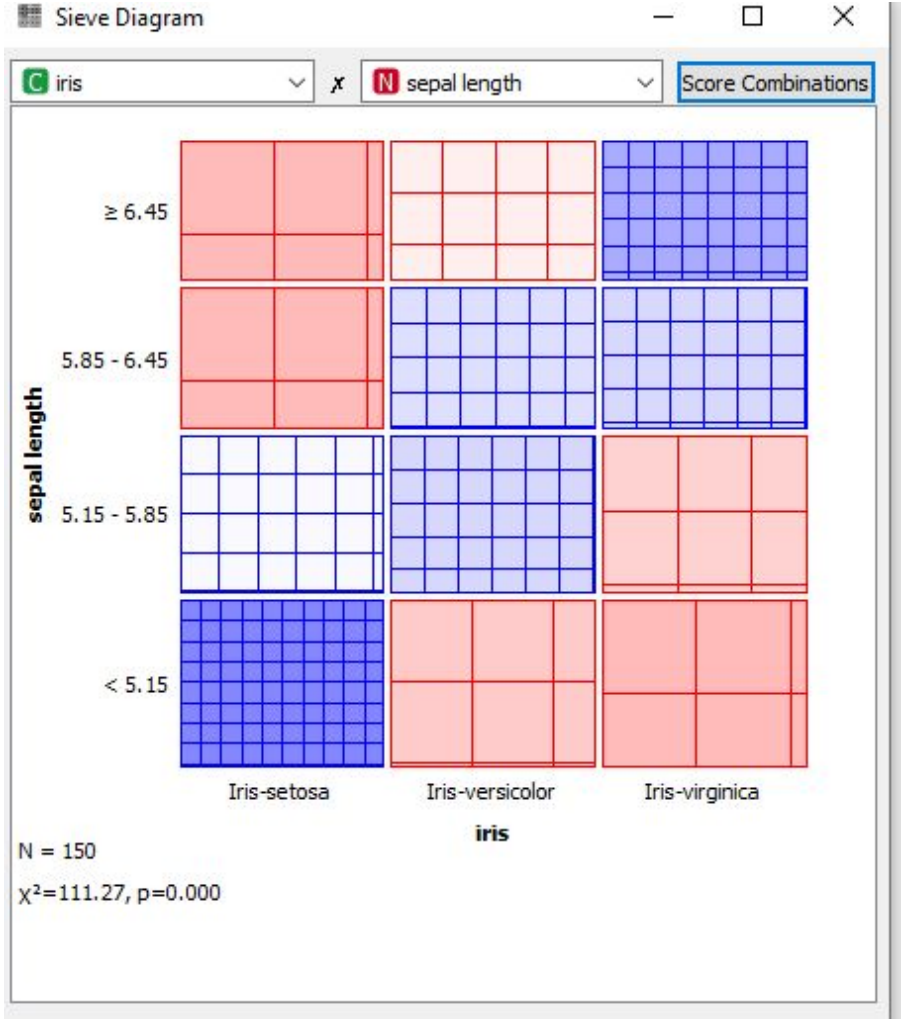




Sieve Diagram

Sieve Diagram summarizes the relationship between the categorical variables using frequencies

It was proposed by Riedwyl and Schüpbach in a technical report in 1983 and later called a parquet diagram (Riedwyl and Schüpbach 1994).



end basic

atividade 1

Atividade 1

monet or manet - inceptionv3 (google)

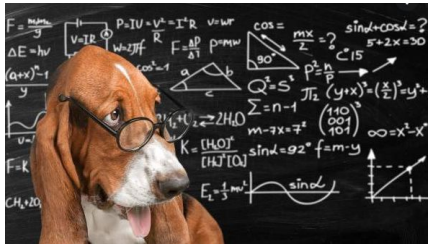
the lost monet - model pre trained painters

end atividade 1

Machine Learning: cat or dog?

dataset

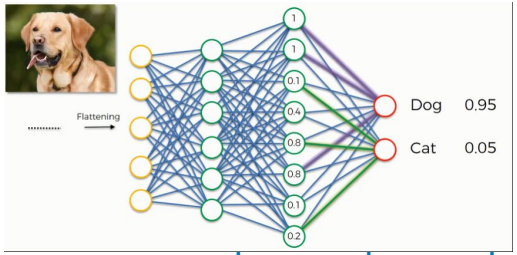
https://www.kaggle.com/tongpython/cat-and-dog



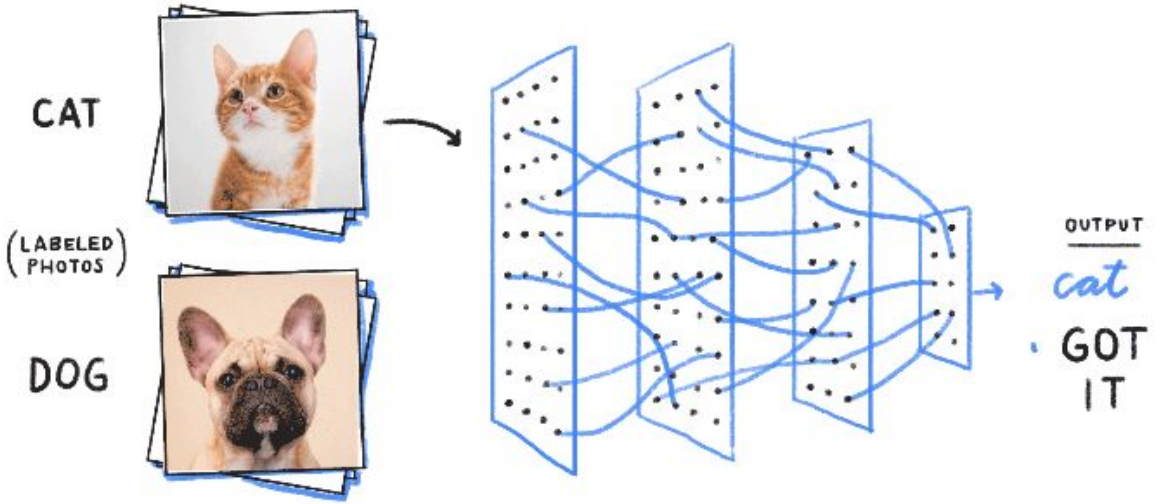
kaggle

This dataset is for running the code from this site: https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8.

solve an image classification problem

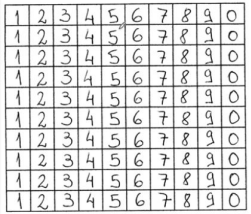


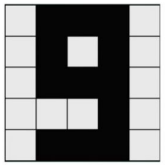
[como uma rede neural aprende]



training an artificial neural network on few thousand images of cats and dogs and make the Neural Network learn to predict which class the image belongs to, next time it sees an image having a cat or dog in it.

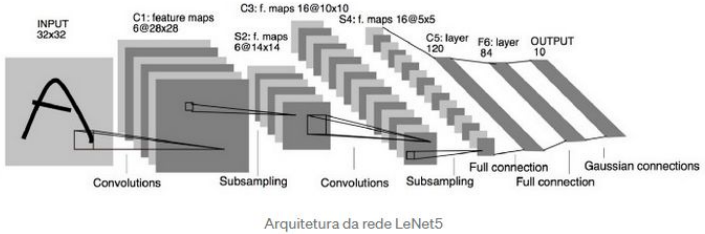
rede neural



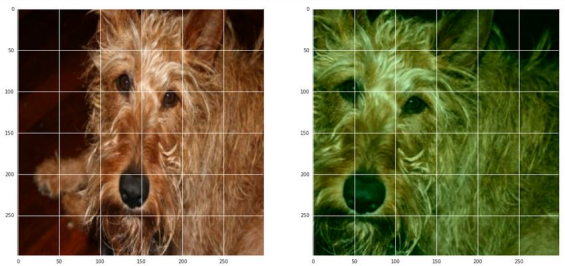


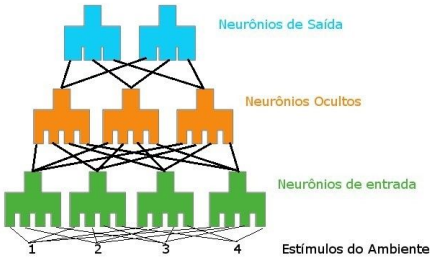
0,1,1,1,0,0,1,0,1,0,0,1,1,1,0,0,0,0,1,0,0,1,1,1,0,9

primeira aplicação com sucesso de uma CNN foi desenvolvida por [Yann LeCun em 1998]

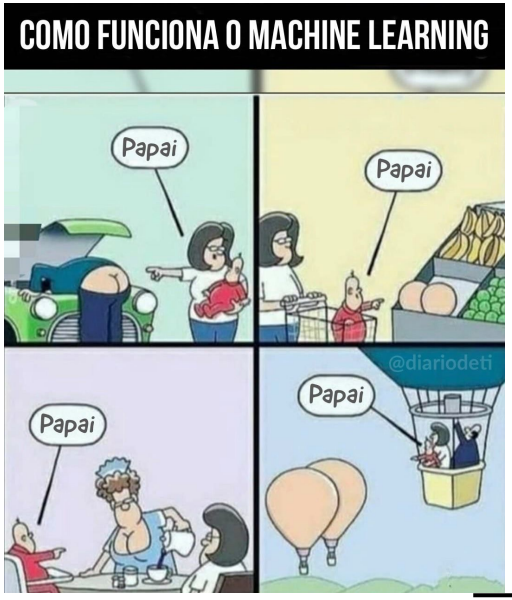


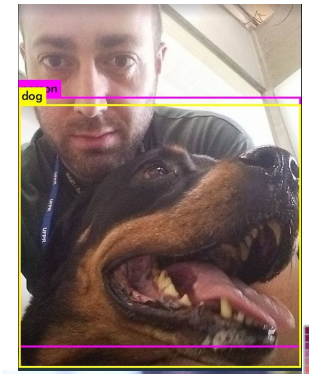
Arquitetura da rede LeNet5

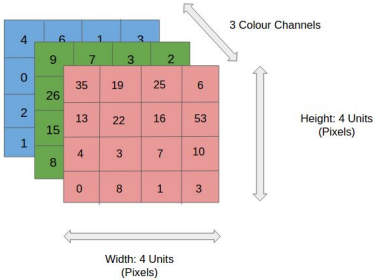


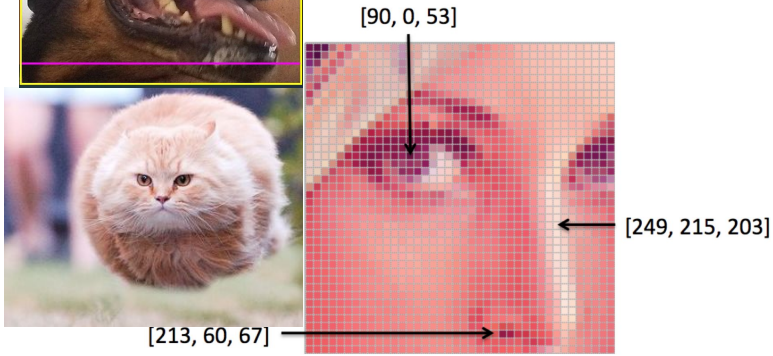


colored images - rgb



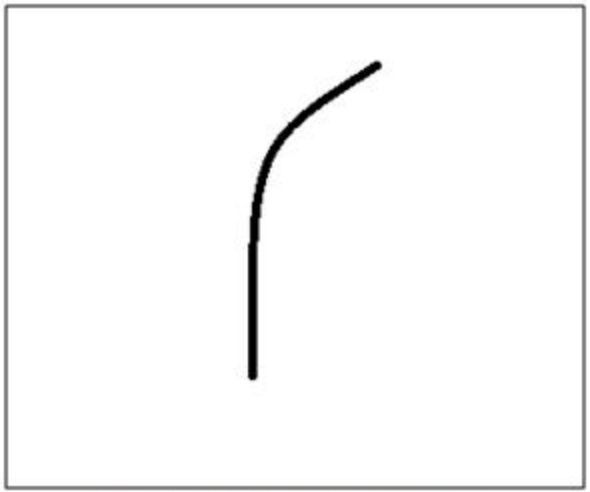




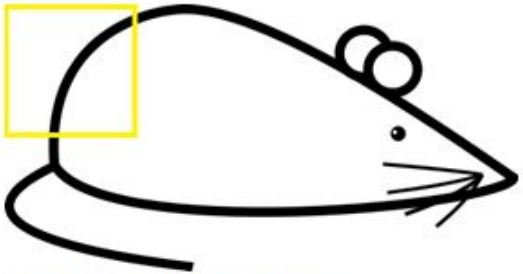


0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

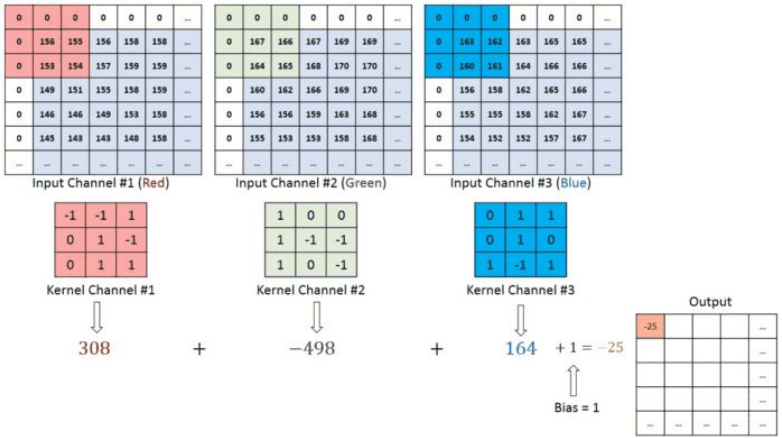


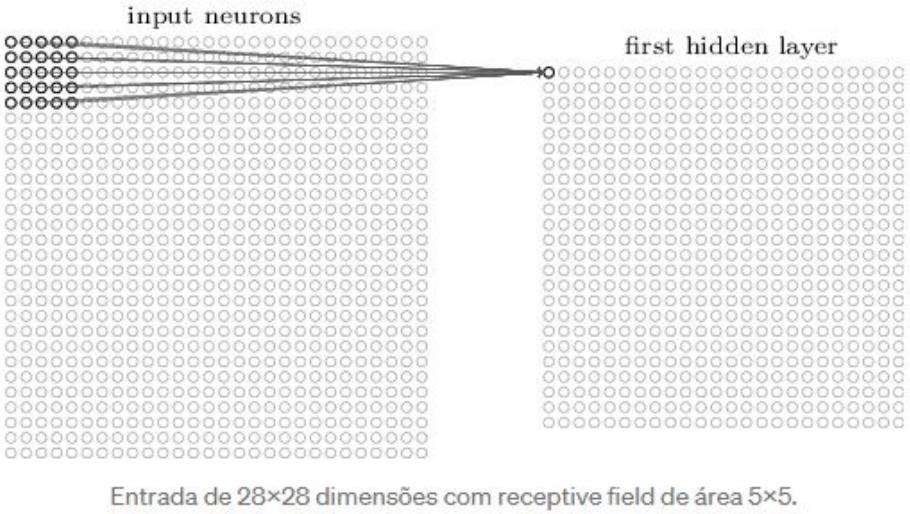
Visualization of the filter on the image

Convoluções

filtros = kernel formados por pesos e atualizados por backpropagation

https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184





Entrada de 28×28 dimensões com receptive field de área 5×5.

1 - The input image is the image being detected. The feature detector is a matrix, usually 3x3. A feature detector is also referred to as a kernel or a filter.

CNN (Convolutional Neural Network)

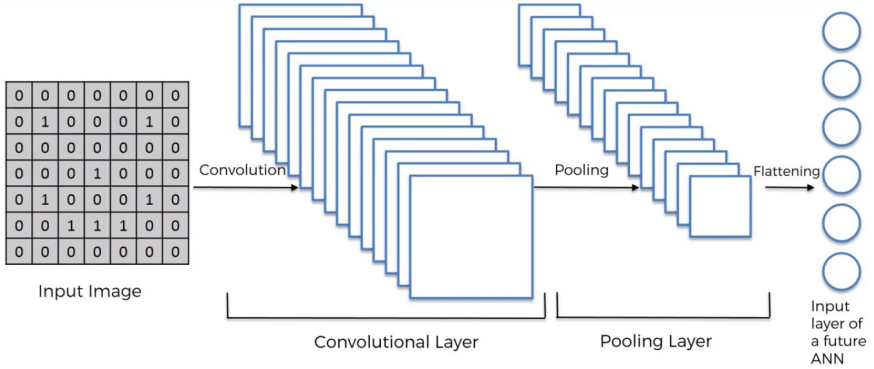
The process of building a Convolutional Neural Network always involves four major steps.

Step - 1 : Convolution

Step - 2 : Pooling

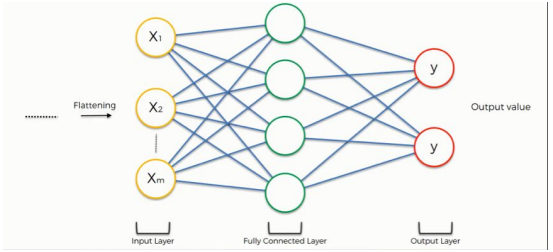
Step - 3 : Flattening

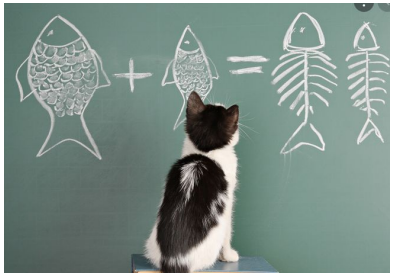
Step - 4 : Full connection



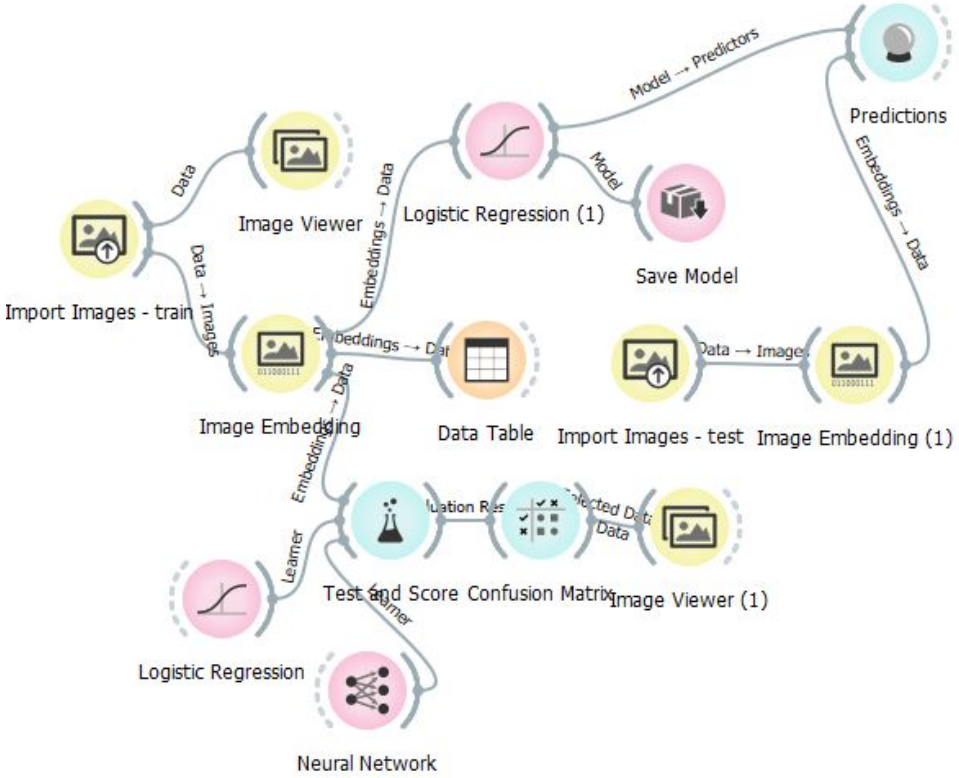
compile and Train model

new predictions from trained model

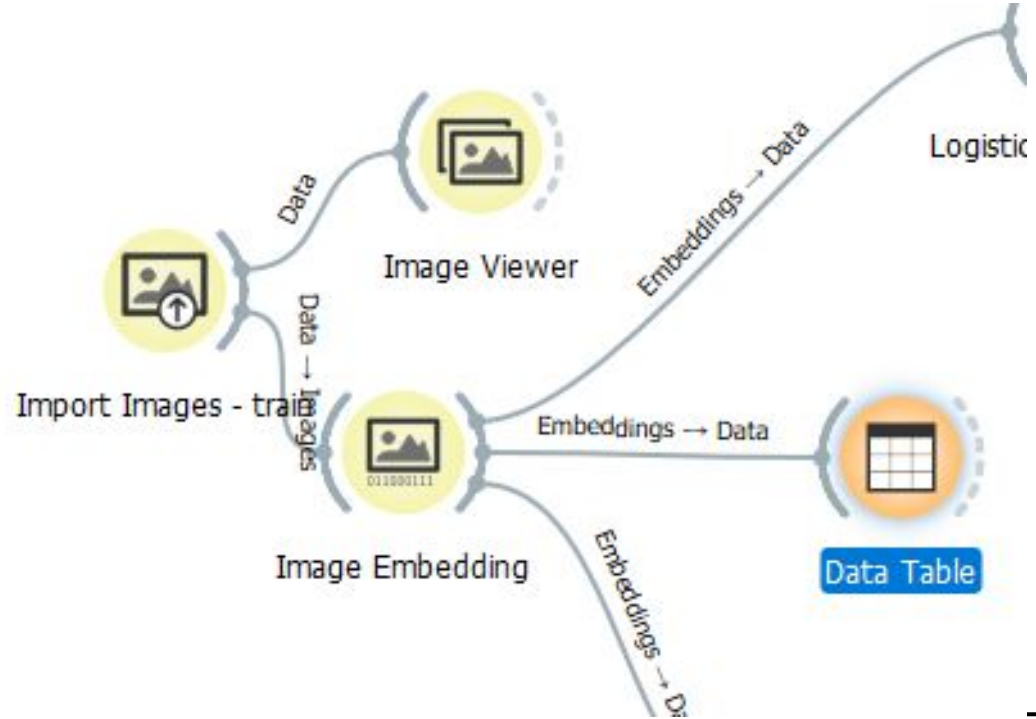




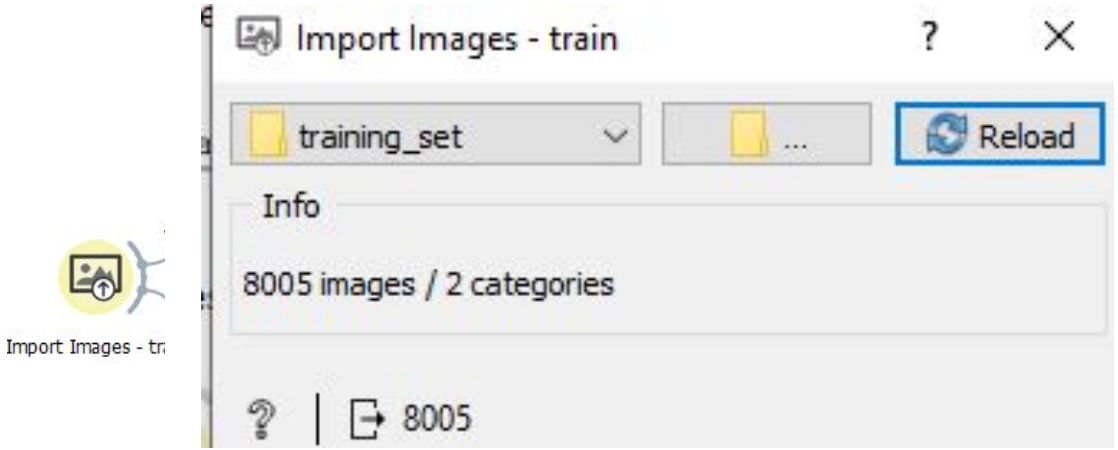
this


Model → Predictors
Predictions
Embeddings → Data
Data
Image Viewer
Logistic Regression (1)
Model
Save Model
Import Images - train
Data → Images
Embeddings → Data
Image Embedding
Data Table
Import Images - test
Data → Images
Image Embedding (1)
Embedding → Data
Learner
Test and Score
Confusion Matrix
Data
Image Viewer (1)
Logistic Regression
Learner
Neural Network

import - view - infos



folders named “training_set” and “test_set”




Data
Image Viewer

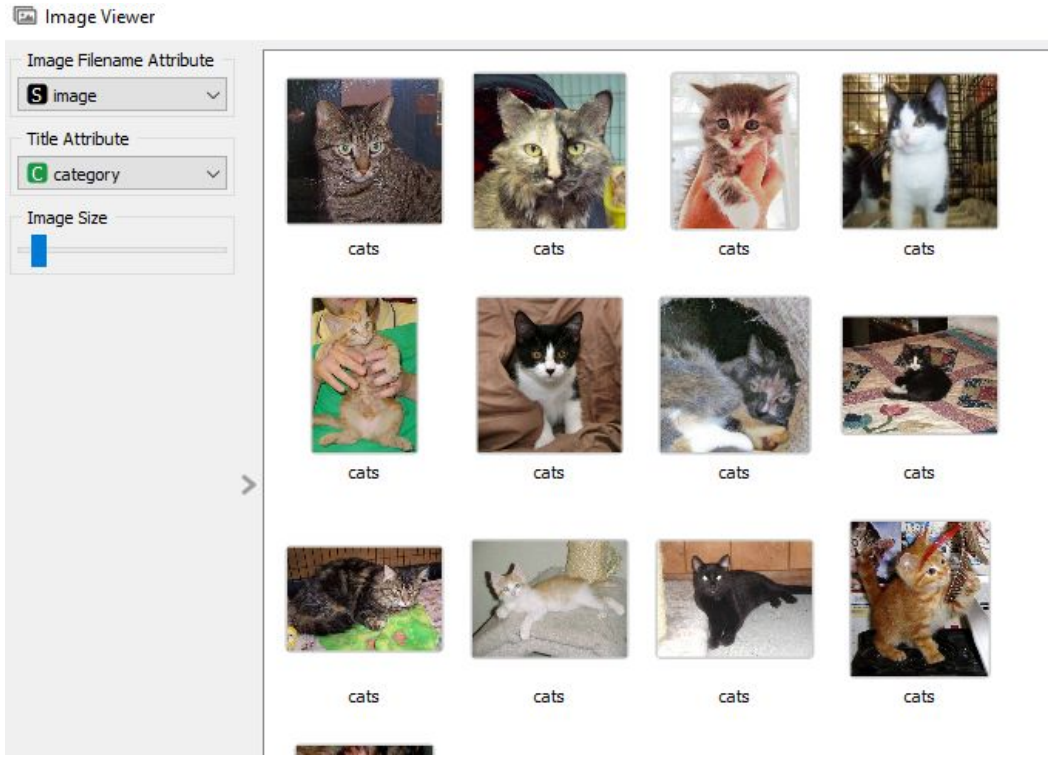
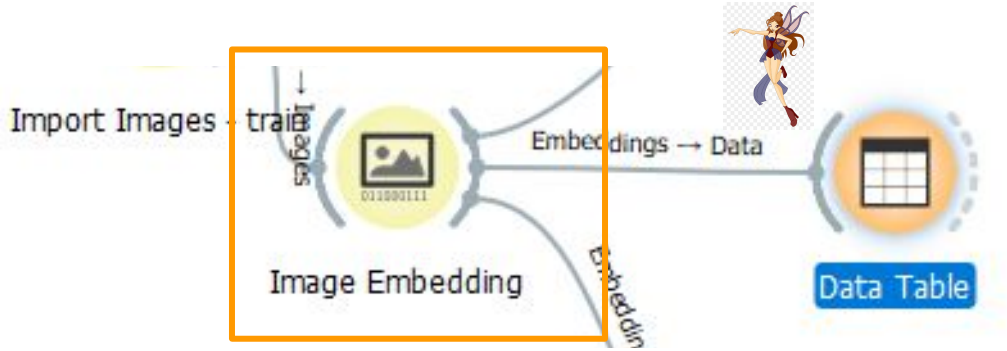
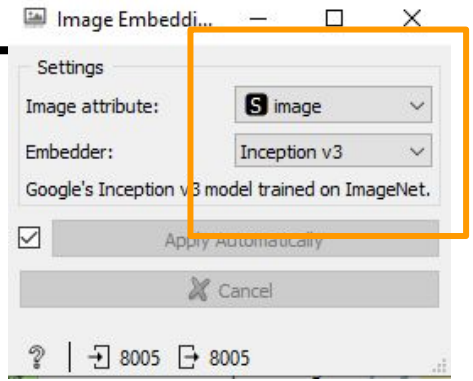


Image Viewer
Image Filename Attribute
image
Title Attribute
category
Image Size
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats
cats





Inception v3: [Google's Inception v3](https://arxiv.org/abs/1512.00567)

Image embedding through deep neural networks.

Inputs: Images: List of images.

Outputs: Embeddings: Images represented with a vector of numbers.

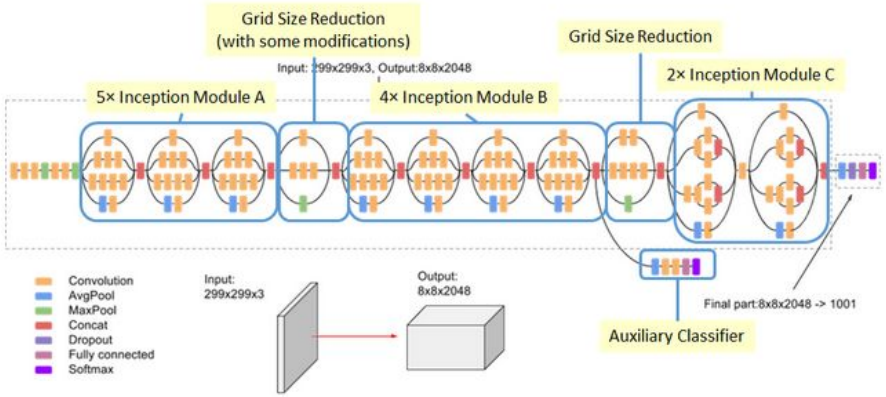
Skipped Images: List of images where embeddings were not calculated.

Image Embedding reads images and uploads them to a remote server or evaluate them locally. Deep learning models are used to calculate a feature vector for each image. It returns an enhanced data table with additional columns (image descriptors).

https://orange3-imageanalytics.readthedocs.io/en/latest/widgets/imageembedding.html

https://keras.io/api/applications/inceptionv3/

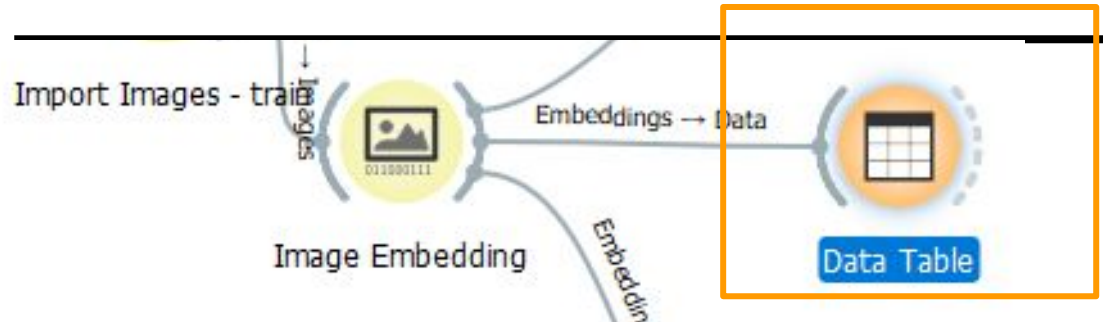
Inception v3 (*GoogLeNet*)



Inception-v3 Architecture (Batch Norm and ReLU are used after Conv)

[inception v3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.Going deeper with convolutions. InProceedings of the IEEEConference on Computer Vision and Pattern Recognition,pages 1–9, 2015

https://cloud.google.com/tpu/docs/inception-v3-advanced



Data Table

Info

8005 instances (no missing data)
2048 features
Target with 2 values
5 meta attributes

Variables

- [x] Show variable labels (if present)
- [ ] Visualize numeric values
- [x] Color by instance classes

Selection

- [x] Select full rows

	hidden origin	category	image name	image loads/orange test/(image	size	width	height	n0 True	n1 True	n2 True
1		cats	cat.1	cats\cat.1.jpg	16880	300	280	0.475694	0.432614	0.69527
2		cats	cat.10	cats\cat.10.jpg	34315	489	499	0.31333	0.241181	0.432035
3		cats	cat.100	cats\cat.100.jpg	28377	403	499	0.629442	0.265338	0.541589
4		cats	cat.1000				149	0.663272	0.333273	1.13996
5		cats	cat.1001				499	0.345729	0.111364	0.275736
6		cats	cat.1002				407	0.576959	0.631882	0.716025
7		cats	cat.1003				269	0.366972	0.44163	0.867934
8		cats	cat.1004				375	0.432891	0.0436033	0.38925

n2043 True	n2044 True	n2045 True	n2046 True	n2047 True
0.371487	0.147471	0.492729	0.140827	0.976613
0.147725	0.00344998	0.0549949	0.0806171	0.0727724
0.427751	0.181229	0.813569	0.211402	0.264475
0.640509	0	0.239425	0.195675	0.174547
0.243388	0.356838	0.0456228	0.112791	0.293549
0.072368	0.0131649	0.373584	0.0435946	0.241524
0.0247444	0.0243987	0.082787	0.0150555	0.595918
0.329187	0.0473317	0.243795	0.254472	0.0699269

https://orange3-imageanalytics.readthedocs.io/en/latest/widgets/imageembedding.html

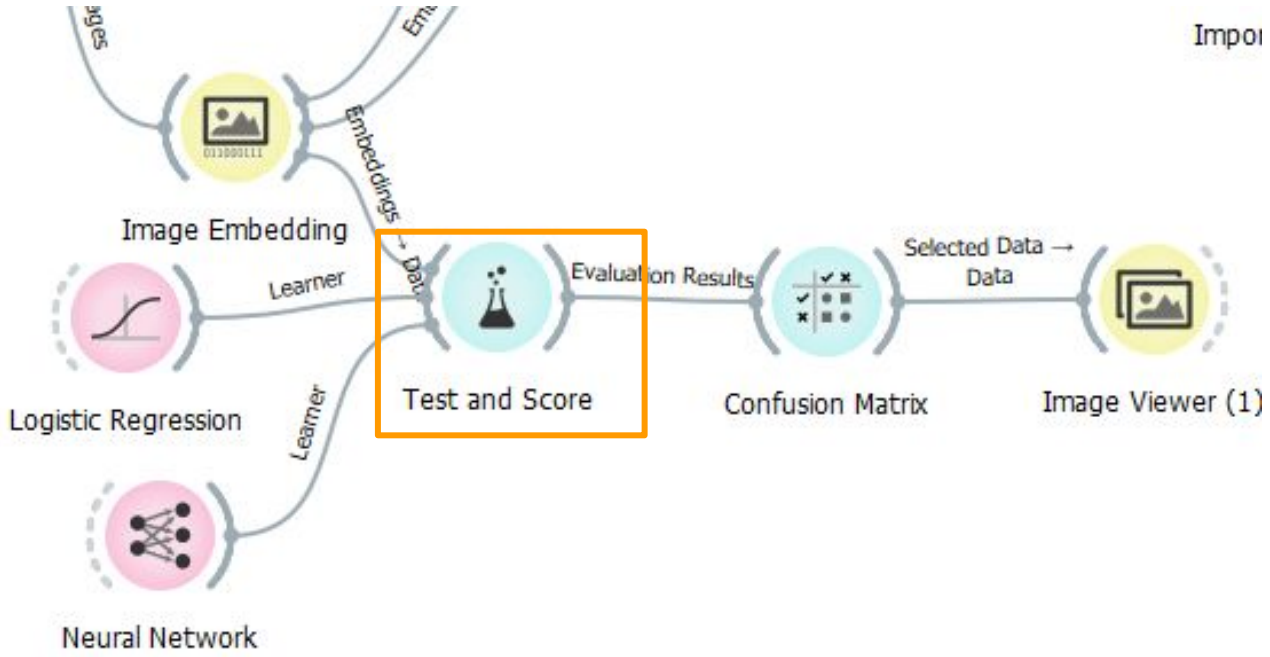


Image Embedding
Embeddings
Learner
Logistic Regression
Learner
Neural Network
Test and Score
Evaluation Results
Confusion Matrix
Selected Data → Data
Image Viewer (1)

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/evaluation.testing.html
https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/evaluation.cd.html

Test and Score

Tests learning algorithms on data.

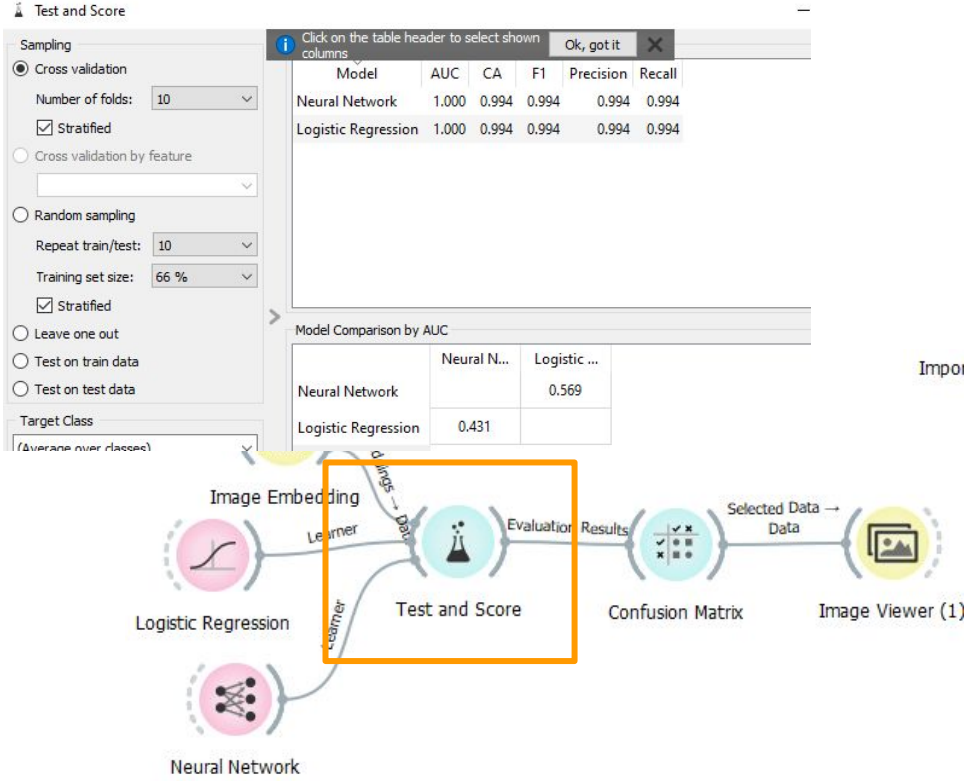
Inputs

- Data: input dataset
- Test Data: separate data for testing
- Learner: learning algorithm(s)

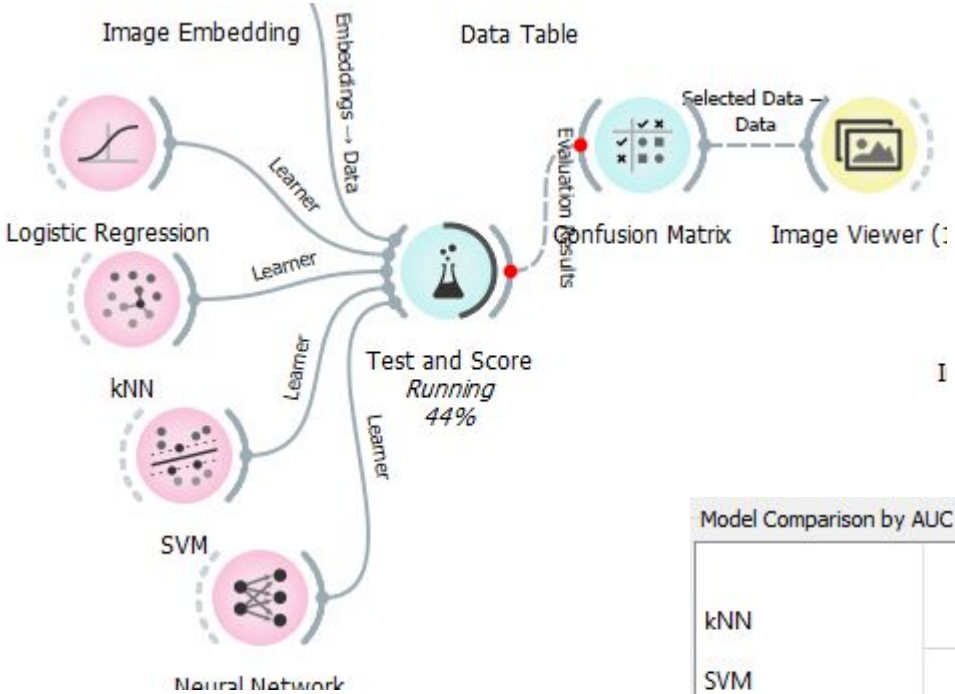
Outputs

- Evaluation Results: results of testing classification algorithms

The widget tests learning algorithms. Different sampling schemes are available, including using separate test data.



https://orangedatamining.com/widget-catalog/evaluate/testandscore/



Evaluation Results

Model	AUC	CA	F1	Precision	Recall
kNN	0.997	0.993	0.993	0.993	0.993
SVM	1.000	0.994	0.994	0.994	0.994
Neural Network	1.000	0.994	0.994	0.994	0.994
Logistic Regression	1.000	0.994	0.994	0.994	0.994

Model	AUC	CA	F1	Precision	Recall
kNN	0.997	0.993	0.993	0.993	0.993
SVM	1.000	0.994	0.994	0.994	0.994
Random Forest	0.999	0.991	0.991	0.991	0.991
Neural Network	1.000	0.994	0.994	0.994	0.994
Naive Bayes	0.989	0.984	0.984	0.985	0.984
Logistic Regression	1.000	0.994	0.994	0.994	0.994

Model Comparison by AUC

	kNN	SVM	Rando...	Neural ...	Naive B...	Logistic...
kNN		0.006	0.025	0.006	1.000	0.007
SVM	0.994		0.938	0.618	1.000	0.641
Random Forest	0.975	0.062		0.068	1.000	0.050
Neural Network	0.994	0.382	0.932		1.000	0.569
Naive Bayes	0.000	0.000	0.000	0.000		0.000
Logistic Regression	0.993	0.359	0.950	0.431	1.000	

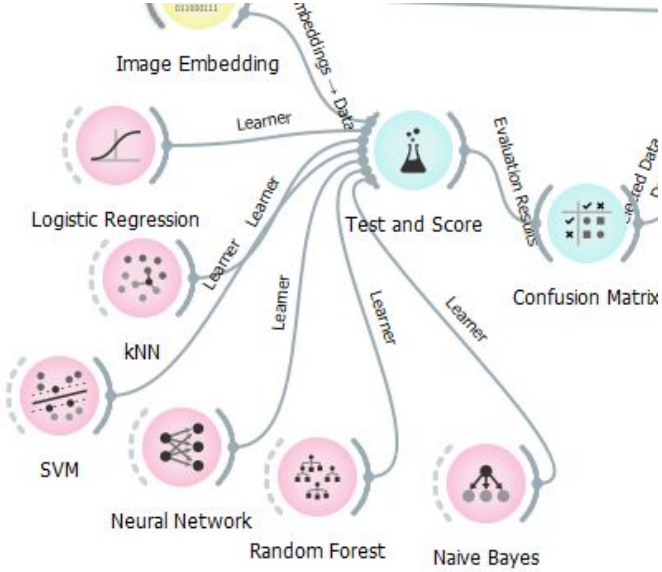
Model Comparison by AUC

kNN

SVM

Neural Network

Logistic Regression

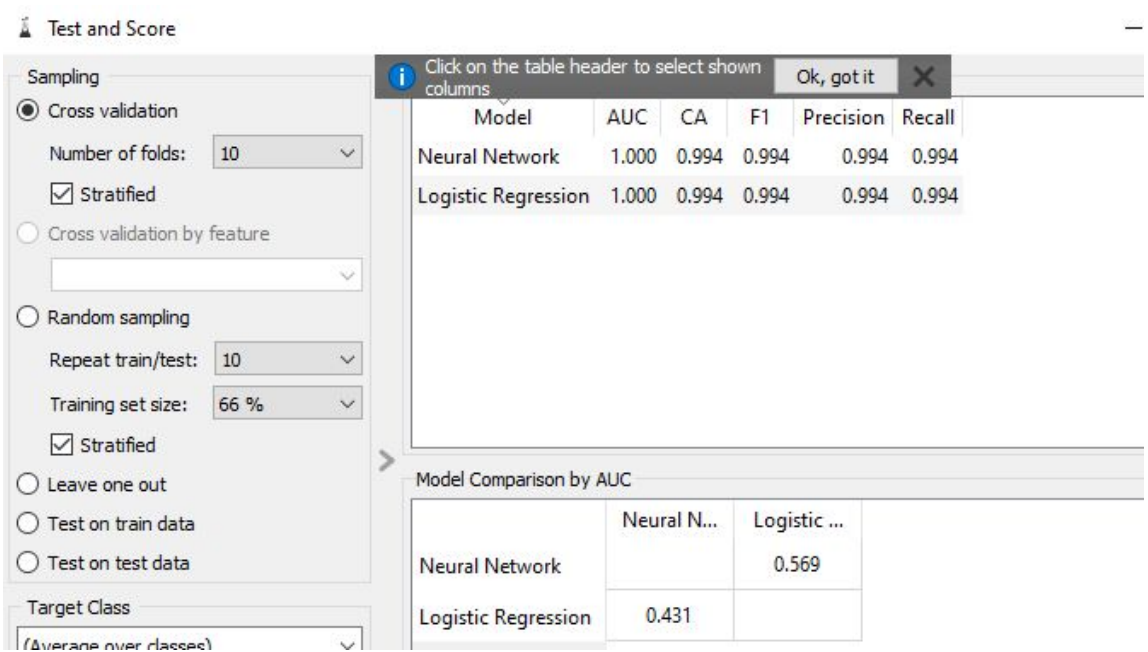


Model	AUC	CA	F1	Precision	Recall
kNN	0.997	0.993	0.993	0.993	0.993
SVM	1.000	0.994	0.994	0.994	0.994
Random Forest	0.999	0.991	0.991	0.991	0.991
Neural Network	1.000	0.994	0.994	0.994	0.994
Naive Bayes	0.989	0.984	0.984	0.985	0.984
Logistic Regression	1.000	0.994	0.994	0.994	0.994

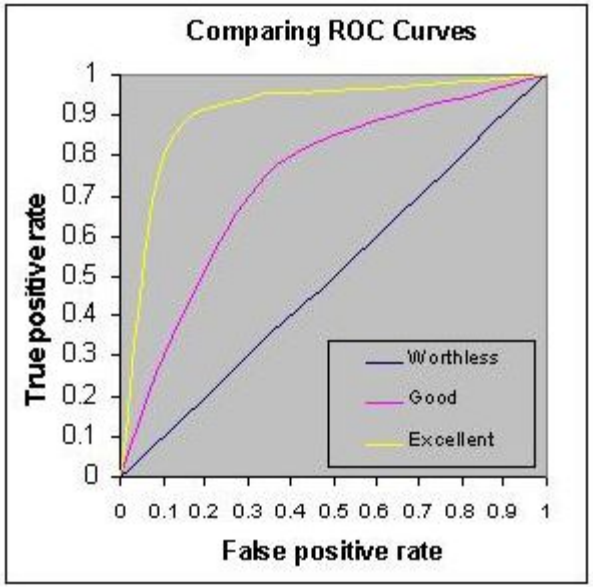
Model Comparison by AUC

	kNN	SVM	Rando...	Neural ...	Naive B...	Logistic...
kNN		0.006	0.025	0.006	1.000	0.007
SVM	0.994		0.938	0.618	1.000	0.641
Random Forest	0.975	0.062		0.068	1.000	0.050
Neural Network	0.994	0.382	0.932		1.000	0.569
Naive Bayes	0.000	0.000	0.000	0.000		0.000
Logistic Regression	0.993	0.359	0.950	0.431	1.000	

- Area under ROC is the area under the receiver-operating curve.
- Classification accuracy is the proportion of correctly classified examples.
- F-1 is a weighted harmonic mean of precision and recall (see below).
- Precision is the proportion of true positives among instances classified as positive, e.g. the proportion of *Iris virginica* correctly identified as Iris virginica.
- Recall is the proportion of true positives among all positive instances in the data, e.g. the number of sick among all diagnosed as sick.



- **Area under ROC** is the area under the receiver-operating curve.



- .90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)

Click on the table header to se
columns

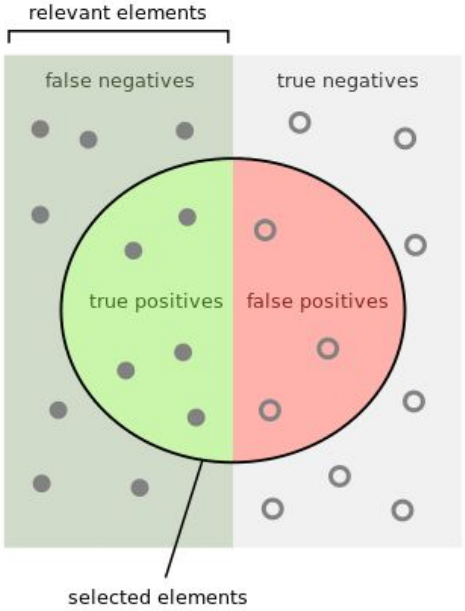
Model	AUC
Neural Network	1.000
Logistic Regression	1.000

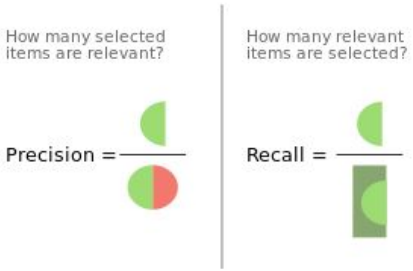
- **Classification accuracy** is the proportion of correctly classified examples.

accuracy is closeness of the measurements to a specific value, while **precision** is the closeness of the measurements to each other.

- **Precision** is the proportion of true positives among instances classified as positive, e.g. the proportion of *Iris virginica* correctly identified as Iris virginica.

- **Recall** is the proportion of true positives among all positive instances in the data, e.g. the number of sick among all diagnosed as sick.



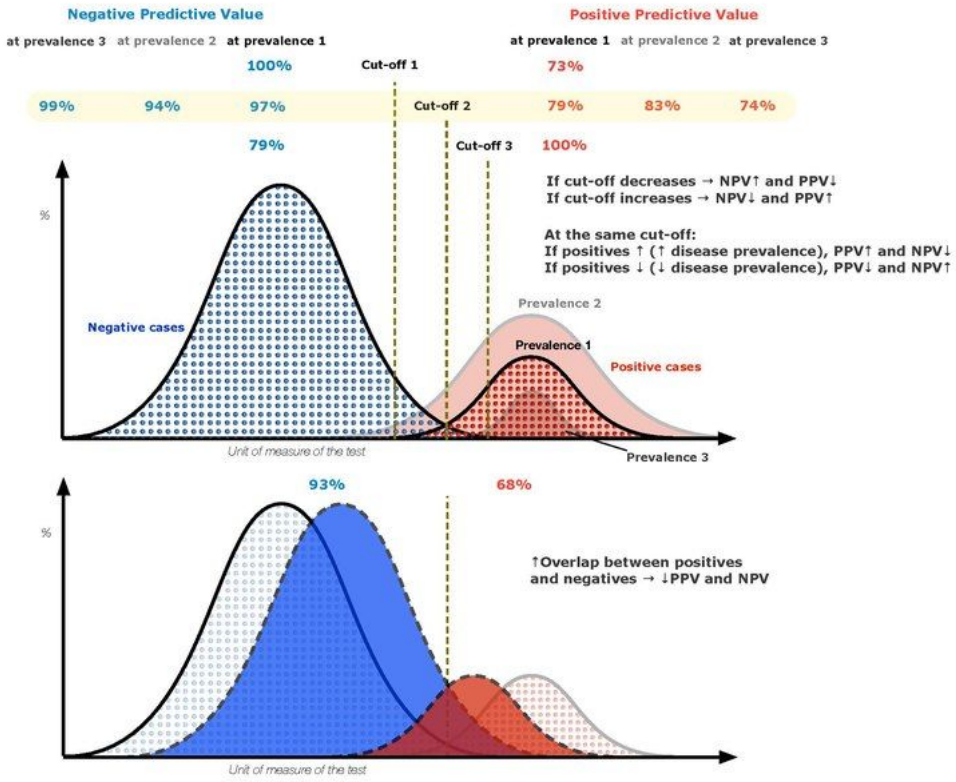


Ok, got it

	Precision	Recall
4	0.994	0.994
4	0.994	0.994

Both precision and recall are therefore based on relevance.

- **Precision** is the proportion of true positives among instances classified as positive, e.g. the proportion of *Iris virginica* correctly identified as Iris virginica.



precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances

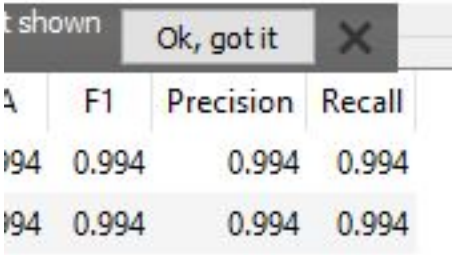
F-score or **F-measure** is a measure of a test's accuracy.

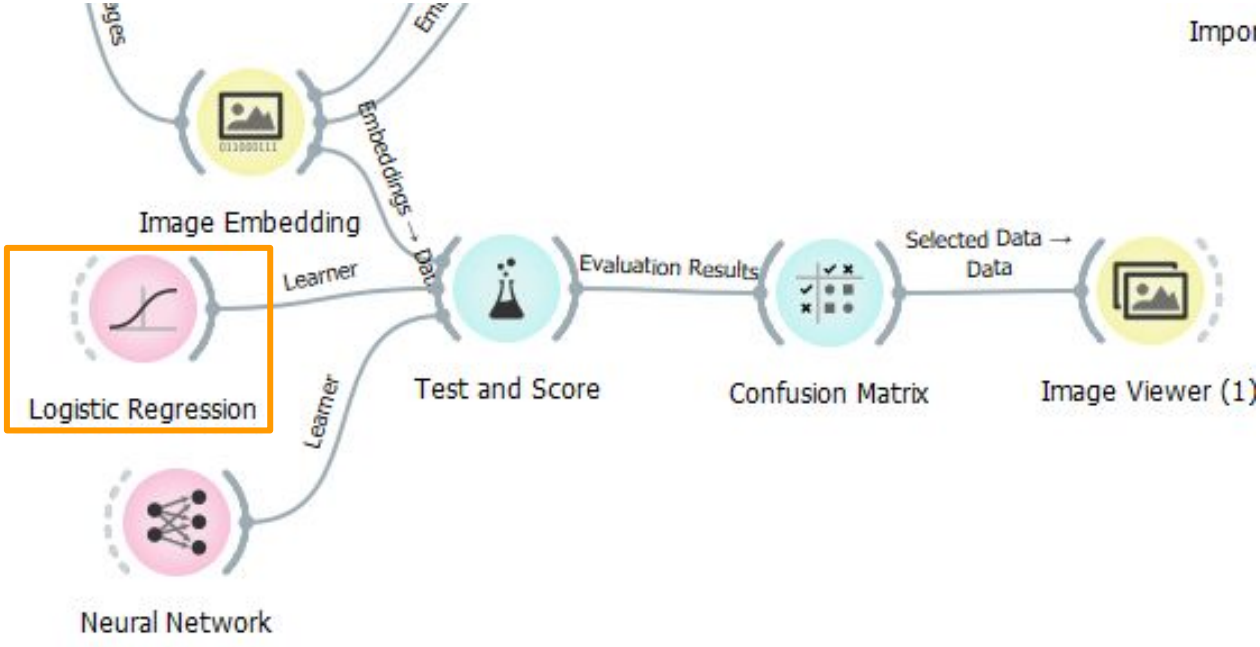
- **F-1** is a weighted harmonic mean of precision and recall .

The traditional F-measure or balanced F-score (**F_1 score**) is the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. The F_1 score is also known as the Dice similarity coefficient (DSC)




Impor
Image Embedding
Learner
Logistic Regression
Test and Score
Evaluation Results
Confusion Matrix
Selected Data →
Data
Image Viewer (1)
Learner
Neural Network

Logistic Regression

```
def computeLogisticRegressionModel(XTrain, yTrain):
    from sklearn.linear_model import LogisticRegression

    classifier = LogisticRegression(solver='lbfgs')
    classifier.fit(XTrain[0], yTrain)

    return classifier

def computeLogisticRegressionExample(filename):
    X, y, _, _ = pre.loadDataset(filename, ",")
    X = pre.fillMissingData(X, 2, 3)

    X = pre.computeCategorization(X)
    X = pre.computeCategorization(X)
```

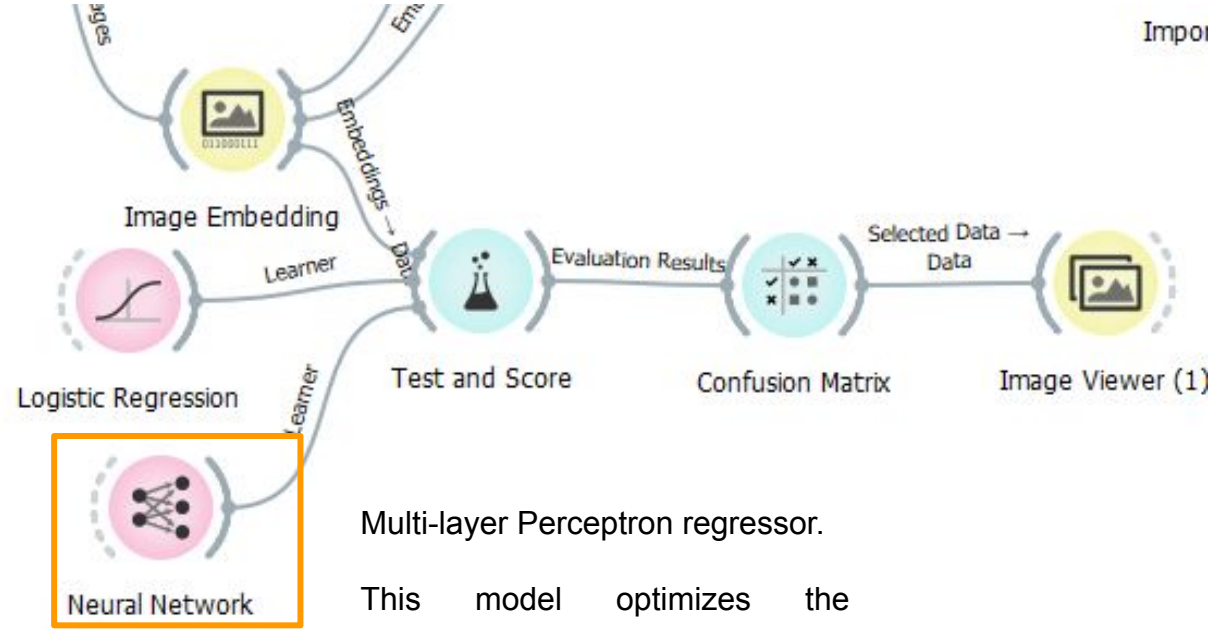
https://www.youtube.com/watch?v=DMDY0Gar7Fw

regressão logística (ou, classificador de máxima entropia) é uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou binária

Seja $p(x)$ a probabilidade de êxito quando o valor da variável preditiva é x.
Então,

$$p(x) = \frac{1}{1 + e^{-(B_0 + B_1 x)}} = \frac{e^{B_0 + B_1 x}}{1 + e^{B_0 + B_1 x}}.$$

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/classification.html?highlight=logistic#logistic-regression

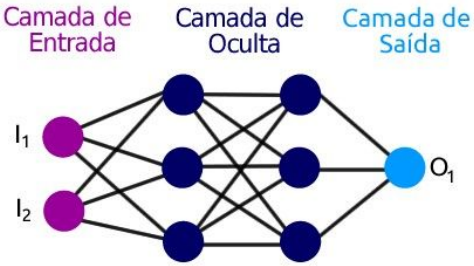


Multi-layer Perceptron regressor.

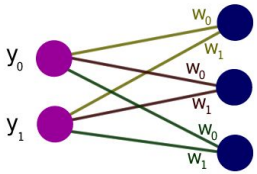
This model optimizes the squared-loss using LBFGS or stochastic gradient descent.

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/regression.html?highlight=neural#neural-network

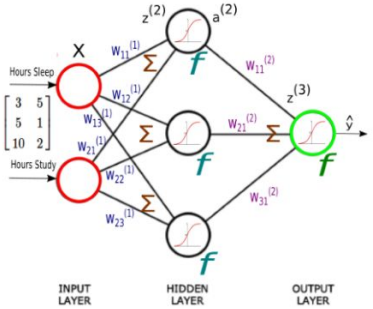
Neural Network

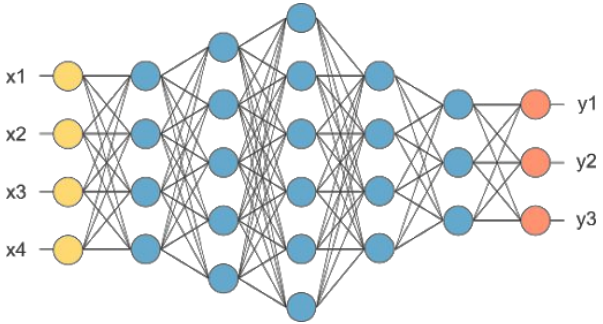


I = [I_1, I_2] = Vetor de Entrada
O = [O_1] = Vetor de Saída

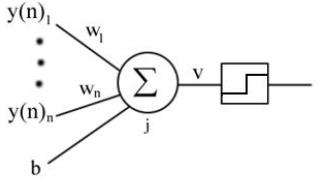


$$v_j = \sum_{i=0}^{m} w_i y_i + b$$





computador aprende a realizar uma tarefa analisando exemplos de treinamento, que normalmente são indicados manualmente com antecedência.

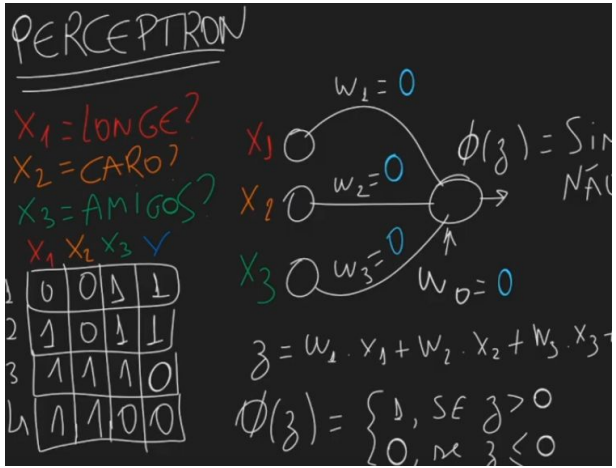


- Processamento de linguagem natural;
- Reconhecimento de fala e imagens;
- Previsão de valores.

Neural Network

$$(\boldsymbol{X}\boldsymbol{W_1})\boldsymbol{w} = \boldsymbol{y}$$

$$\begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \dots & x_{nd} \end{bmatrix} \times \begin{bmatrix} w_{01} & w_{01} & \dots & w_{0m} \\ w_{11} & w_{11} & \dots & w_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{d1} & w_{d1} & \dots & w_{dm} \end{bmatrix} \times \begin{bmatrix} w_{01} \\ w_{11} \\ \vdots \\ w_{d1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$



https://www.youtube.com/watch?v=tYXGzQs31Og

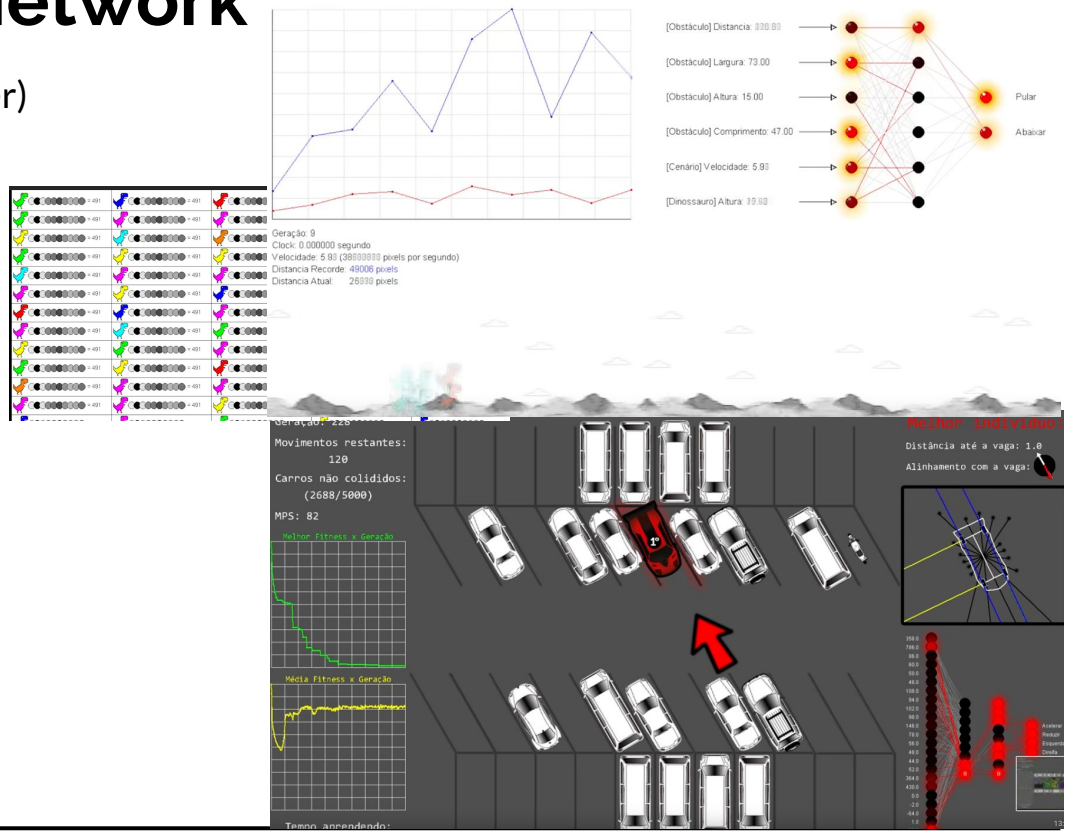
Neural Network

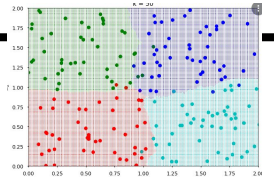
Rede Neural Artificial (Perceptron Multilayer)

Jogando!

[Dino:]

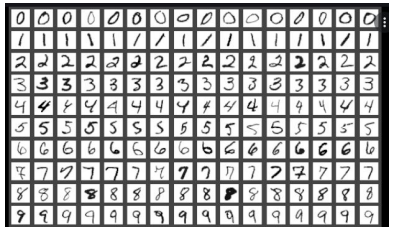
[Estacionando:]





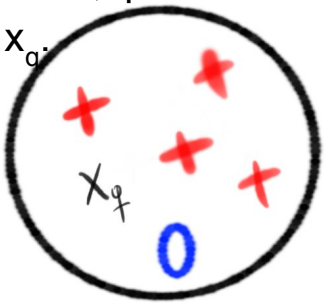
kNN - K-nearest neighbors

dada uma instância de teste x_q, o algoritmo encontra os *k* vizinhos mais próximos de x_q no conjunto de treinamento. Em seguida, a classe de x_q é dada pela classe que ocorrer com maior frequência entre os *k* vizinhos.



Na figura, são mostrados os cinco vizinhos mais próximos da instância de teste x_q. Dessas cinco instâncias, 4 são da classe “+” (vermelha) e 1 da classe “0” (azul). Ao aplicar o *k*NN, com *k*=5, a instância x_q é classificada como sendo da classe vermelha, pois essa classe possui mais representantes na vizinhança de x_q.

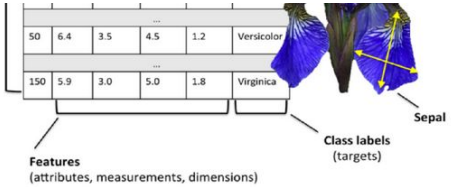
$$D_E(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$



http://docs.biolab.si/orange/2/reference/rst/Orange.classification.knn.html#Orange.classification.knn.kNNClassifier.__call__

kNN - K-nearest neighbors

[knn colab]



> **To perform some extra checks to the Dataframe, the Pandas library itself mu**
> work on any data it is essencial to do a prelimiary analysis of the data: the typ
> missing values. Then, take the first conclusions and correct any issue* in app

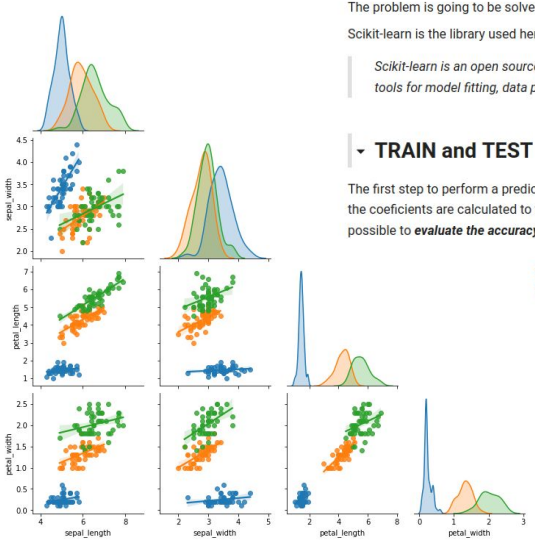
*this is not the case of this well-known data but the checks are to be performed ar

```
[ ] import pandas as pd
```

```
[ ] pd.value_counts(iris.species)
```

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

The dataset is made of 150 flowers (150 rows), 50 of each specie.



MACHINE LEARNING - Predictive Analysis

After **performing the exploratory** analysis, what is basically, evaluate and reaching some conclusions looking to the charts of distribution and scatter of the data.

Now, the predictive analysis is going to be performed not by a person itself but with the aid of a computer, mathematical algorithms, **machine learning**.

The problem is going to be solved through the **KNN** algorithm. The best pratices are going to be performed and explained.

Scikit-learn is the library used here for machine learning in Python. From it several modulus are imported as needed.

> *Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.*

TRAIN and TEST Data

The first step to perform a predictive analysis through a Machine Learning algorithm is to separate the data in two parts. One to be **trained**, so the coeficients are calculated to the best fit to this data, according to the algorithm chosen. The second part is reserved to be **tested**, so it is possible to ***evaluate the accuracy*** of the method with a different data, so that you can extrapolate to any data.

[Atividade com knn no colab]

SVM - Support Vector Machine

SVM é um algoritmo de aprendizado de máquina **supervisionado** que pode ser usado para desafios de classificação ou regressão.

Para treinamento e classificação de um dataset.

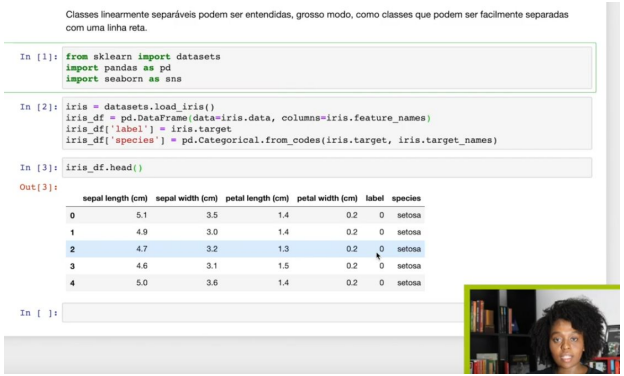
SVM é uma fronteira que melhor segrega as duas classes (hiperplano / linha).

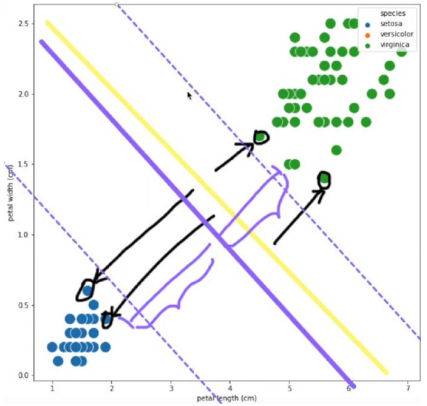
https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/svm.html

SVM - scikit - python

colab - svm

CLASSIFICAÇÃO com Máquinas de Vetores de Suporte (SVM) | Machine Learning #05

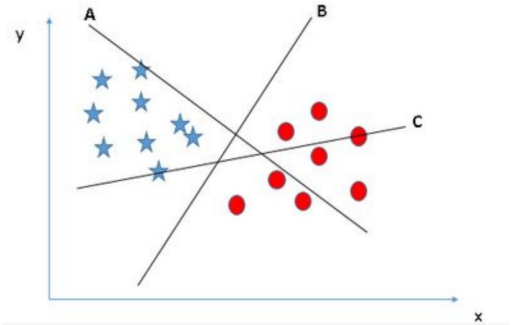




SVM - scikit - python

SVM no site do scikit-learn

supervised learning methods used for classification, regression and outliers detection.



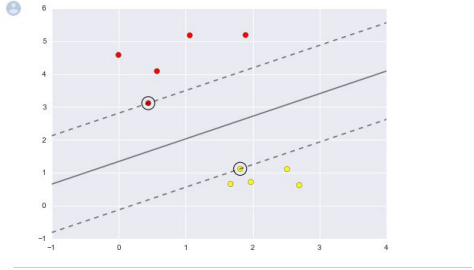
SVM - scikit - python

SVM no colab

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Python Data Science handbook

```
from ipywidgets import interact, fixed
interact(plot_svm, N=[10, 200], ax=fixed(None));
```

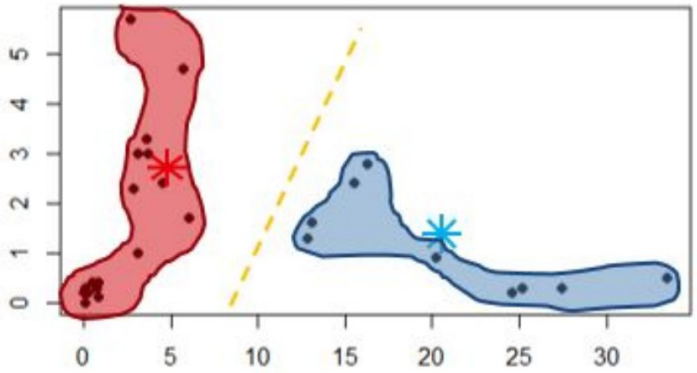


```
fig, ax = plt.subplots(3, 5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],
            xlabel=faces.target_names[faces.target[i]])
```


```
from sklearn.metrics import confusion_matrix
res = confusion_matrix(y_test, predictions)
print("Confusion Matrix")
print(res)
print(f"Test Set: {len(X_test)}")
print(f"Accuracy = {percentage*100} %")
```

```
Confusion Matrix
[[63  8]
 [ 1 99]]
Test Set: 171
Accuracy = 94.73684210526315 %
```

K-Means

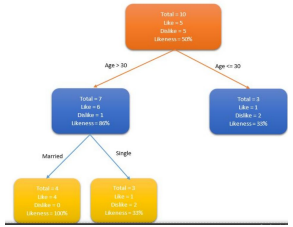


Naïve Bayes

classificadores probabilísticos simples com base na aplicação Bayes ‘teorema com forte independência entre as características. A imagem em destaque é a equação – em que P (A| B) é a probabilidade posterior, P (B|A) é a probabilidade, P (A) é a probabilidade prévia e P (B) é preditor de probabilidade prévia.

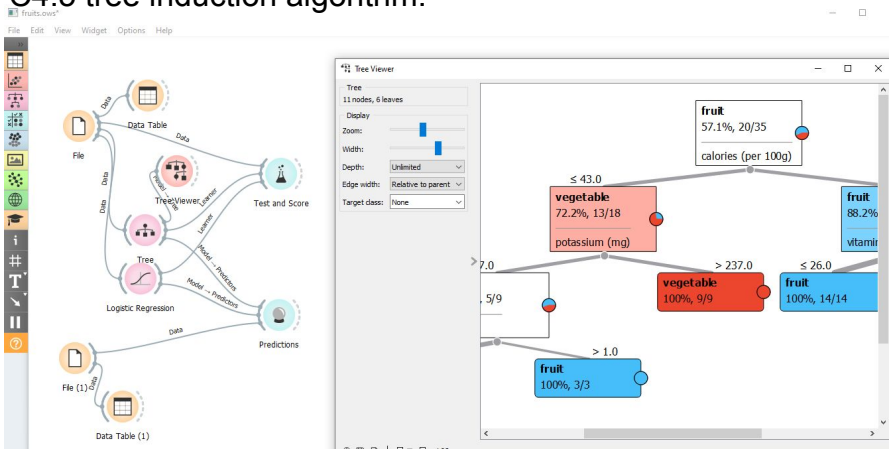
http://docs.biolab.si/orange/2/reference/rst/Orange.classification.bayes.html#Orange.classification.bayes.NaiveClassifier.__call__

Árvore de decisão



Uma árvore de decisão é uma ferramenta de apoio que utiliza um gráfico ou modelo de decisões e suas possíveis consequências.

C4.5 tree induction algorithm.



https://docs.biolab.si/orange/2/reference/rst/Orange.classification.tree.html

Supervisionado

O aprendizado supervisionado é baseado na regressão básica e classificação. O humano fornece um banco de dados e ensina a máquina a reconhecer o que é uma bicicleta, por exemplo, entre padrões e semelhanças. A cor e tamanho pode variar, mas a máquina aprende que uma bicicleta possui pedais, duas rodas, guidão e outros elementos-chave.

não supervisionado

- Associação;
- Clusterização.

A associação: permite o descobrimento de regras e correlações, identificando conjuntos de itens que frequentemente ocorrem juntos. Os varejistas costumam usar esta análise em carrinhos de compras, para descobrir itens frequentemente comprados em conjunto, desenvolvendo assim estratégias mais eficazes de marketing e merchandising.

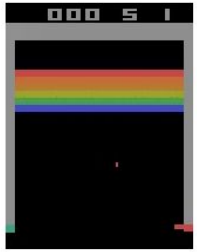
Na clusterização (ou agrupamento): o conjunto todo em análise sofre segmentações em vários grupos, com base nas semelhanças encontradas. É uma técnica que permite dividir automaticamente um conjunto de dados em grupos de acordo com medidas de similaridade ou de distância.

não supervisionado

a máquina começa a analisar, sozinha, os dados e a identificar os padrões — aprendendo a separar o que é uma lata de uma garrafa, por exemplo.

+Demorado

Performance inicial do agente:



Após 15 minutos de treinamento:

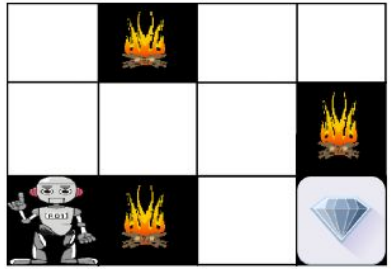
Após 30 minutos de treinamento:

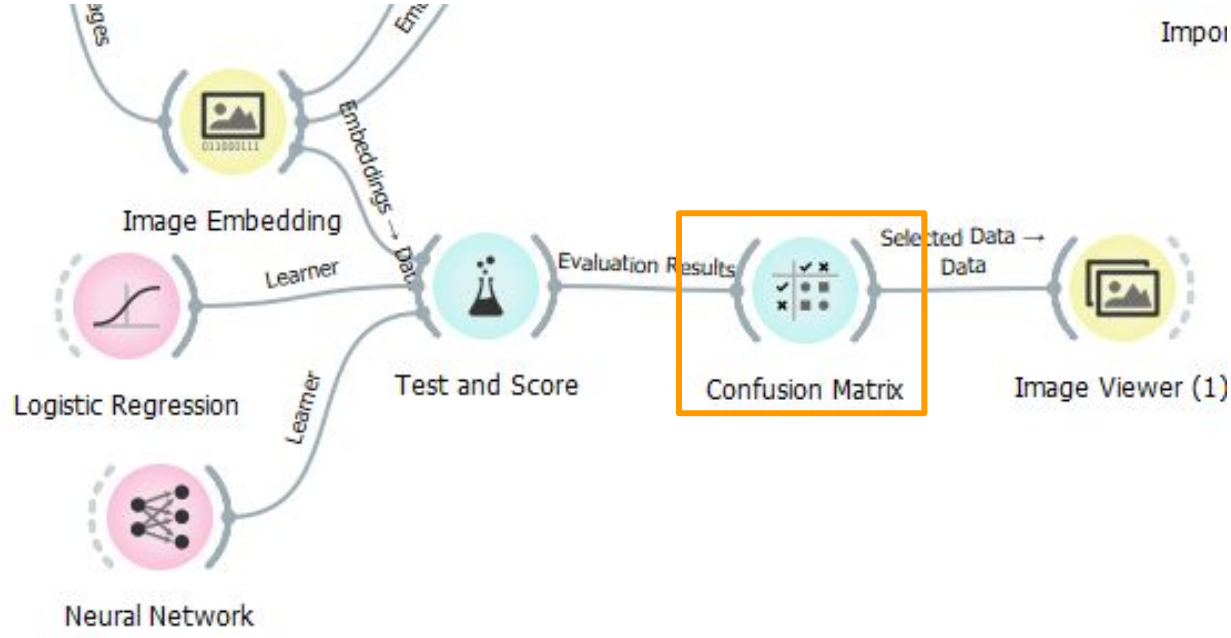
Reforço

No Aprendizado por Reforço o modelo aprende executando ações e avaliando recompensas.

O agente realiza uma ação num dado ambiente, alterando seu estado inicial, o que gera uma recompensa ao agente. De forma cíclica, o agente avalia esta recompensa (que pode ser positiva ou negativa) e age novamente no ambiente, gerando o aprendizado.

- Q-Learning;
- Aproximação por função com atualização por gradiente;
- Multi-Armed Bandits;
- Contextual Bandits;
- k-Armed Bandits




Impor
Image Embedding
Embeddings →
Learner
Logistic Regression
Learner
Neural Network
Test and Score
Evaluation Results
Confusion Matrix
Selected Data →
Data
Image Viewer (1)

Confusion Matrix

Em análise preditiva, a **matriz de confusão** (às vezes também chamada de *'matriz de erro' ou 'tabela de confusão'*) é uma tabela com duas linhas e duas colunas que relata o número de *falsos positivos* , *falsos negativos* , *verdadeiros positivos* e *verdadeiros negativos* . Isso permite uma análise mais detalhada do que a mera proporção de classificações corretas (precisão). A **precisão** produzirá resultados enganosos se o conjunto de dados estiver desequilibrado; isto é, quando o número de observações em diferentes classes variam muito.

Por exemplo, se houver 95 gatos e apenas 5 cachorros nos dados, um determinado classificador pode classificar todas as observações como gatos. A precisão geral seria de 95%, mas com mais detalhes o classificador teria uma taxa de reconhecimento de 100% para a classe de gatos, mas uma taxa de reconhecimento de 0% para a classe de cães.

O **F1 Score (ou f-measure)** é ainda mais confiável em tais casos, e aqui renderia mais de 97,4%, enquanto remove o viés e produz 0 como a probabilidade de uma decisão informada para qualquer forma de suposição (supondo gato).

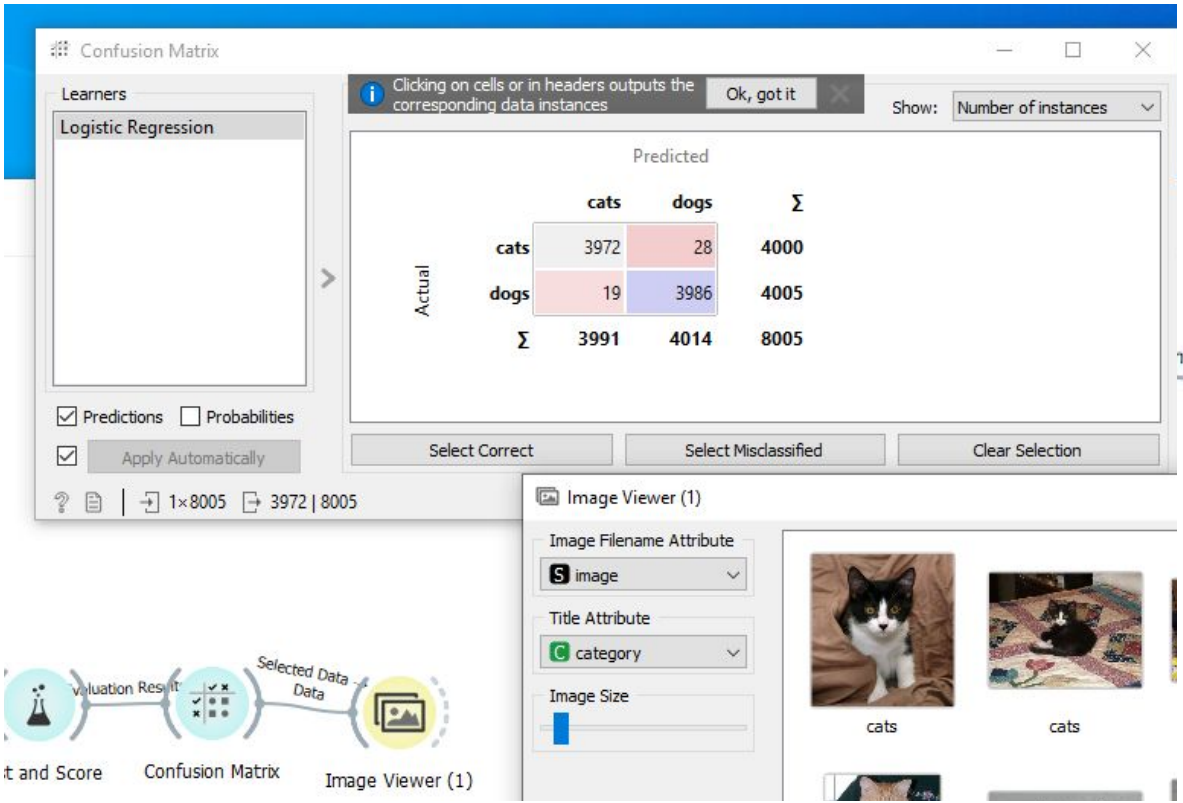
Descomplicando a Matriz de Confusão

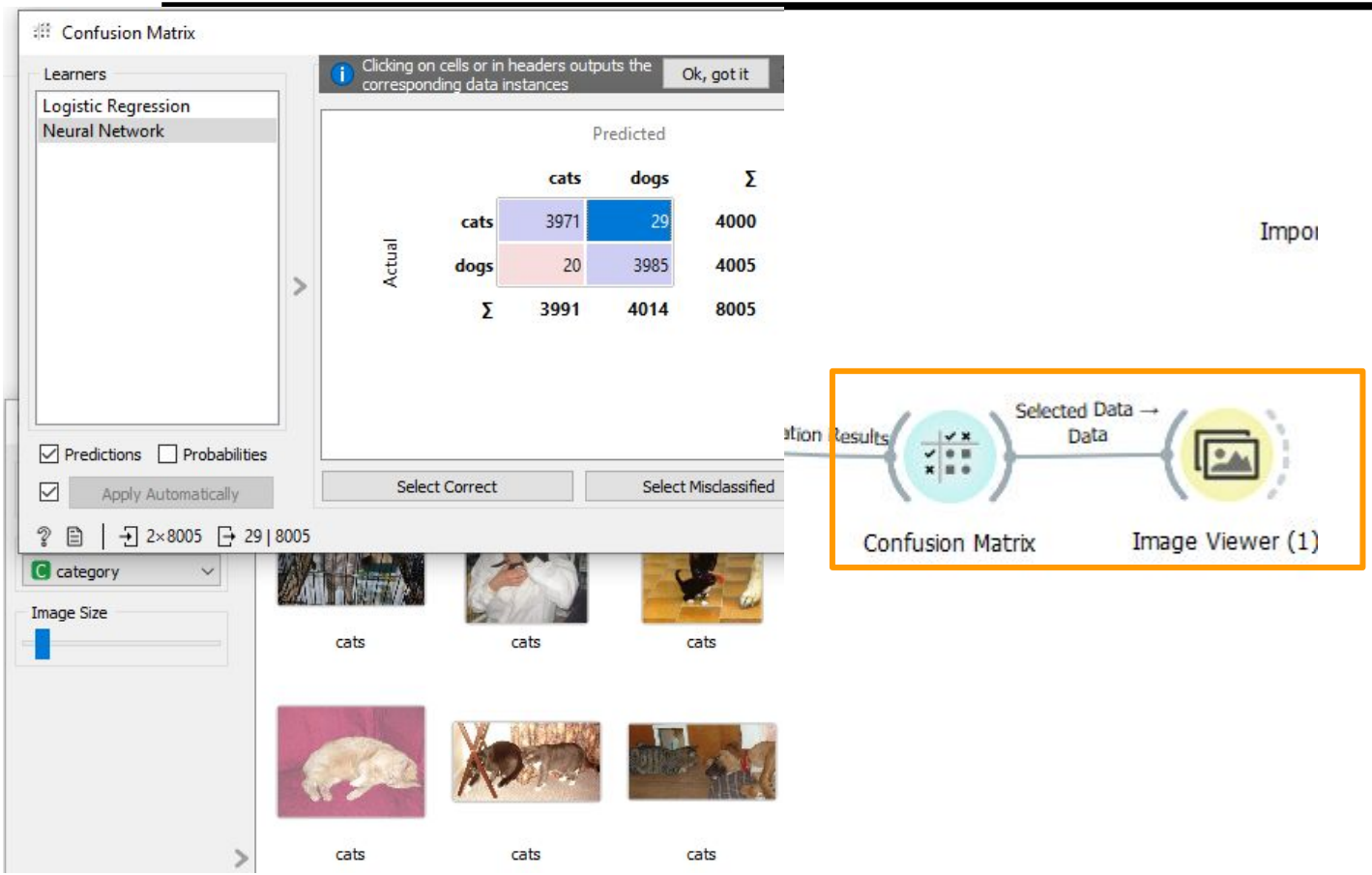
Colab - Matrix Confusão

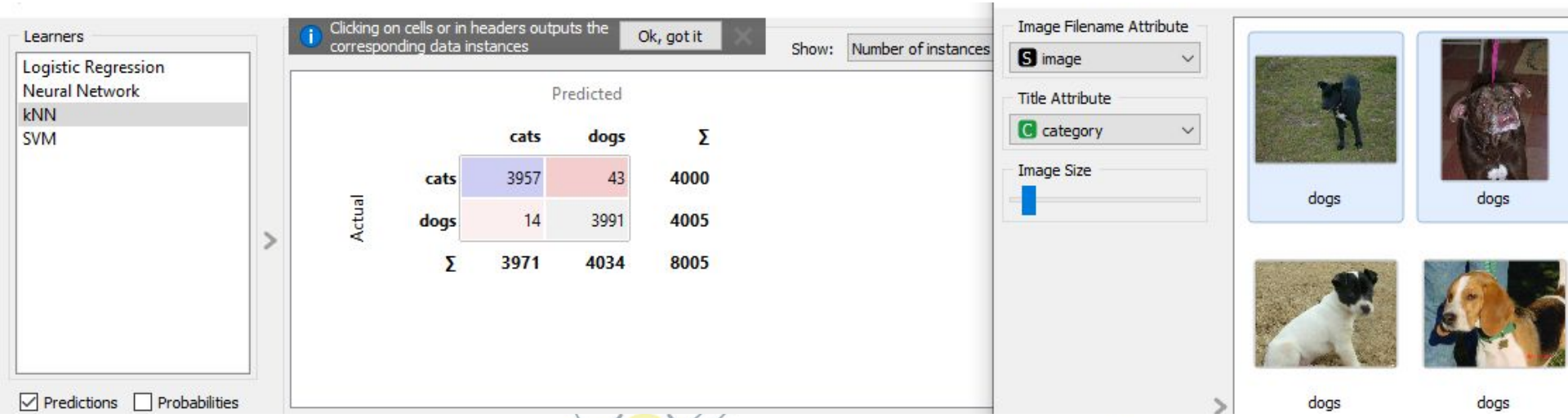
Confusion Matrix

```
import numpy as np

# 1 para grávida, 0 para não grávida
valores_reais    = [1, 0, 1, 0, 0, 0, 1, 0, 1, 0]
valores_preditos = [1, 0, 0, 1, 0, 0, 1, 1, 1, 0]

def get_confusion_matrix(reais, preditos, labels):
..

..

...

get_confusion_matrix(reais=valores_reais, preditos=valores_preditos, labels=[ 1,0])
# array([[3, 1], [2, 4]])
```

Colab - Matrix Confusão


Confusion Matrix
Learners
Logistic Regression
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Show:
Number of instances
Predicted
cats
dogs
Σ
Actual
cats
3972
28
4000
dogs
19
3986
4005
Σ
3991
4014
8005
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection
1×8005
3972 | 8005
Image Viewer (1)
Image Filename Attribute
image
Title Attribute
category
Image Size
cats
cats
Selected Data
Data
Confusion Matrix
Image Viewer (1)


Confusion Matrix
Learners
Logistic Regression
Neural Network
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Predicted
cats
dogs
Σ
Actual
cats 3971 29 4000
dogs 20 3985 4005
Σ 3991 4014 8005
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
2×8005
29 | 8005
category
Image Size
cats
cats
cats
cats
cats
cats
Impor
Selected Data →
Data
Confusion Matrix
Image Viewer (1)


Learners
Logistic Regression
Neural Network
kNN
SVM
Predictions
Probabilities
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Show:
Number of instances
Predicted
cats
dogs
Σ
Actual
cats 3957 43 4000
dogs 14 3991 4005
Σ 3971 4034 8005
Image Filename Attribute
image
Title Attribute
category
Image Size
dogs
dogs
dogs
dogs

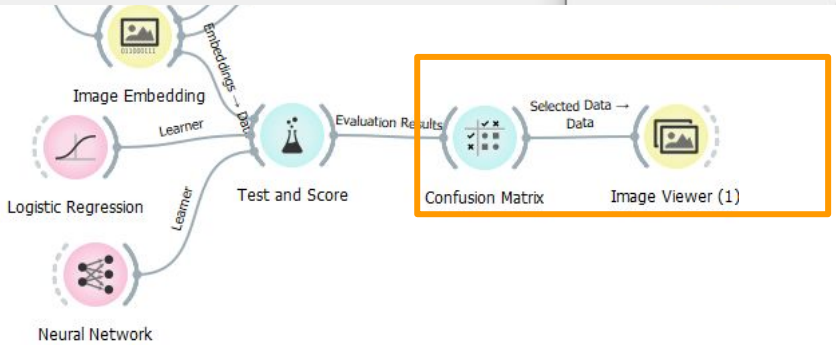
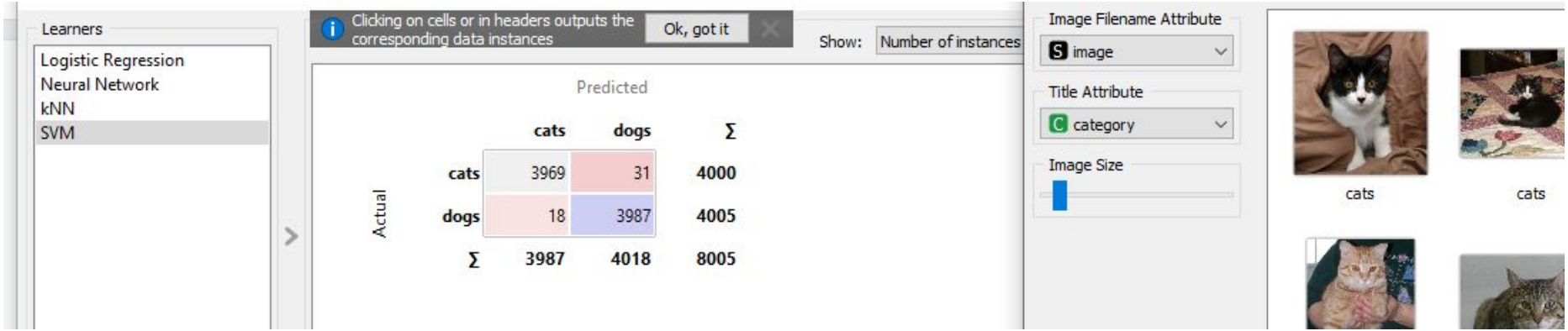


Image Embedding
Embeddings → Data
Learner
Logistic Regression
Learner
Neural Network
Test and Score
Evaluation Results
Confusion Matrix
Selected Data → Data
Image Viewer (1)


Learners
Logistic Regression
Neural Network
kNN
SVM
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Show:
Number of instances
Predicted
cats
dogs
Σ
Actual
cats 3969 31 4000
dogs 18 3987 4005
Σ 3987 4018 8005
Image Filename Attribute
image
Title Attribute
category
Image Size
cats
cats

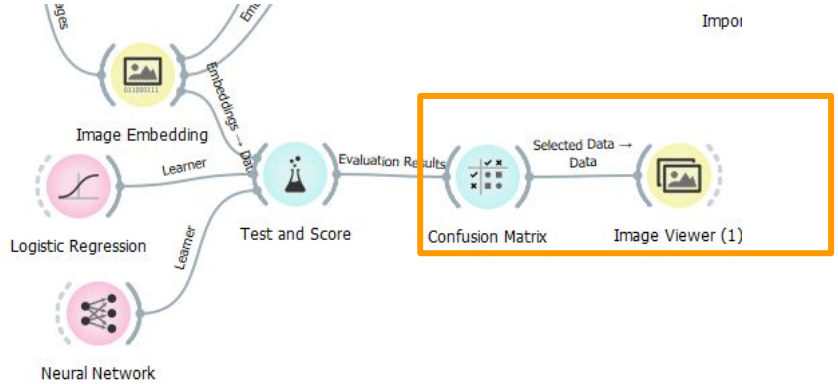
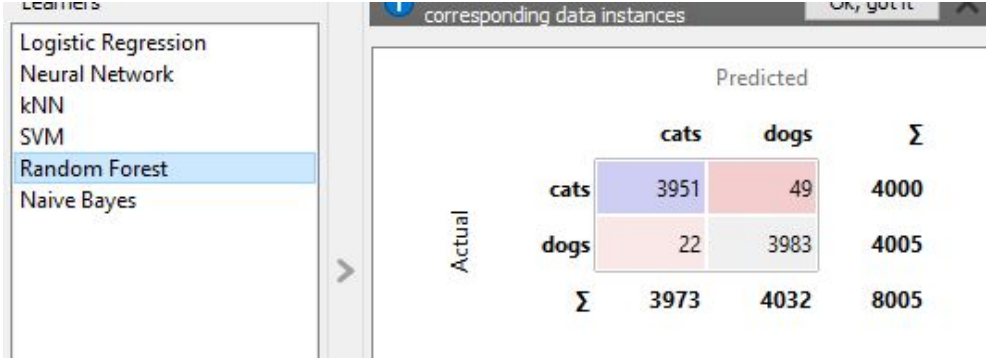


Image Embedding
Learner
Logistic Regression
Learner
Neural Network
Test and Score
Evaluation Results
Confusion Matrix
Selected Data → Data
Image Viewer (1)


Learners
Logistic Regression
Neural Network
kNN
SVM
Random Forest
Naive Bayes
corresponding data instances
Predicted
cats
dogs
Σ
Actual
cats 3951 49 4000
dogs 22 3983 4005
Σ 3973 4032 8005

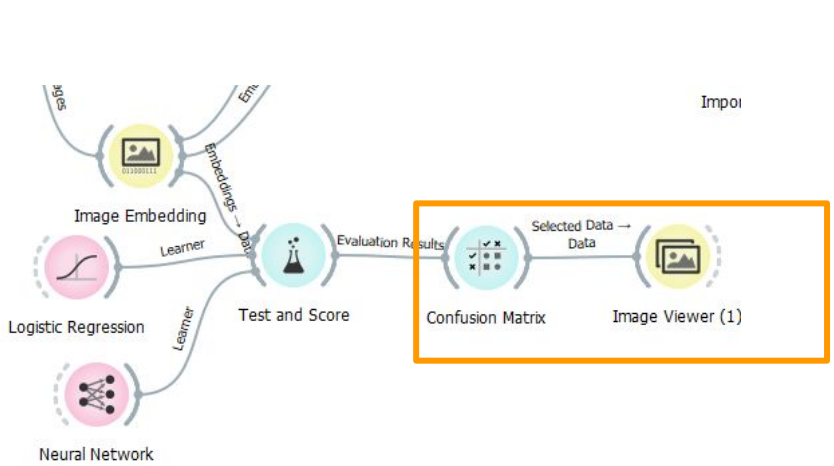
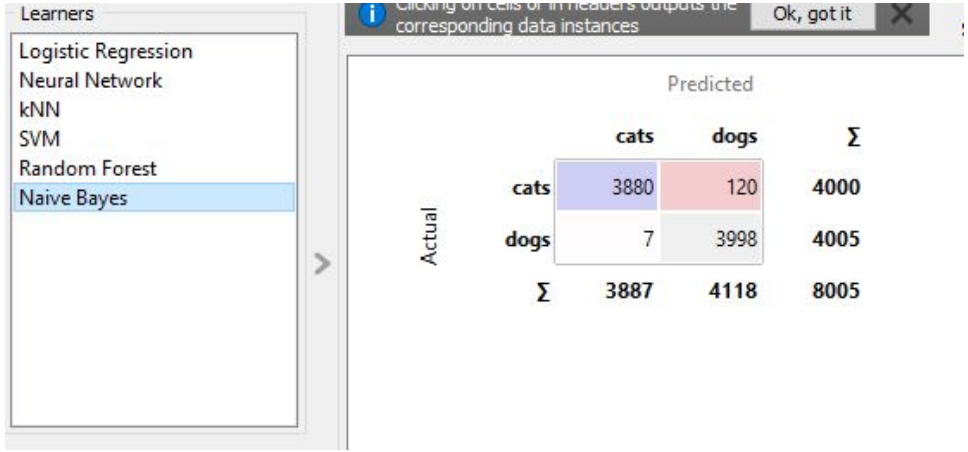
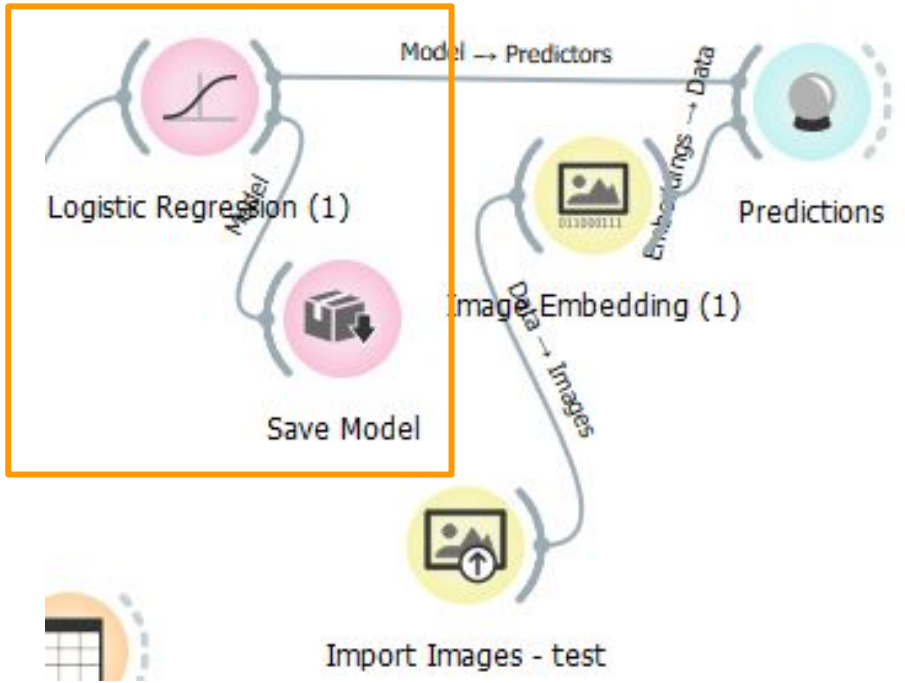


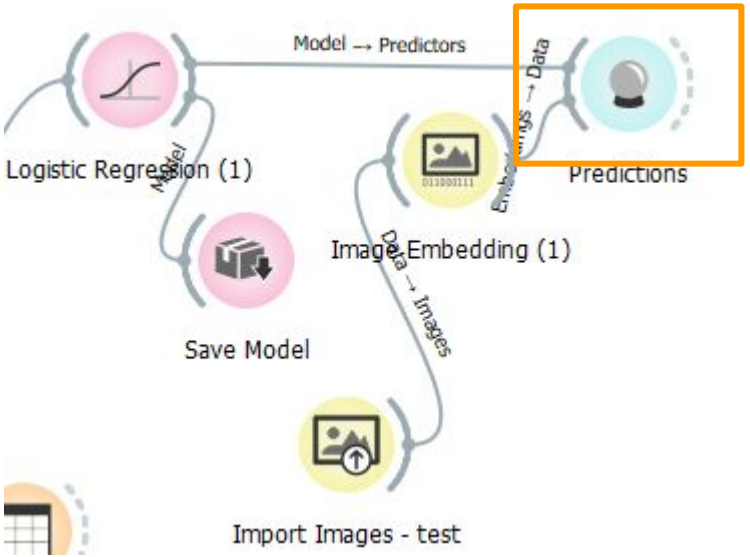
Image Embedding
Embeddings → Data
Learner
Logistic Regression
Test and Score
Evaluation Results
Selected Data → Data
Confusion Matrix
Image Viewer (1)
Learner
Neural Network

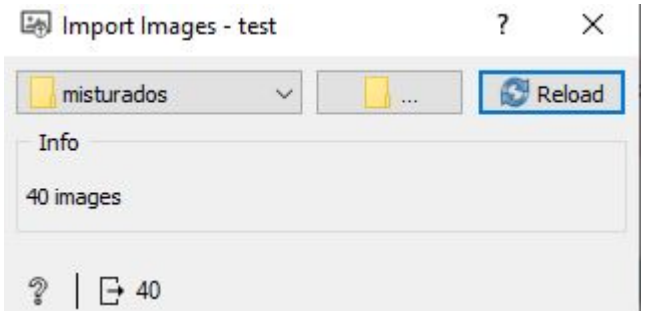

Learners
Logistic Regression
Neural Network
kNN
SVM
Random Forest
Naive Bayes
Ok, got it
corresponding data instances
Predicted
cats
dogs
Σ
Actual
cats 3880 120 4000
dogs 7 3998 4005
Σ 3887 4118 8005

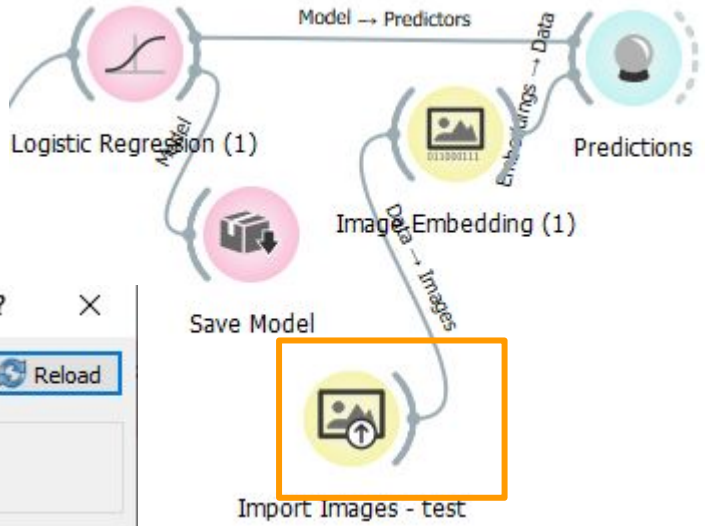
Model


Model → Predictors
Logistic Regression (1)
Model
Save Model
Predictions
Embeddings → Data
Image Embedding (1)
Data → Images
Import Images - test

Predictor


Model → Predictors
Data → Data
Logistic Regression (1)
Model
Predictions
Image Embedding (1)
Data → Images
Save Model
Import Images - test


Import Images - test
misturados
...
Reload
Info
40 images
40


Model → Predictors
Data
Logistic Regression (1)
Model
Predictions
Image Embedding (1)
Data → Images
Save Model
Import Images - test

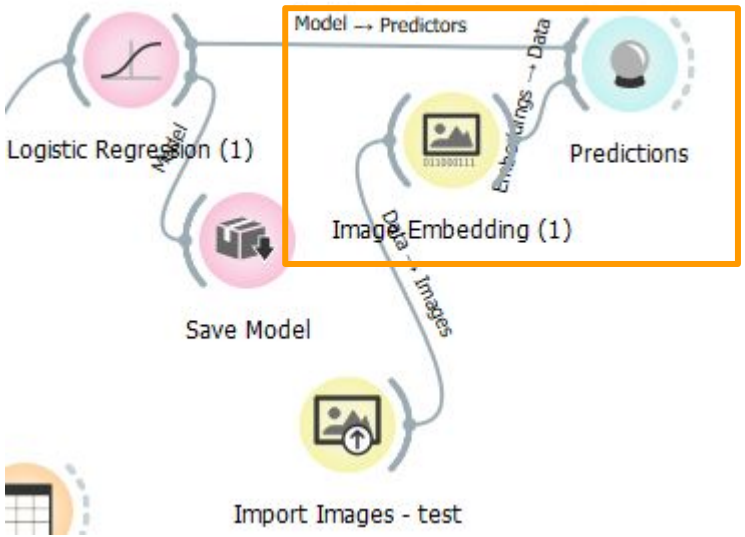
Predictions

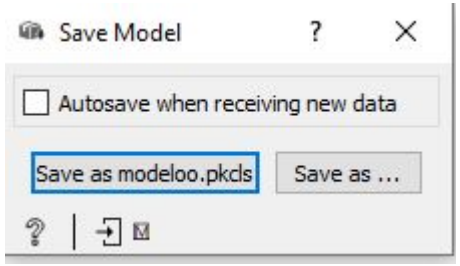
Show probabilities for

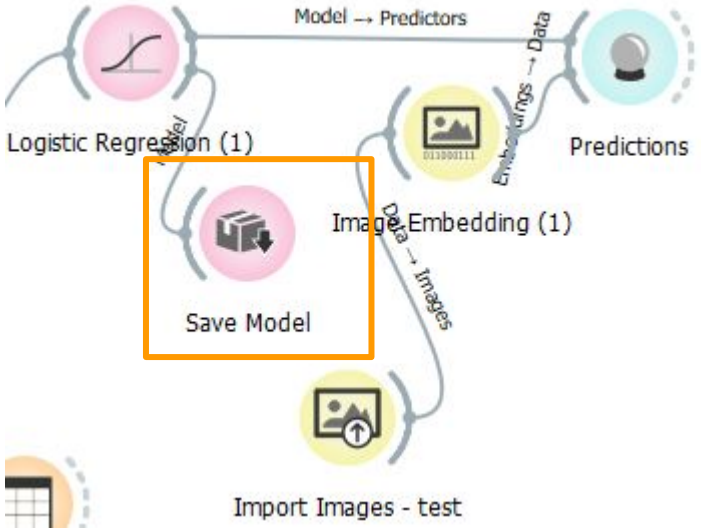
cats
dogs

Restore Original Order

	Logistic Regression	image name	image	size
22	cats	cat.435	cat.435.jpg	15110
23	dogs	dog.4072	dog.4072.jpg	22541
24	dogs	dog.4073	dog.4073.jpg	16045
25	dogs	dog.4074	dog.4074.jpg	25053
26	dogs	dog.4075	dog.4075.jpg	30518
27	dogs	dog.4076	dog.4076.jpg	20626
28	dogs	dog.4077	dog.4077.jpg	24940
29	dogs	dog.4080	dog.4080.jpg	31661
30	dogs	dog.4081	dog.4081.jpg	20310
31	dogs	dog.4082	dog.4082.jpg	29628
32	dogs	dog.4083	dog.4083.jpg	13272
33	dogs	dog.4084	dog.4084.jpg	14323
34	dogs	dog.4085	dog.4085.jpg	25691
35	dogs	dog.4088	dog.4088.jpg	37892
36	dogs	dog.4089	dog.4089.jpg	14107
37	dogs	dog.4090	dog.4090.jpg	6497
38	dogs	dog.4091	dog.4091.jpg	7081
39	dogs	dog.4092	dog.4092.jpg	18295
40	dogs	dog.4093	dog.4093.jpg	20787



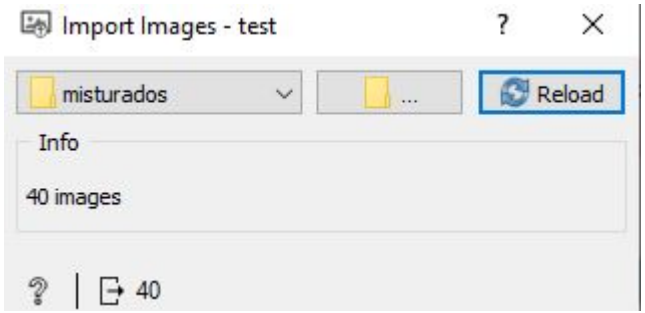

Save Model
Autosave when receiving new data
Save as modeloo.pkcls
Save as ...

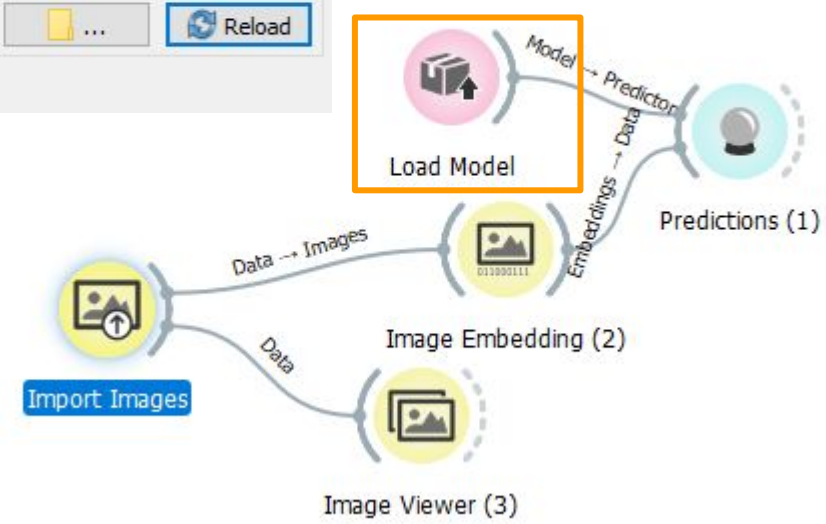

Model → Predictors
Logistic Regression (1)
Predictions
Image Embedding (1)
Save Model
Data → Images
Import Images - test

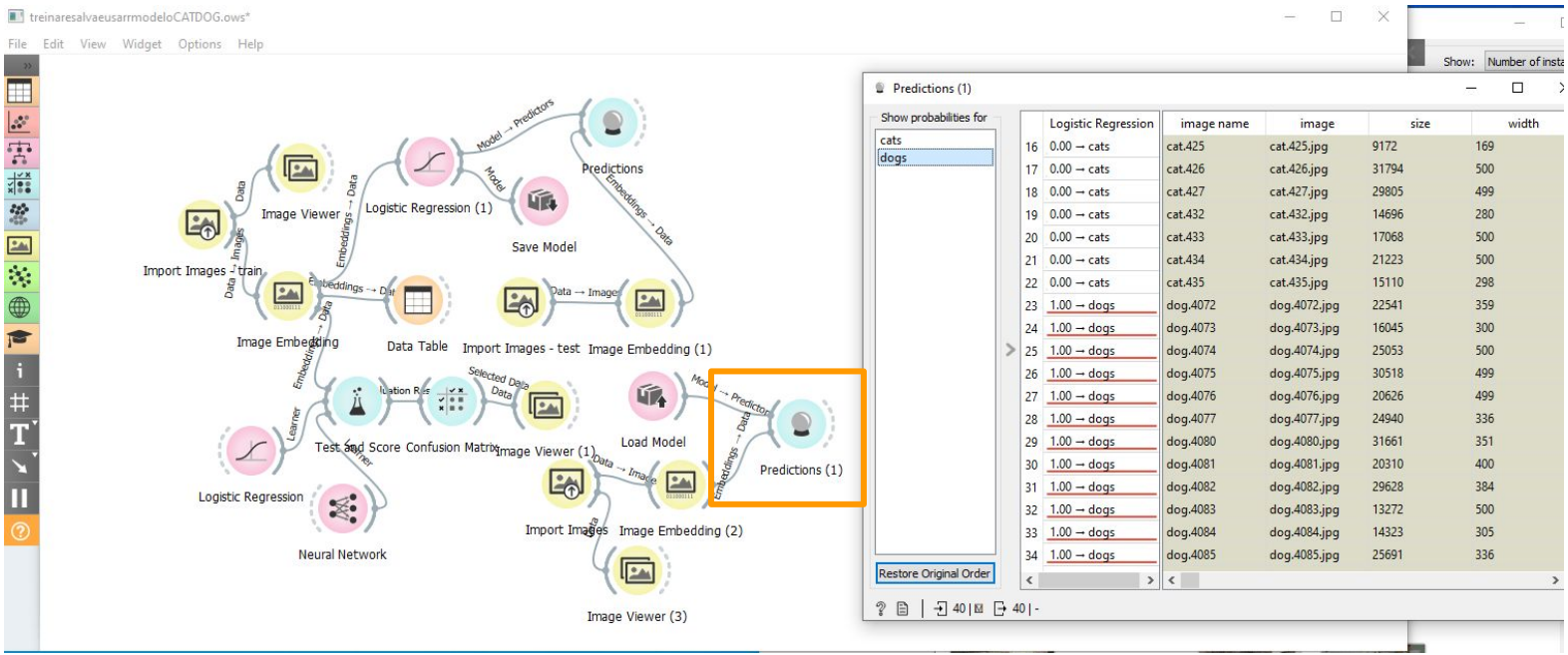
After trained....

use

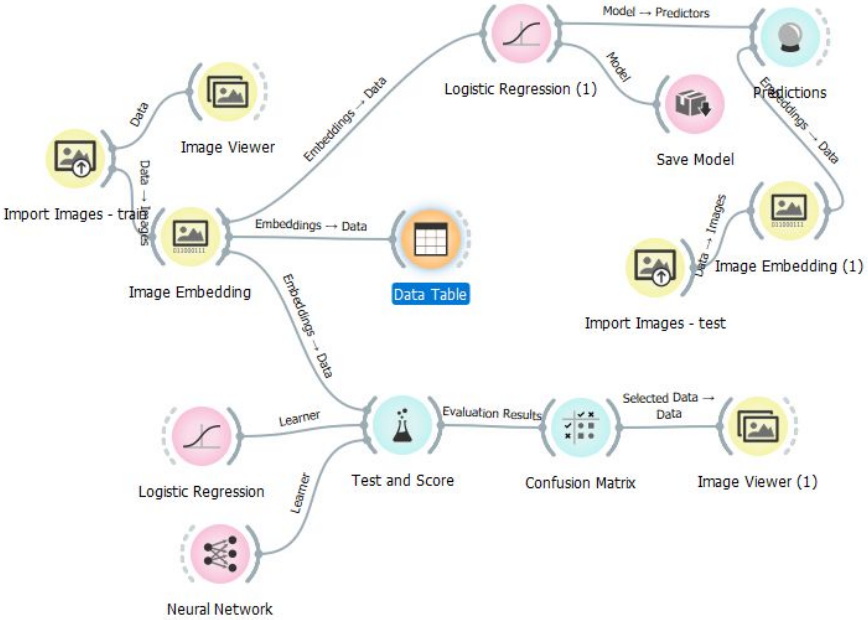
load model and use

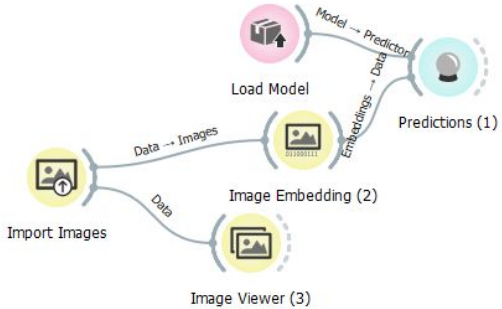





treinaresalvaeusarrmodeloCATDOG.ows*
File Edit View Widget Options Help
Import Images - train
Image Viewer
Logistic Regression (1)
Predictions
Save Model
Image Embedding
Data Table
Import Images - test
Image Embedding (1)
Test and Score
Confusion Matrix
Image Viewer (1)
Load Model
Predictions (1)
Logistic Regression
Neural Network
Import Images
Image Embedding (2)
Image Viewer (3)
Predictions (1)
Show probabilities for
cats
dogs
Restore Original Order
Logistic Regression
image name
image
size
width
16 0.00 → cats cat.425 cat.425.jpg 9172 169
17 0.00 → cats cat.426 cat.426.jpg 31794 500
18 0.00 → cats cat.427 cat.427.jpg 29805 499
19 0.00 → cats cat.432 cat.432.jpg 14696 280
20 0.00 → cats cat.433 cat.433.jpg 17068 500
21 0.00 → cats cat.434 cat.434.jpg 21223 500
22 0.00 → cats cat.435 cat.435.jpg 15110 298
23 1.00 → dogs dog.4072 dog.4072.jpg 22541 359
24 1.00 → dogs dog.4073 dog.4073.jpg 16045 300
25 1.00 → dogs dog.4074 dog.4074.jpg 25053 500
26 1.00 → dogs dog.4075 dog.4075.jpg 30518 499
27 1.00 → dogs dog.4076 dog.4076.jpg 20626 499
28 1.00 → dogs dog.4077 dog.4077.jpg 24940 336
29 1.00 → dogs dog.4080 dog.4080.jpg 31661 351
30 1.00 → dogs dog.4081 dog.4081.jpg 20310 400
31 1.00 → dogs dog.4082 dog.4082.jpg 29628 384
32 1.00 → dogs dog.4083 dog.4083.jpg 13272 500
33 1.00 → dogs dog.4084 dog.4084.jpg 14323 305
34 1.00 → dogs dog.4085 dog.4085.jpg 25691 336

complete view





pause orange

coding an example - cat or dog

Colab + python + keras + matplotlib + numpy = dogs vc cats

2,000 JPG pictures of cats and dogs

```
base_dir = '/tmp/cats_and_dogs_filtered'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

# Directory with our training cat pictures
train_cats_dir = os.path.join(train_dir, 'cats')

# Directory with our training dog pictures
train_dogs_dir = os.path.join(train_dir, 'dogs')

# Directory with our validation cat pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')

# Directory with our validation dog pictures
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
```

```
[7] train_cat_fnames = os.listdir(train_c
    print(train_cat_fnames[:10])

    train_dog_fnames = os.listdir(train_c
    train_dog_fnames.sort()
    print(train_dog_fnames[:10])
```

```
['cat.427.jpg', 'cat.256.jpg', 'cat.4
['dog.0.jpg', 'dog.1.jpg', 'dog.10.jp
```

Let's find out the total number of cat and dog ir

```
print('total training cat images:', l
print('total training dog images:', l
print('total validation cat images:',
print('total validation dog images:',
```

```
total training cat images: 1000
total training dog images: 1000
total validation cat images: 500
total validation dog images: 500
```

```
[11] from tensorflow.keras import layers
     from tensorflow.keras import Model
```

```
# Our input feature map is 150x150x3: 150x1
# the three color channels: R, G, and B
img_input = layers.Input(shape=(150, 150, 3

# First convolution extracts 16 filters tha
# Convolution is followed by max-pooling la
x = layers.Conv2D(16, 3, activation='relu')
x = layers.MaxPooling2D(2)(x)

# Second convolution extracts 32 filters th
# Convolution is followed by max-pooling la
x = layers.Conv2D(32, 3, activation='relu')
x = layers.MaxPooling2D(2)(x)

# Third convolution extracts 64 filters tha
# Convolution is followed by max-pooling la
x = layers.Conv2D(64, 3, activation='relu')
x = layers.MaxPooling2D(2)(x)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

# Flow training images in batches of 20 using train_datagen generat
train_generator = train_datagen.flow_from_directory(
        train_dir,  # This is the source directory for training ima
        target_size=(150, 150),  # All images will be resized to 15
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary lab
        class_mode='binary')

# Flow validation images in batches of 20 using val_datagen generat
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

Training

Let's train on all 2,000 images available, for 15 epochs, and validate on all 1,000 validation images.

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,  # 2000 images = batch_size * steps
      epochs=15,
      validation_data=validation_generator,
      validation_steps=50,  # 1000 images = batch_size * steps
      verbose=2)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/15
100/100 - 40s - loss: 0.8383 - acc: 0.5505 - val_loss: 0.6629 - val_acc: 0.6150
Epoch 2/15
100/100 - 8s - loss: 0.6415 - acc: 0.6420 - val_loss: 0.6087 - val_acc: 0.6670
Epoch 3/15
100/100 - 8s - loss: 0.5803 - acc: 0.7095 - val_loss: 0.6058 - val_acc: 0.6770
Epoch 4/15
100/100 - 8s - loss: 0.5173 - acc: 0.7775 - val_loss: 0.7317 - val_acc: 0.6690
Epoch 5/15
100/100 - 8s - loss: 0.4405 - acc: 0.8040 - val_loss: 0.7667 - val_acc: 0.6730
Epoch 6/15
100/100 - 8s - loss: 0.3744 - acc: 0.8405 - val_loss: 0.8034 - val_acc: 0.6960
Epoch 7/15
100/100 - 8s - loss: 0.2713 - acc: 0.8905 - val_loss: 0.8882 - val_acc: 0.6930
Epoch 8/15
100/100 - 8s - loss: 0.2026 - acc: 0.9190 - val_loss: 0.8854 - val_acc: 0.7070
Epoch 9/15
100/100 - 8s - loss: 0.1391 - acc: 0.9495 - val_loss: 1.3200 - val_acc: 0.6850
Epoch 10/15
100/100 - 8s - loss: 0.1057 - acc: 0.9695 - val_loss: 1.7593 - val_acc: 0.6970
Epoch 11/15
```

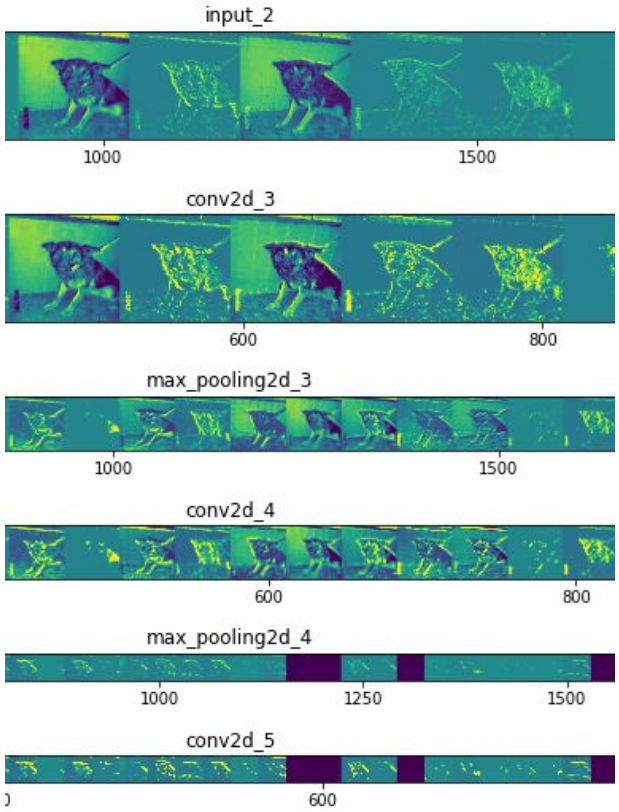
Visualizing Intermediate Representations

To get a feel for what kind of features our convnet has learned, one fun thing to do is to visualize h
through the convnet.

Let's pick a random cat or dog image from the training set, and then generate a figure where each
the row is a specific filter in that output feature map. Rerun this cell to generate intermediate repre

```
import numpy as np
import random
from tensorflow.keras.preprocessing.image import img_to_array, load_img

# Let's define a new Model that will take an image as input, and will output
# intermediate representations for all layers in the previous model after
# the first.
successive_outputs = [layer.output for layer in model.layers[1:]]
visualization_model = Model(img_input, successive_outputs)

# Let's prepare a random input image of a cat or dog from the training set.
cat_img_files = [os.path.join(train_cats_dir, f) for f in train_cat_fnames]
dog_img_files = [os.path.join(train_dogs_dir, f) for f in train_dog_fnames]
img_path = random.choice(cat_img_files + dog_img_files)

img = load_img(img_path, target_size=(150, 150))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255
```

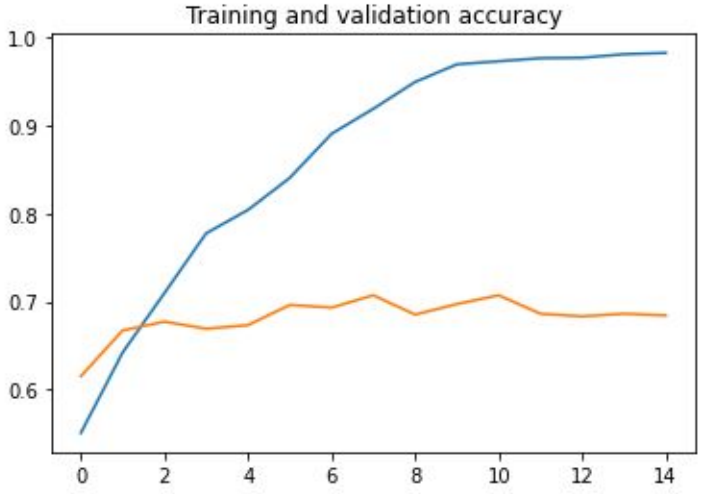
er.py:43: RuntimeWarning: invalid value encounte



Evaluating Accuracy and Loss for the Model

Let's plot the training/validation accuracy and loss as collected du

```
# Retrieve a list of accuracy results on training and
# sets for each training epoch
acc = history.history['acc']
val_acc = history.history['val_acc']

# Retrieve a list of list results on training and vali
# sets for each training epoch
loss = history.history['loss']
val_loss = history.history['val_loss']

# Get number of epochs
epochs = range(len(acc))

# Plot training and validation accuracy per epoch
plt.plot(epochs, acc)
plt.plot(epochs, val_acc)
plt.title('Training and validation accuracy')

plt.figure()

# Plot training and validation loss per epoch
plt.plot(epochs, loss)
plt.plot(epochs, val_loss)
plt.title('Training and validation loss')
```


Training and validation accuracy
1.0
0.9
0.8
0.7
0.6
0
2
4
6
8
10
12
14


Training and validation loss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14

Aprenda no colab:

Exercise 1: Building a Convnet from Scratch

Exercise 2: Reducing Overfitting

Feature Extraction and Fine-Tuning

little more

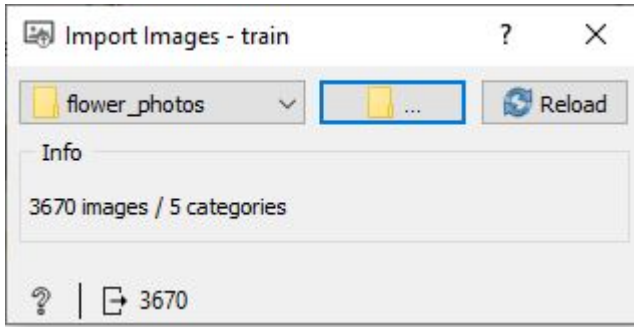
With flowers?

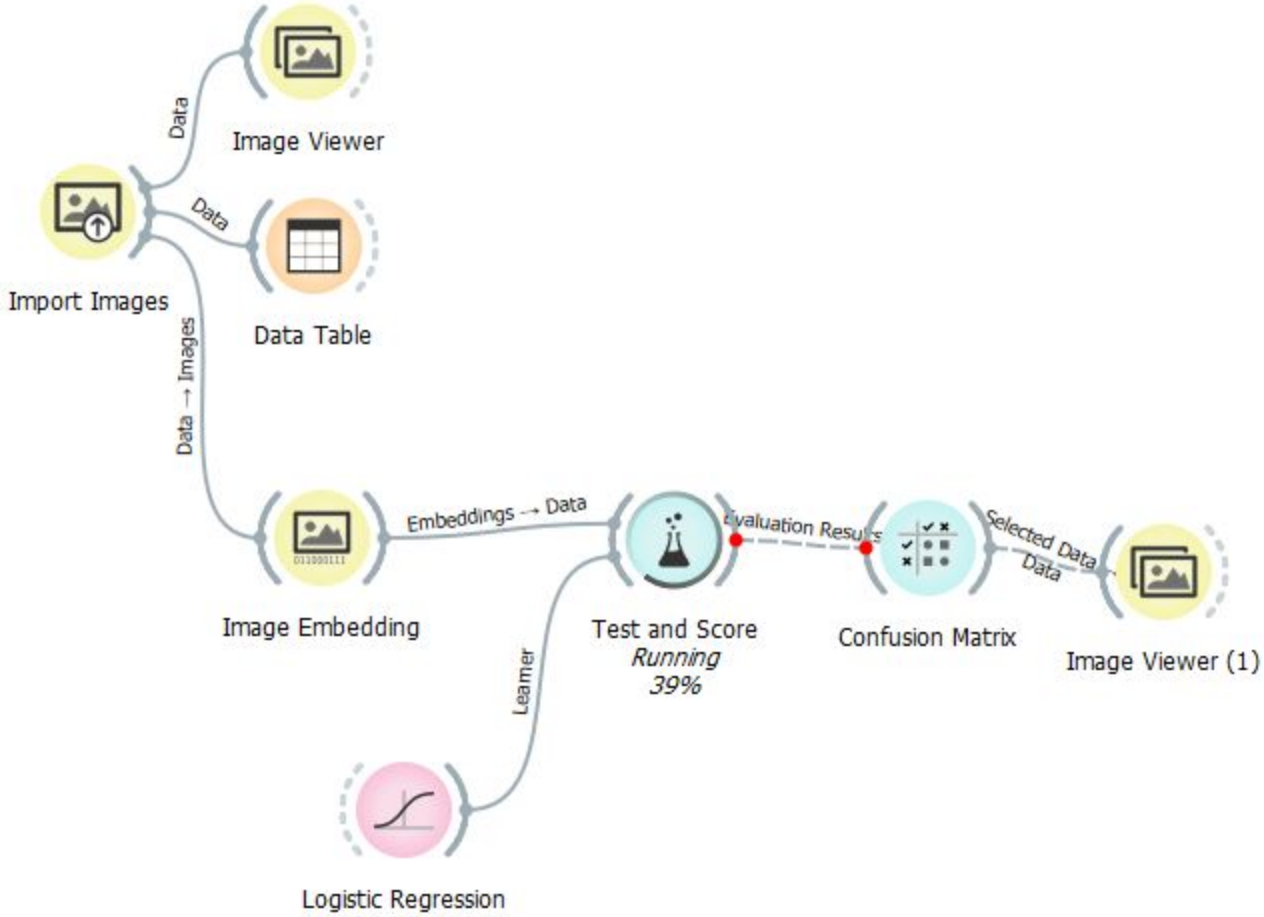
Other

try yourself with other example

now, with this imagens

https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz




Data
Image Viewer
Data
Import Images
Data Table
Data → Images
Embeddings → Data
Image Embedding
Test and Score
Running
39%
Evaluation Results
Confusion Matrix
Selected Data
Data
Image Viewer (1)
Learner
Logistic Regression

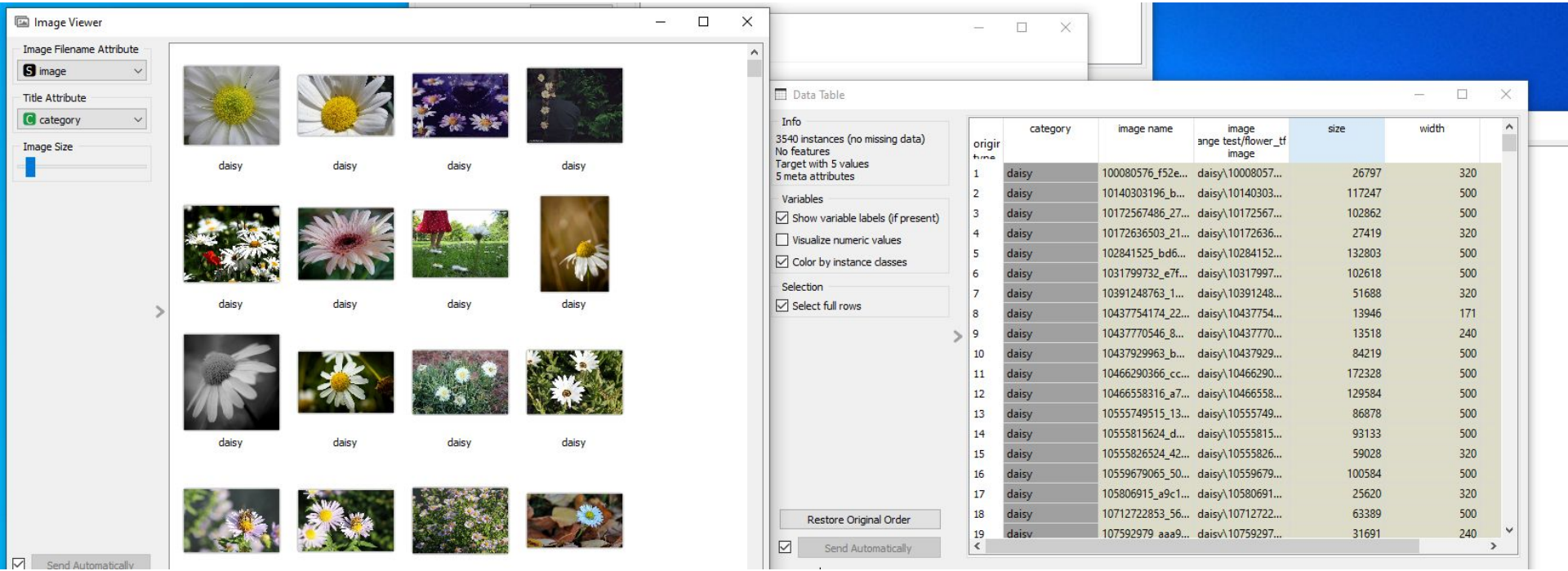


Image Viewer
Image Filename Attribute
image
Title Attribute
category
Image Size
daisy
Send Automatically
Data Table
Info
3540 instances (no missing data)
No features
Target with 5 values
5 meta attributes
Variables
Show variable labels (if present)
Visualize numeric values
Color by instance classes
Selection
Select full rows
Restore Original Order
Send Automatically
category
image name
image ange test/flower_tf image
size
width
1 daisy 100080576_f52e... daisy\10008057... 26797 320
2 daisy 10140303196_b... daisy\10140303... 117247 500
3 daisy 10172567486_27... daisy\10172567... 102862 500
4 daisy 10172636503_21... daisy\10172636... 27419 320
5 daisy 102841525_bd6... daisy\10284152... 132803 500
6 daisy 1031799732_e7f... daisy\10317997... 102618 500
7 daisy 10391248763_1... daisy\10391248... 51688 320
8 daisy 10437754174_22... daisy\10437754... 13946 171
9 daisy 10437770546_8... daisy\10437770... 13518 240
10 daisy 10437929963_b... daisy\10437929... 84219 500
11 daisy 10466290366_cc... daisy\10466290... 172328 500
12 daisy 10466558316_a7... daisy\10466558... 129584 500
13 daisy 10555749515_13... daisy\10555749... 86878 500
14 daisy 10555815624_d... daisy\10555815... 93133 500
15 daisy 10555826524_42... daisy\10555826... 59028 320
16 daisy 10559679065_50... daisy\10559679... 100584 500
17 daisy 105806915_a9c1... daisy\10580691... 25620 320
18 daisy 10712722853_56... daisy\10712722... 63389 500
19 daisy 107592979_aaa9... daisy\10759297... 31691 240

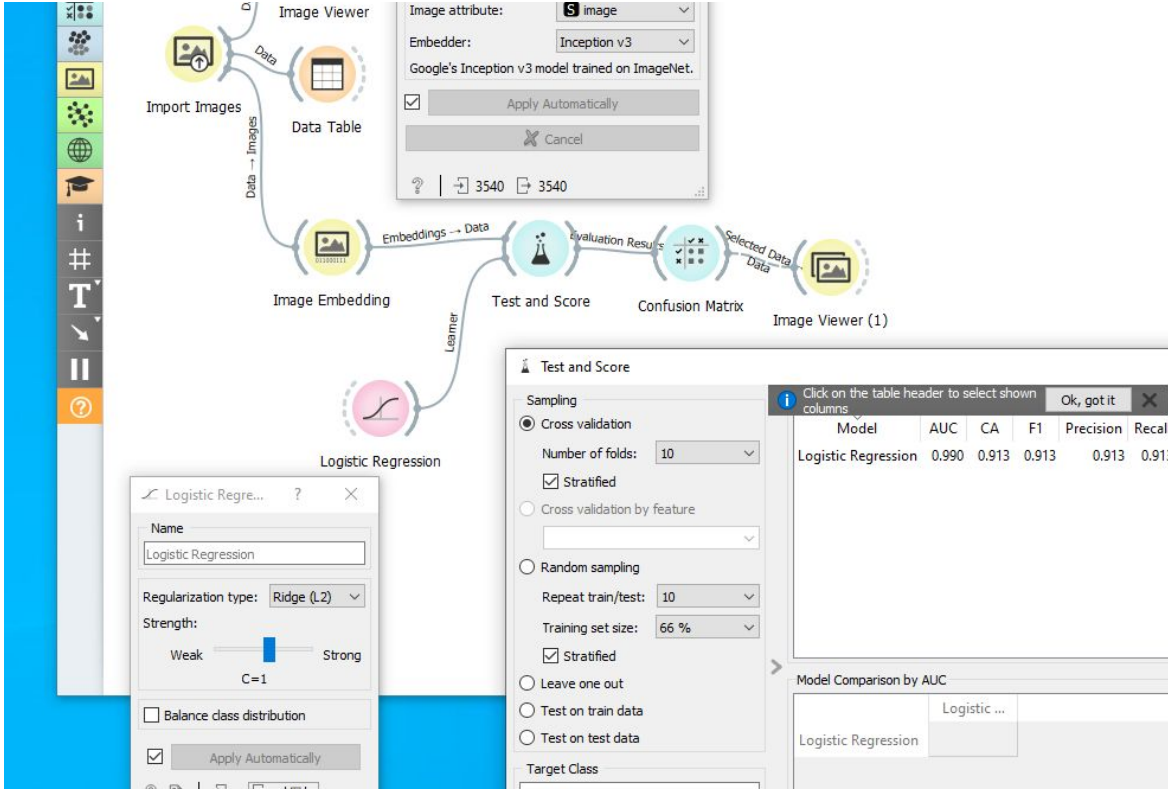
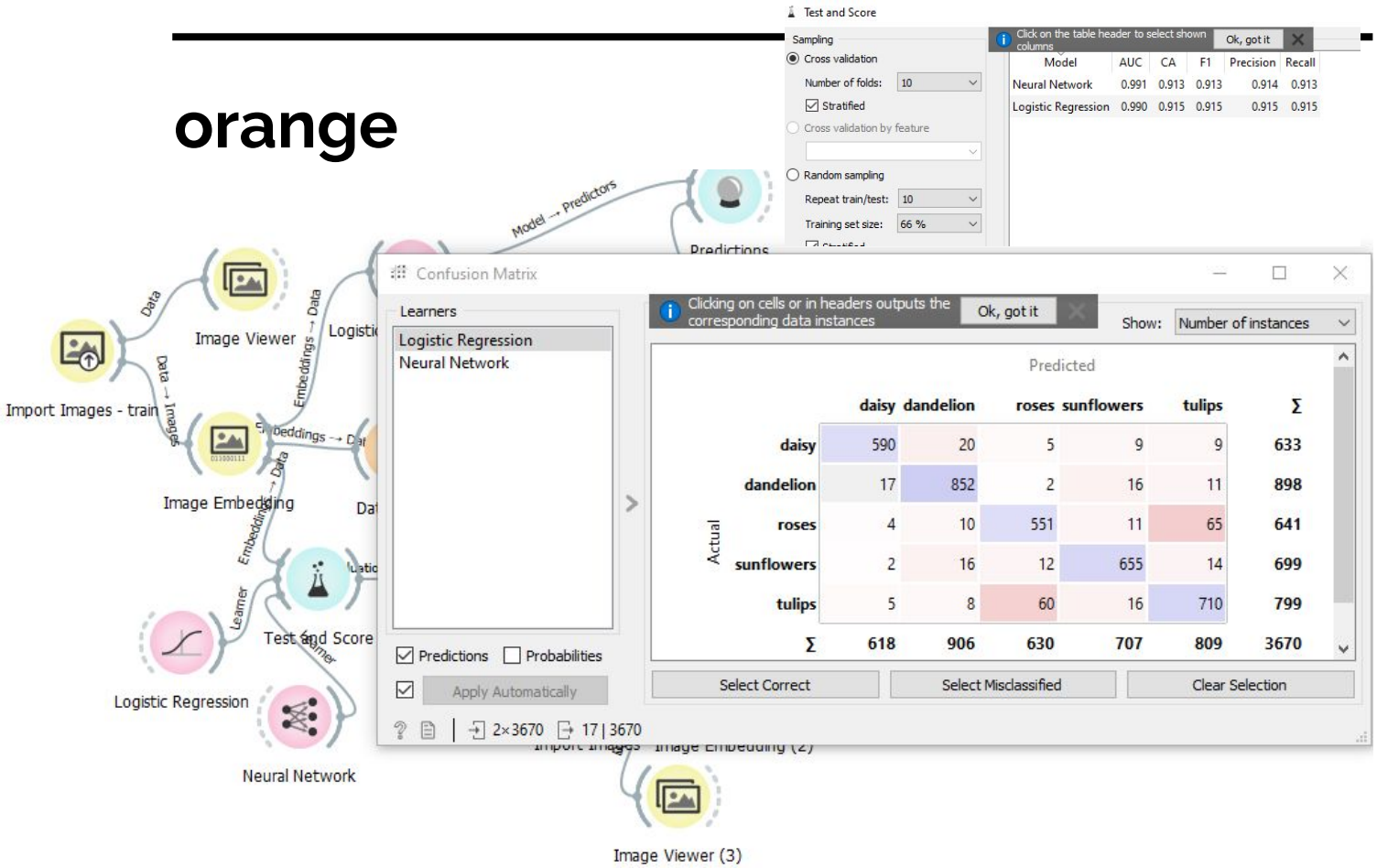


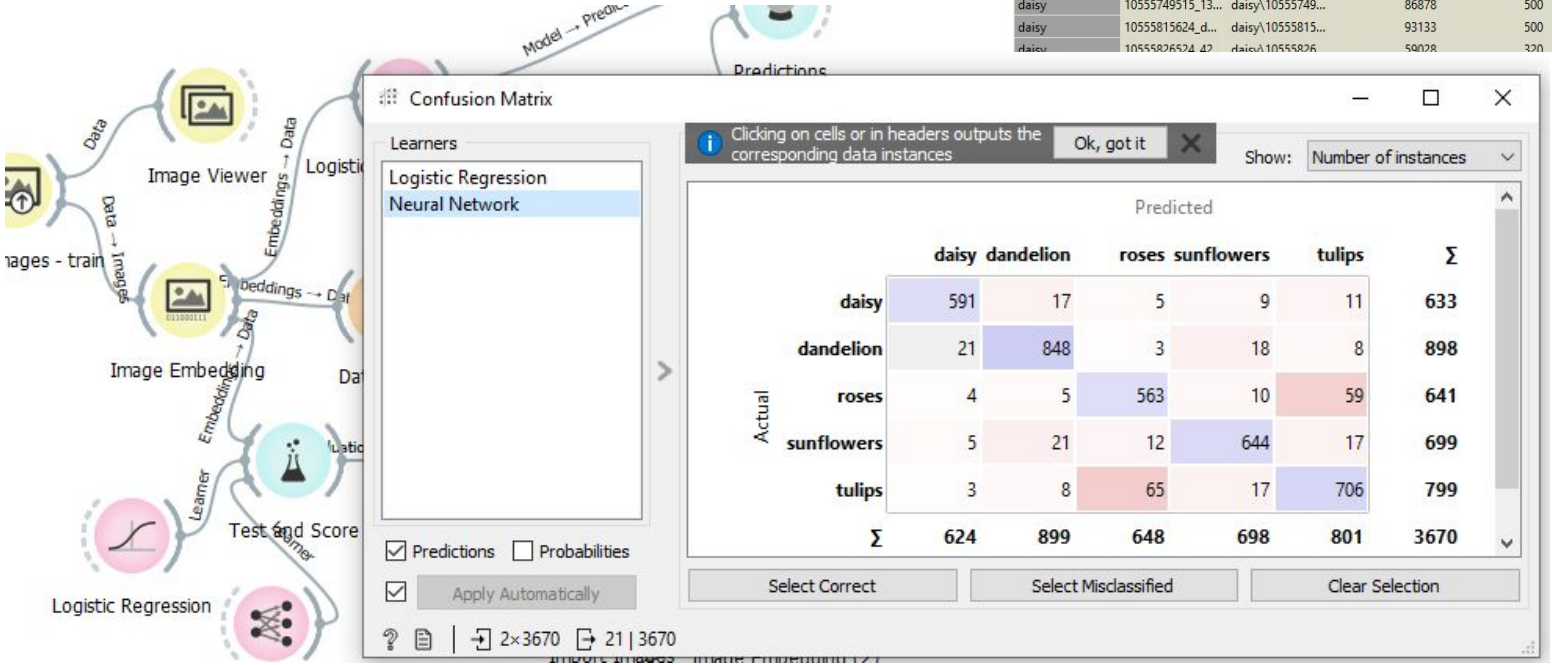
Image Viewer
Import Images
Data
Data Table
Data → Images
Image attribute:
image
Embedder:
Inception v3
Google's Inception v3 model trained on ImageNet.
Apply Automatically
Cancel
3540
3540
Embeddings → Data
Image Embedding
Test and Score
Evaluation Results
Confusion Matrix
Selected Data
Data
Image Viewer (1)
Learner
Logistic Regression
Logistic Regre...
Name
Logistic Regression
Regularization type:
Ridge (L2)
Strength:
Weak
Strong
C=1
Balance class distribution
Apply Automatically
Test and Score
Sampling
Cross validation
Number of folds:
10
Stratified
Cross validation by feature
Random sampling
Repeat train/test:
10
Training set size:
66 %
Stratified
Leave one out
Test on train data
Test on test data
Target Class
Click on the table header to select shown columns
Ok, got it
Model
AUC
CA
F1
Precision
Recal
Logistic Regression
0.990
0.913
0.913
0.913
0.91
Model Comparison by AUC
Logistic ...
Logistic Regression

orange


Test and Score
Sampling
Cross validation
Number of folds: 10
Stratified
Cross validation by feature
Random sampling
Repeat train/test: 10
Training set size: 66 %
Click on the table header to select shown columns
Ok, got it
Model AUC CA F1 Precision Recall
Neural Network 0.991 0.913 0.913 0.914 0.913
Logistic Regression 0.990 0.915 0.915 0.915 0.915
Confusion Matrix
Learners
Logistic Regression
Neural Network
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Show: Number of instances
Predicted
Actual
daisy dandelion roses sunflowers tulips Σ
daisy 590 20 5 9 9 633
dandelion 17 852 2 16 11 898
roses 4 10 551 11 65 641
sunflowers 2 16 12 655 14 699
tulips 5 8 60 16 710 799
Σ 618 906 630 707 809 3670
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection
2×3670
17 | 3670
Import Images - train
Image Viewer
Image Embedding
Test and Score
Logistic Regression
Neural Network
Predictions
Model → Predictors
Data
Data → Images
Embeddings → Data
Learner
Image Viewer (3)

orange

category	image name	image s/orange test/flowe image	size	width
daisy	1031799732_e7f...	daisy\10317997...	102618	500
daisy	10391248763_1...	daisy\10391248...	51688	320
daisy	10437754174_22...	daisy\10437754...	13946	171
daisy	10437770546_8...	daisy\10437770...	13518	240
daisy	10437929963_b...	daisy\10437929...	84219	500
daisy	10466290366_cc...	daisy\10466290...	172328	500
daisy	10466558316_a7...	daisy\10466558...	129584	500
daisy	10555749515_13...	daisy\10555749...	86878	500
daisy	10555815624_d...	daisy\10555815...	93133	500
daisy	10555826524_42	daisy\10555826	59028	320



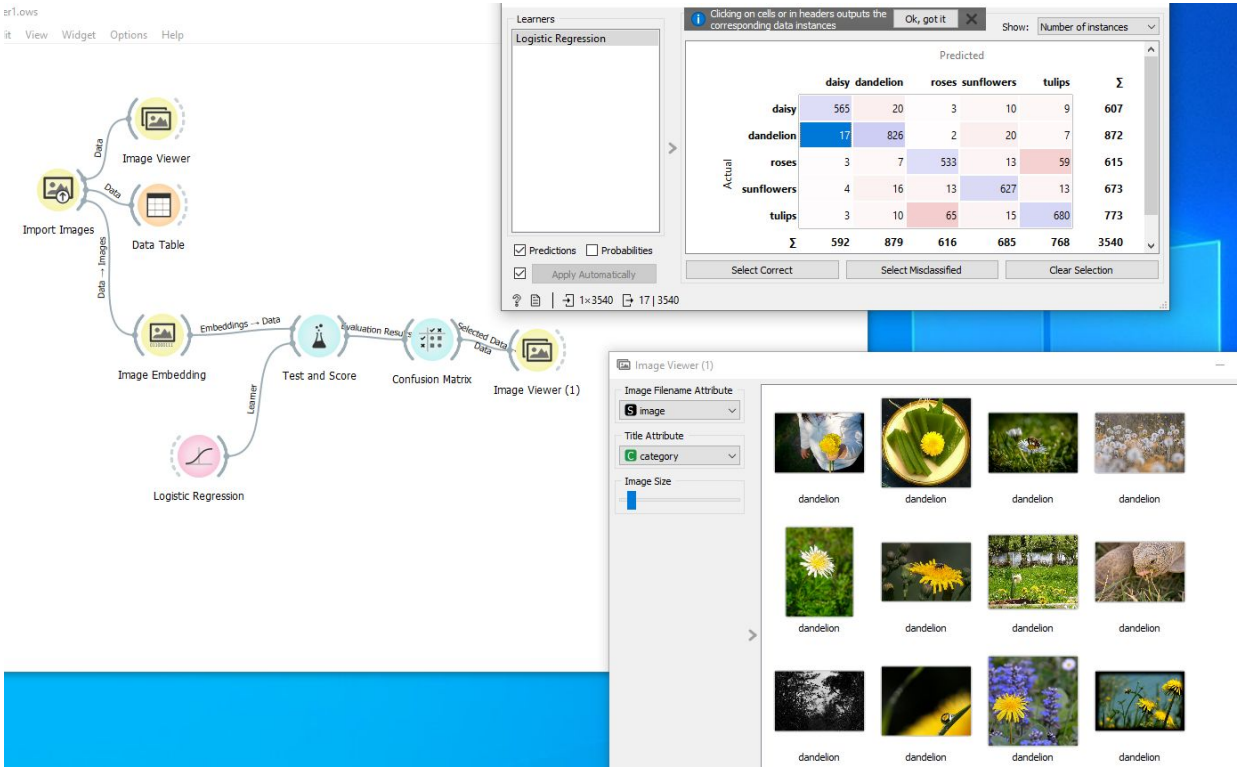


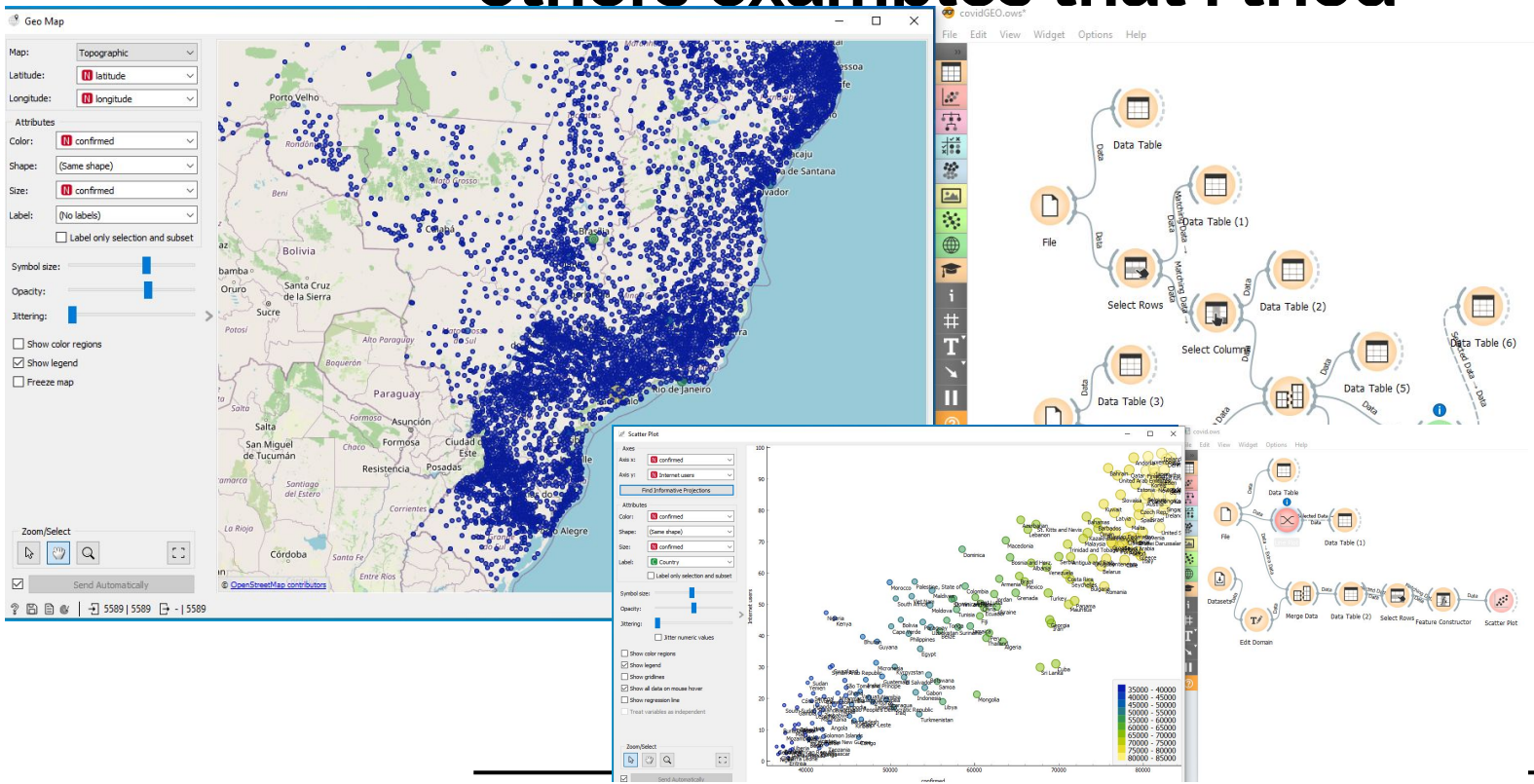
Image Viewer
Import Images
Data Table
Image Embedding
Test and Score
Confusion Matrix
Image Viewer (1)
Logistic Regression
Learners
Logistic Regression
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection
Predicted
daisy dandelion roses sunflowers tulips Σ
Actual
daisy 565 20 3 10 9 607
dandelion 17 826 2 20 7 872
roses 3 7 533 13 59 615
sunflowers 4 16 13 627 13 673
tulips 3 10 65 15 680 773
Σ 592 879 616 685 768 3540
Image Filename Attribute
image
Title Attribute
category
Image Size
dandelion

coding an example - flowers

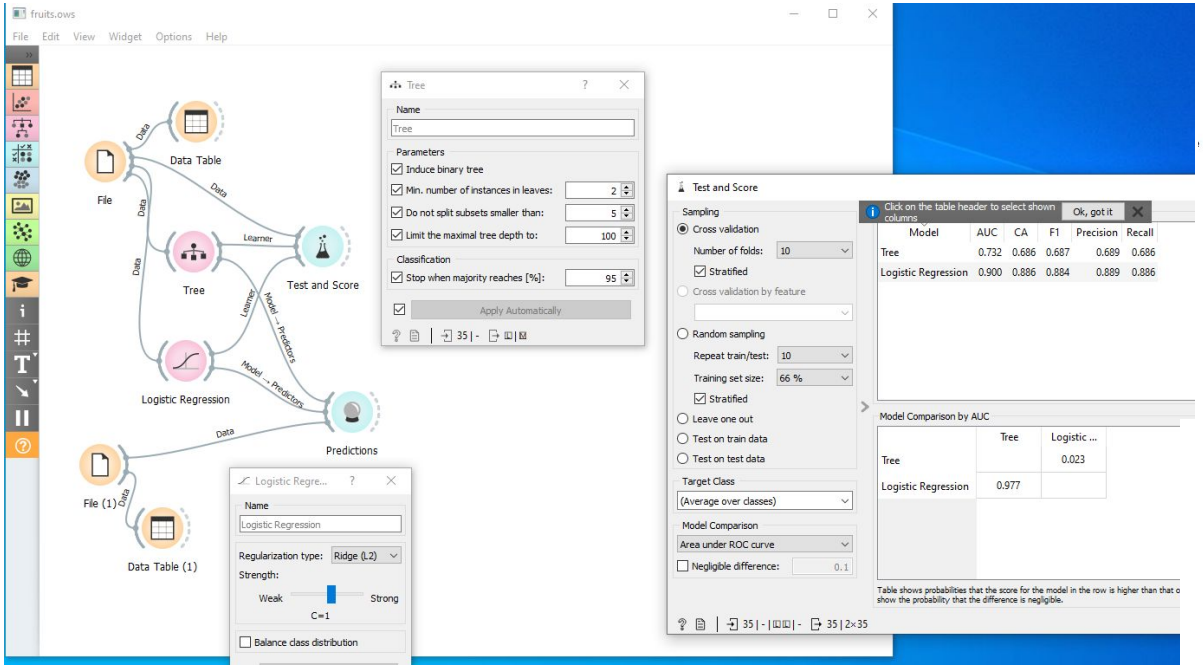
Colab + python + keras + matplotlib + numpy - aprendendo e classificando flores

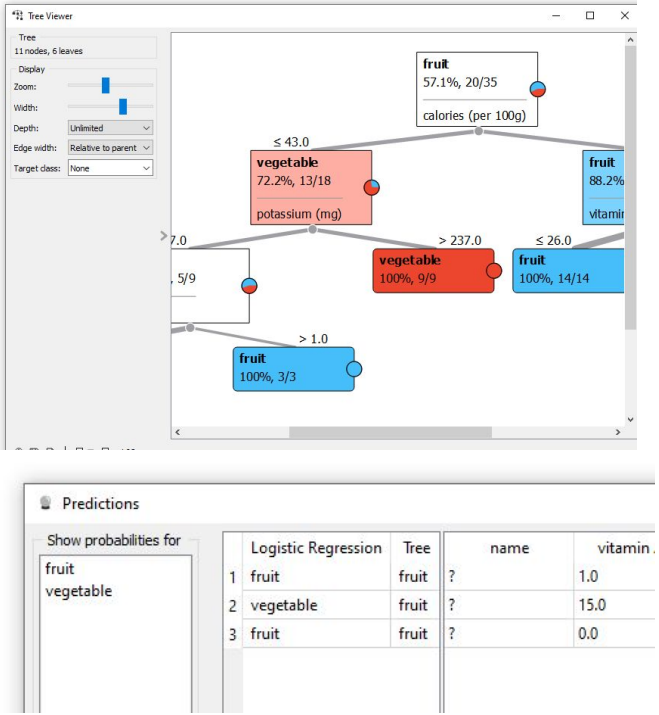
others

others examples that i tried

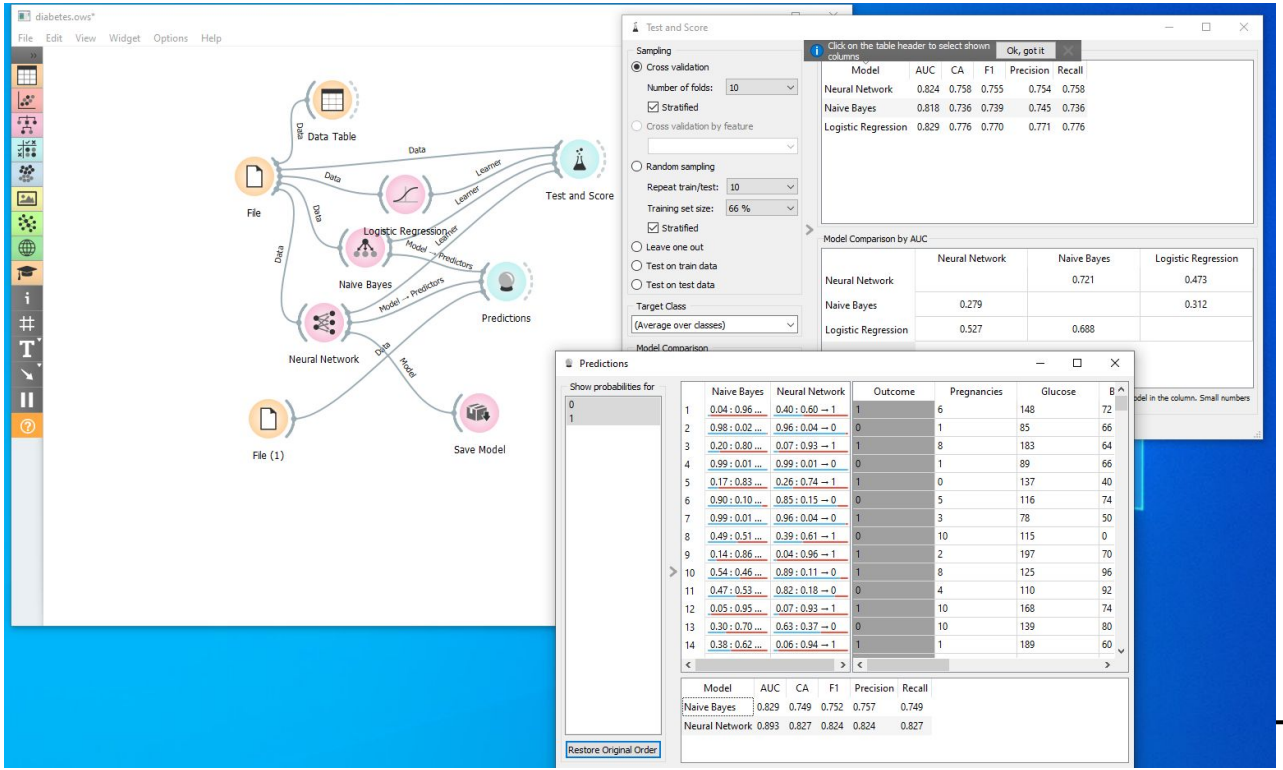


others examples that i tried

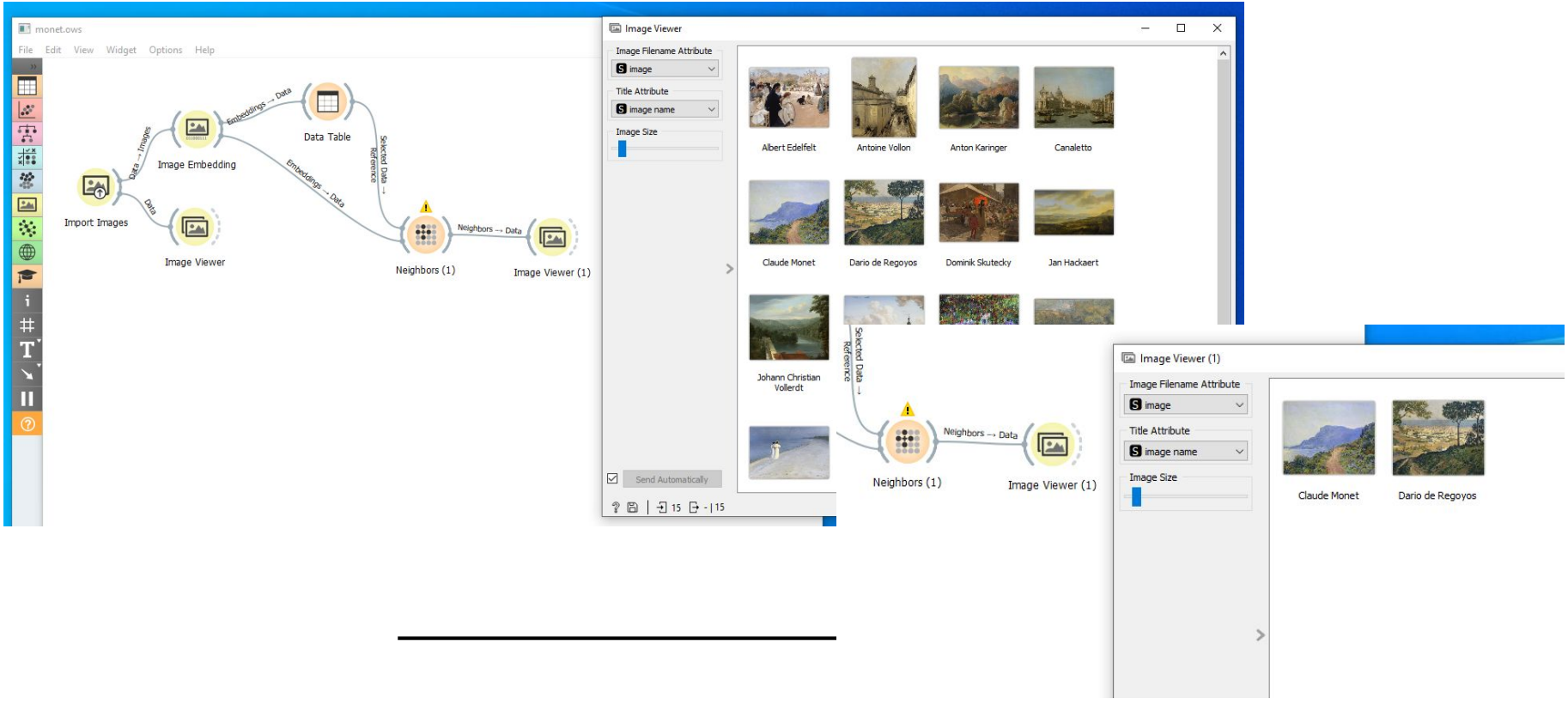




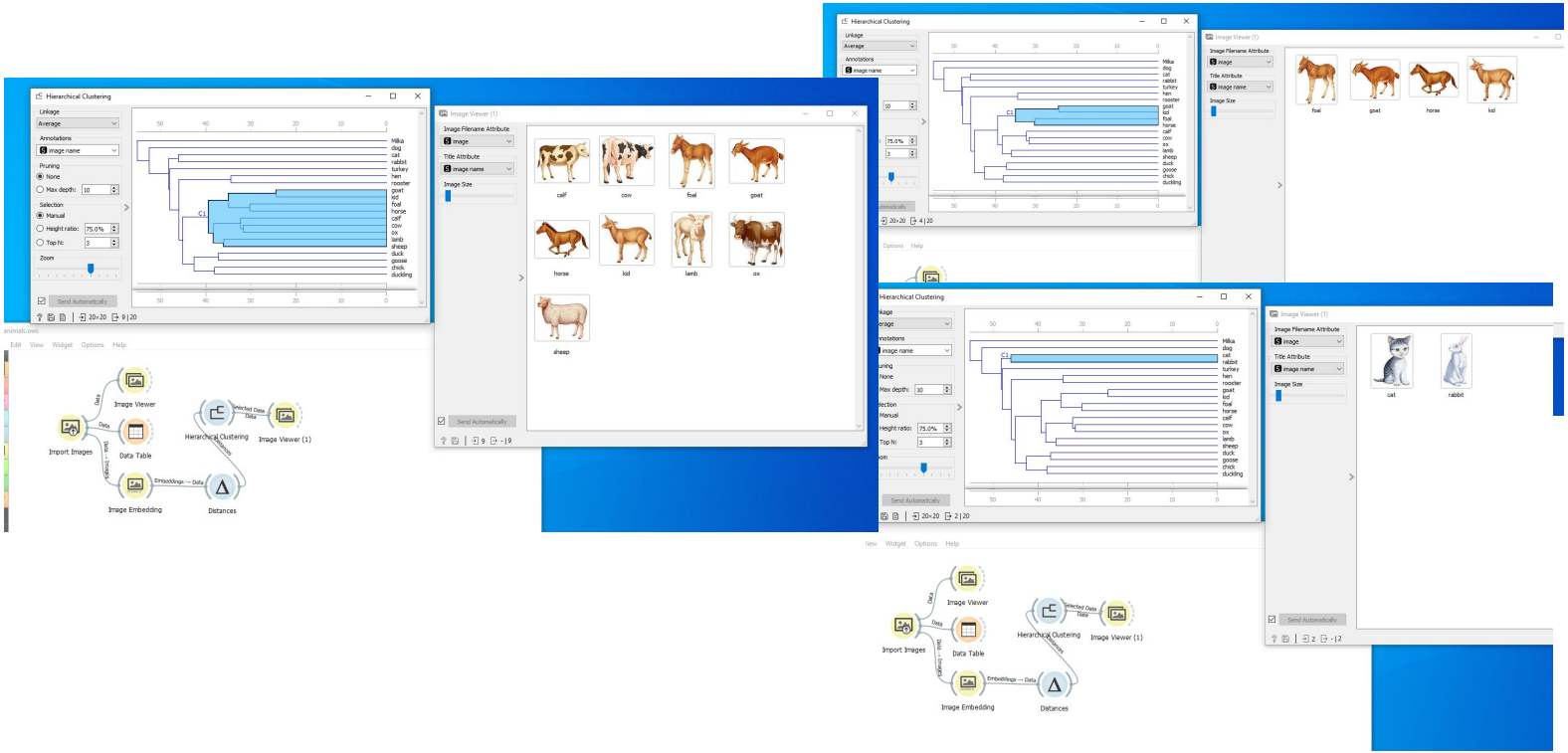
others examples that i tried



others examples that i tried



others examples that i tried



Conclusion

compare

KNN

Predicted

Actual	cats	dogs	Σ
cats	3957	43	4000
dogs	14	3991	4005
Σ	3971	4034	8005

Random Forest

Predicted

Actual	cats	dogs	Σ
cats	3951	49	4000
dogs	22	3983	4005
Σ	3973	4032	8005

LOGIC REGRESSION

Predicted

Actual	cats	dogs	Σ
cats	3972	28	4000
dogs	19	3986	4005
Σ	3991	4014	8005

SVM

Predicted

Actual	cats	dogs	
cats	3969	31	4000
dogs	18	3987	4005
Σ	3987	4018	8005

Naive Bayes

Predicted

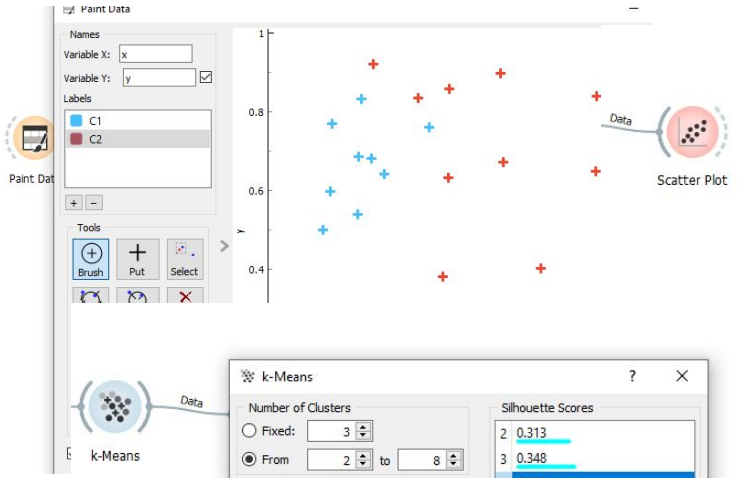
Actual	cats	dogs	Σ
cats	3880	120	4000
dogs	7	3998	4005
Σ	3887	4118	8005

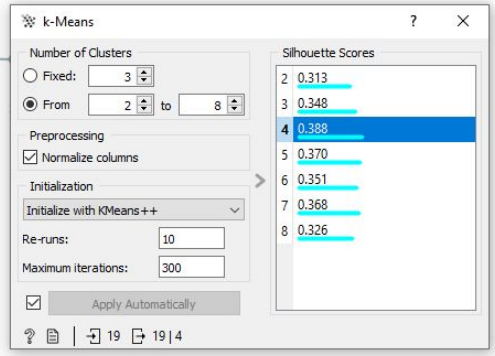
NEURAL NETWORK

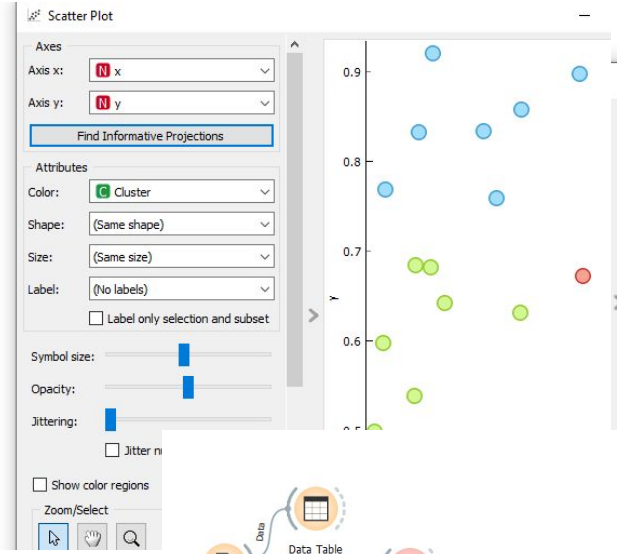
Predicted

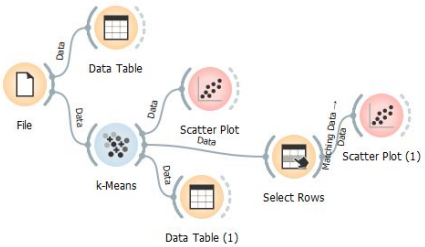
Actual	cats	dogs	Σ
cats	3971	29	4000
dogs	20	3985	4005
Σ	3991	4014	8005

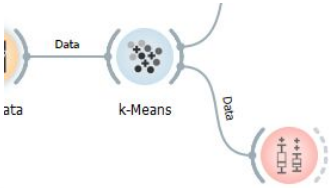
K-means

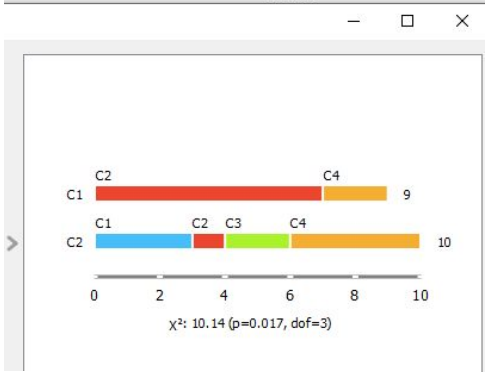

Paint Data
Names
Variable X: x
Variable Y: y
Labels
C1
C2
Tools
Brush
Put
Select
Data
Scatter Plot
k-Means


k-Means
Number of Clusters
Fixed: 3
From 2 to 8
Preprocessing
Normalize columns
Initialization
Initialize with KMeans++
Re-runs: 10
Maximum iterations: 300
Apply Automatically
Silhouette Scores
2 0.313
3 0.348
4 0.388
5 0.370
6 0.351
7 0.368
8 0.326

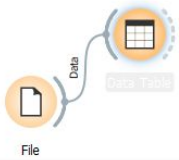

Scatter Plot
Axes
Axis x: x
Axis y: y
Find Informative Projections
Attributes
Color: Cluster
Shape: (Same shape)
Size: (Same size)
Label: (No labels)
Label only selection and subset
Symbol size:
Opacity:
Jittering:
Show color regions
Zoom/Select


Data Table
File
Scatter Plot
k-Means
Select Rows
Scatter Plot (1)
Data Table (1)

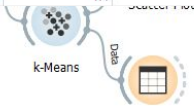

Data
k-Means
Data
Box Plot

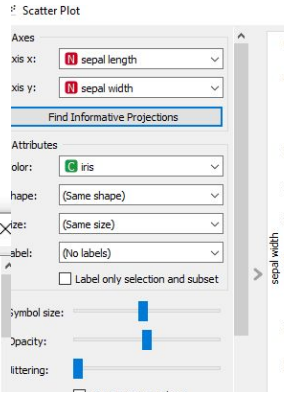

C1
C2
C4
9
C1
C2
C3
C4
10
0
2
4
6
8
10
χ²: 10.14 (p=0.017, dof=3)

Agrupamento - K-means - iris



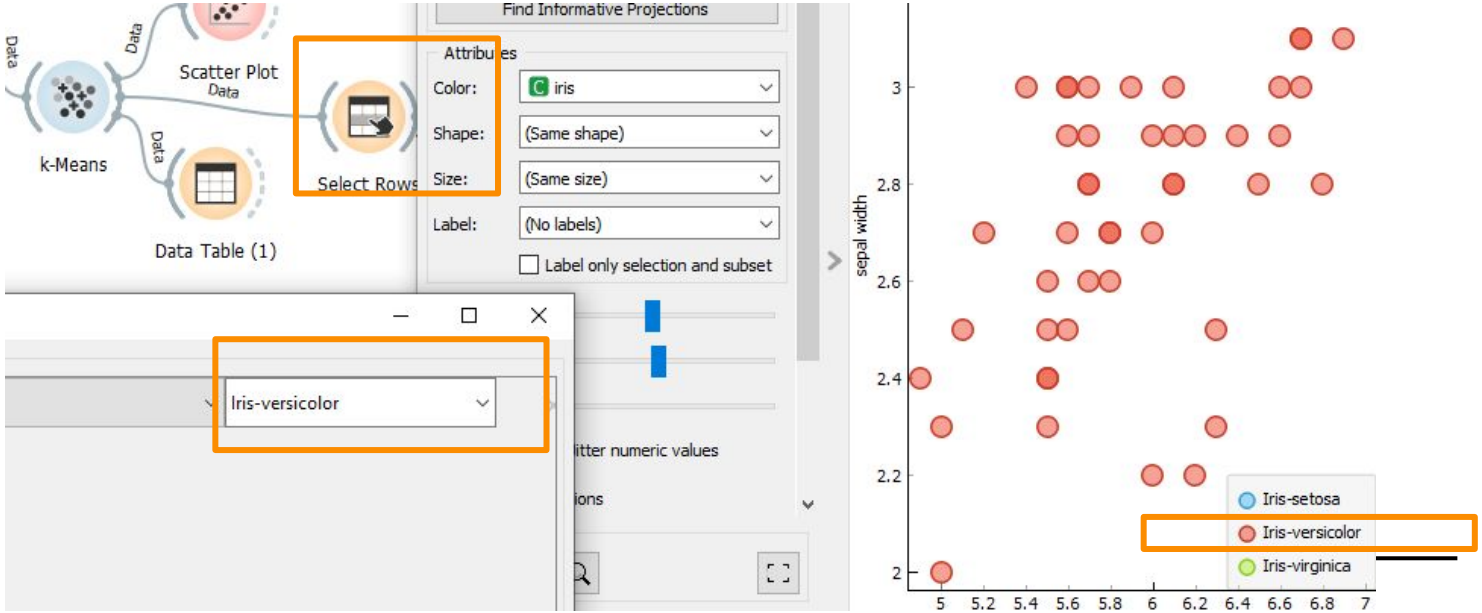
	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.1	3.5	1.4	0.2
2	Iris-setosa	4.9	3.0	1.4	0.2
3	Iris-setosa	4.7	3.2	1.3	0.2
4	Iris-setosa	4.6	3.1	1.5	0.2
5	Iris-setosa	5.0	3.6	1.4	0.2
6	Iris-setosa	5.4			
7	Iris-setosa	4.6			
8	Iris-setosa	5.0			





	iris	Cluster	Silhouette	sepal length	sepal width
1	Iris-setosa	C1	0.706706	5.1	3.5
2	Iris-setosa	C1	0.683687	4.9	3.0
3	Iris-setosa	C1	0.700571	4.7	3.2
4				4.6	3.1
5	Iris-setosa	C1	0.704669	5.0	3.6
6	Iris-setosa	C1	0.677309	5.4	3.9
7	Iris-setosa	C1	0.701974	4.6	3.4
8	Iris-setosa	C1	0.707426	5.0	3.4

Agrupamento - K-means - iris



iris

confusion matrix

Predicted

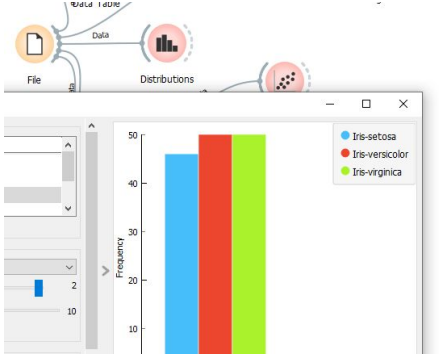
Actual	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	46	4	50
Iris-virginica	0	3	47	50
Σ	50	49	51	150

random forest e logic regression

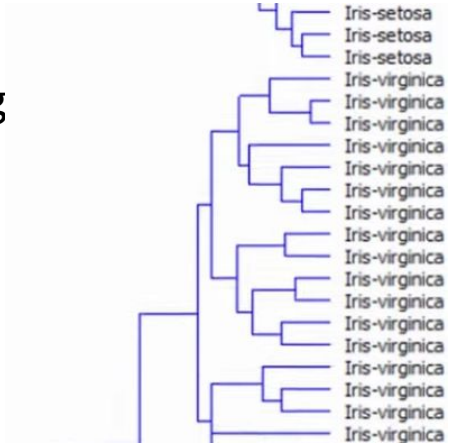
https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/regression.html#random-forest

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	0.970	0.960	0.960	0.962	0.960
RF Classification Learner	0.965	0.953	0.953	0.953	0.953

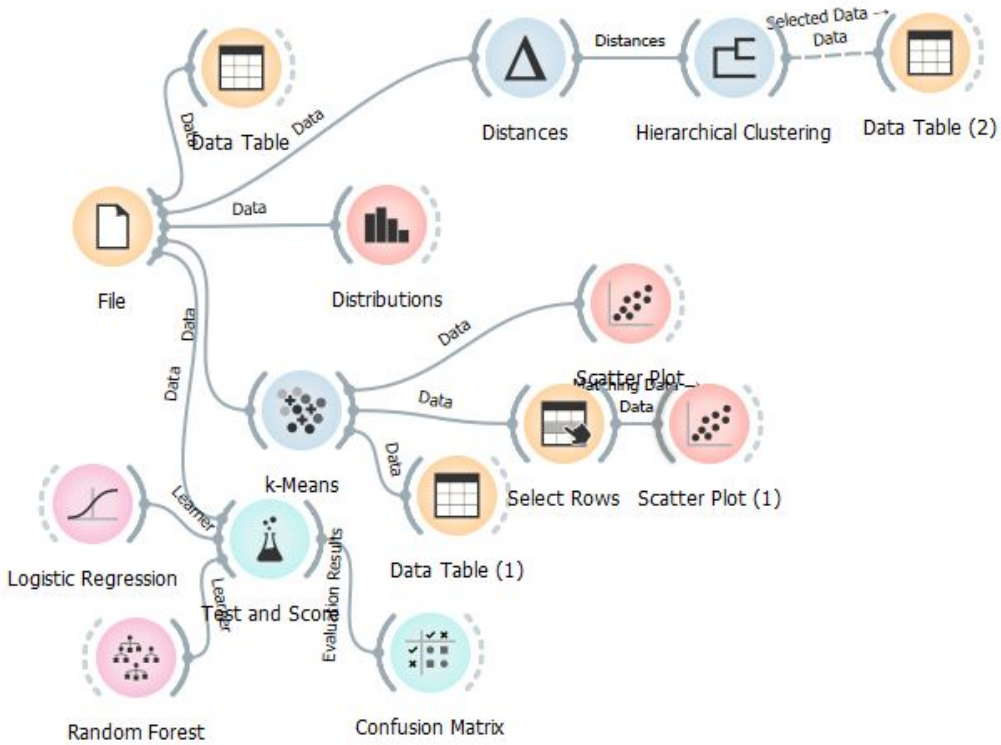
distributions



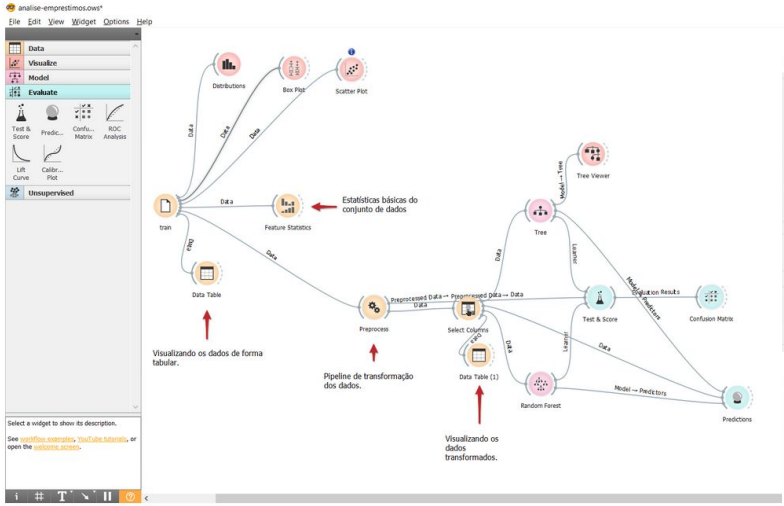
iris clustering



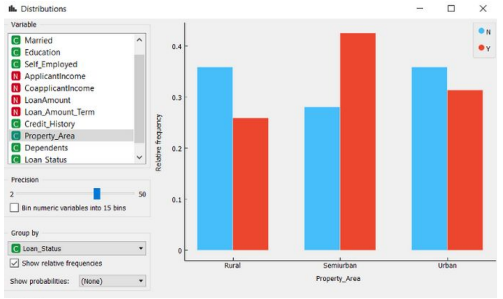
iris


Data Table
Distances
Hierarchical Clustering
Selected Data
Data
Data Table (2)
Distances
File
Distributions
Scatter Plot
k-Means
Select Rows
Matching Data
Scatter Plot (1)
Data Table (1)
Logistic Regression
Learner
Test and Scores
Evaluation Results
Random Forest
Confusion Matrix

example loan - orange



Conhecendo as previsões de cada um dos modelos



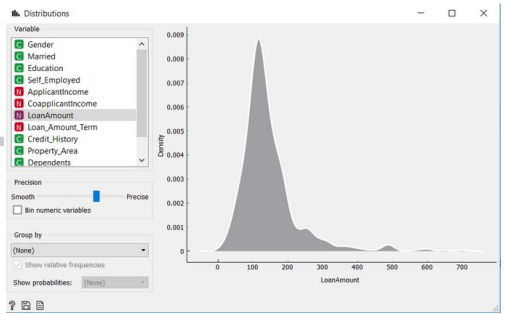
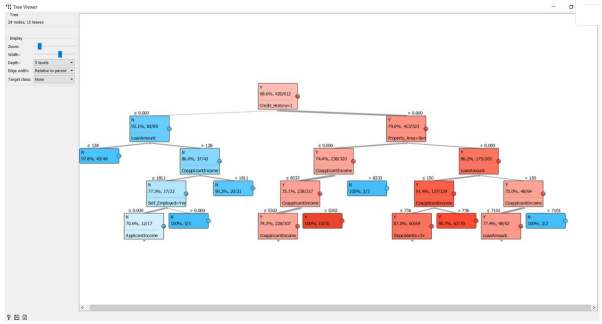
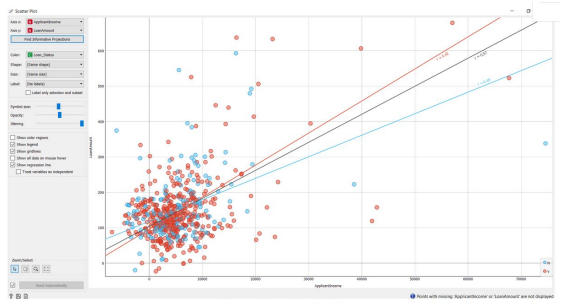


Gráfico de densidade do valor de empréstimo solicitado.



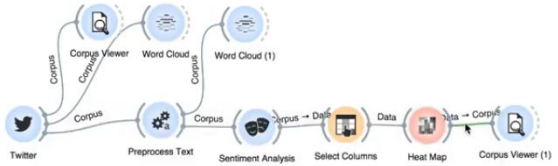


Relação entre renda e valor do empréstimo.

videos

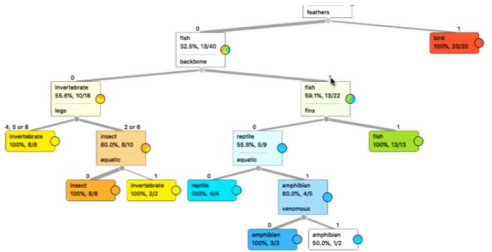
moods on twitter

https://www.youtube.com/watch?v=1stp41wz3LA



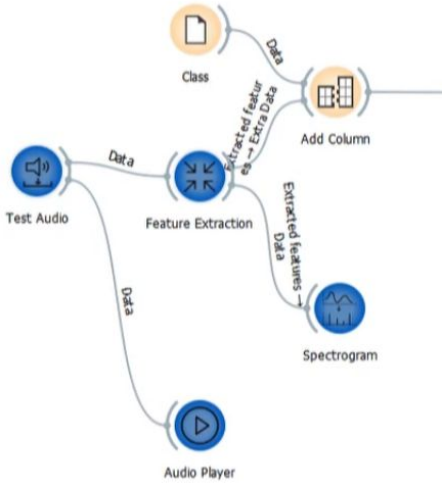
classify tree - fish - reptile - bird - insect

https://www.youtube.com/watch?v=eTe8r5tnfRA&list=PLZ3V9XyVA52_XSu059rS9eqi7P2nY6Qe9&index=7



instrument sound classifier

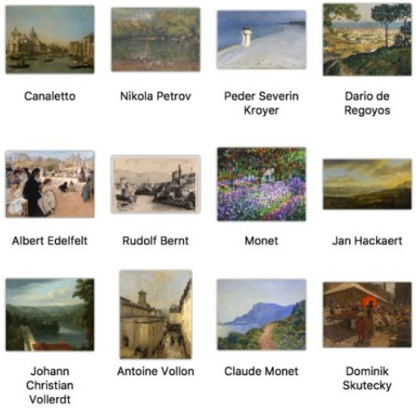
https://www.youtube.com/watch?v=BO0LUlMx6qs



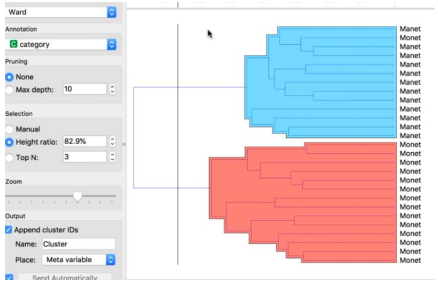
svm - unicamp - orange

https://www.youtube.com/watch?v=1KFx3861Djs

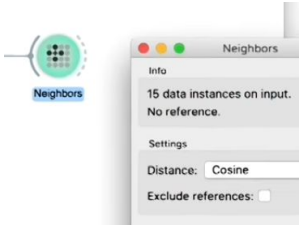
videos - image challenge



[monet or manet - inceptionv3 (google)](#)



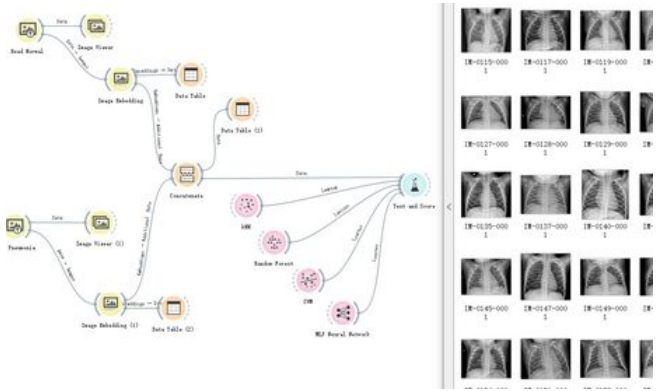
[the lost monet - model pre trained painters](#)



links

Detecting Pneumonia in Chest X-Ray Images under Orange Machine Learning/Deep Learning Platform

https://towardsdatascience.com/detecting-pneumonia-in-chest-x-ray-images-under-orange-machine-learning-deep-learning-platform-dd7b6ca6bd4c



curiosity

article in nature communications: **Democratized image analytics by visual programming through integration of deep models and small-scale machine learning.**
https://www.nature.com/articles/s41467-019-12397-x

Use of Orange Data Mining Toolbox for Data Analysis in Clinical Decision Making: The Diagnosis of Diabetes Disease.
https://www.researchgate.net/publication/329707993_Use_of_Orange_Data_Mining_Toolbox_for_Data_Analysis_in_Clinical_Decision_Making_The_Diagnosis_of_Diabetes_Disease

Analysis of Heart Disease using in Data Mining Tools Orange and Weka
https://globaljournals.org/GJCST_Volume18/4-Analysis-of-Heart-Disease.pdf

sources

https://orangedatamining.com/

colab - CIFAR10

Convolutional Neural Network (CNN) - CIFAR10

```
[ ] import tensorflow as tf

    from tensorflow.keras import datasets, layers, models
    import matplotlib.pyplot as plt
```

Download and prepare the CIFAR10 dataset

The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

```
[ ] (train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

    # Normalize pixel values to be between 0 and 1
    train_images, test_images = train_images / 255.0, test_images / 255.0
```

Verify the data

To verify that the dataset looks correct, let's plot the first 25 images from the training set and display the class n

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
```

https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Create the convolutional base

The 6 lines of code below define the convolutional base using a common pattern: a stack of Conv2D and MaxPooling2D la

As input, a CNN takes tensors of shape (image_height, image_width, color_channels), ignoring the batch size. If you are ne dimensions, color_channels refers to (R,G,B). In this example, you will configure your CNN to process inputs of shape (32, format of CIFAR images. You can do this by passing the argument `input_shape` to your first layer.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
model.add(layers.MaxPooling2D((2, 2
model.add(layers.Conv2D(64, (3, 3),
model.add(layers.MaxPooling2D((2, 2
model.add(layers.Conv2D(64, (3, 3),
```

Compile and train the model

```
[ ] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from
                  metrics=['accuracy'])

    history = model.fit(train_images, train_labels, epochs=10,
                        validation_data=(test_images, test_labels))
```

Evaluate the model

```
[ ] plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.5, 1])
    plt.legend(loc='lower right')

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbo
```

```
[ ] print(test_acc)
```

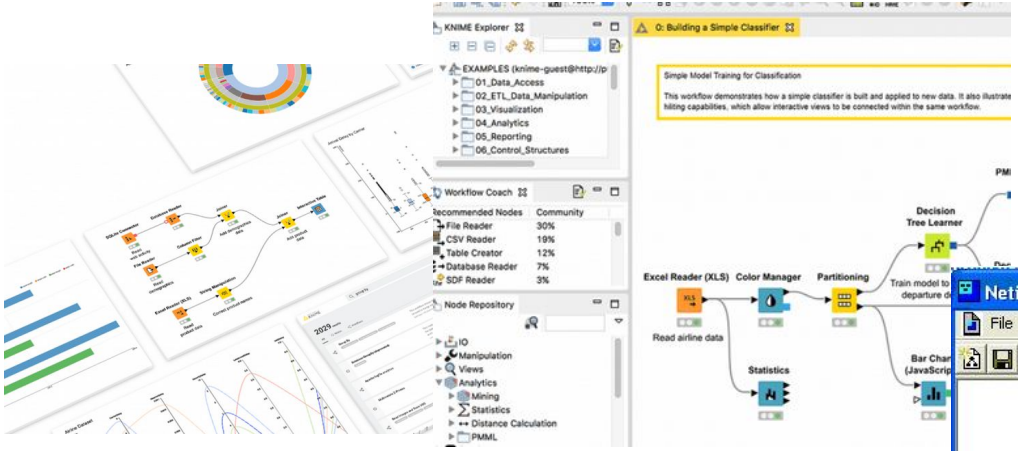
Your simple CNN has achieved a test accuracy of over 70%. Not bad for a few lines of code! For ano quickstart for experts example that uses the Keras subclassing API and `tf.GradientTape`.

https://colab.research.google.com/drive/1Le_aq2TNmLRnkXr-5t10oua8qiF68WfK?usp=sharing

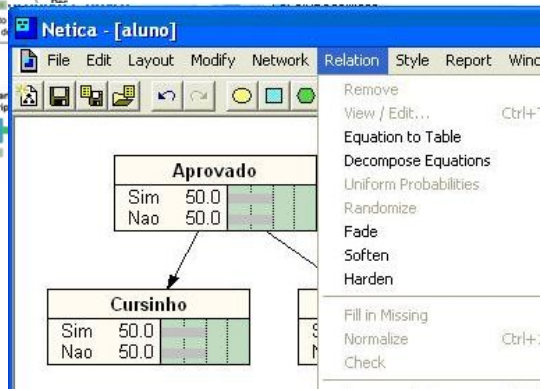
fim
colab

others

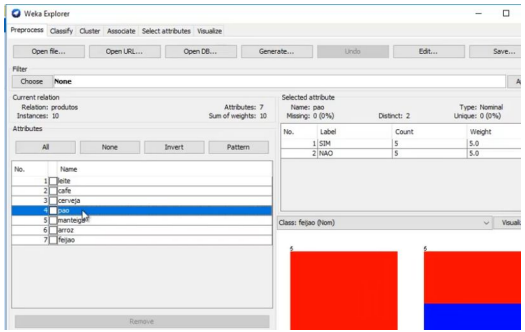
[knime]

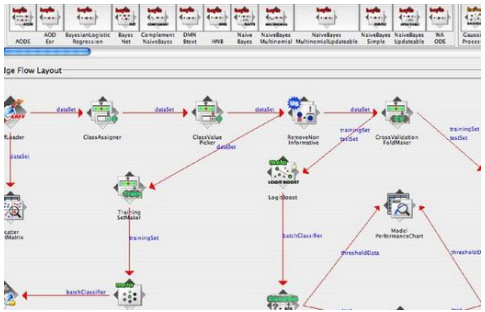


[redes bayesianas com Netica] - [netica]



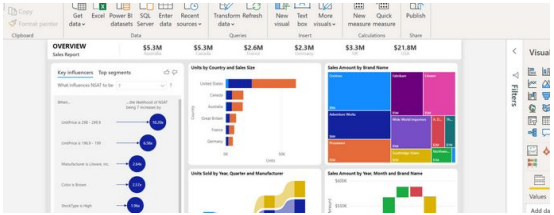
[Regras de associação com weka]
[weka]



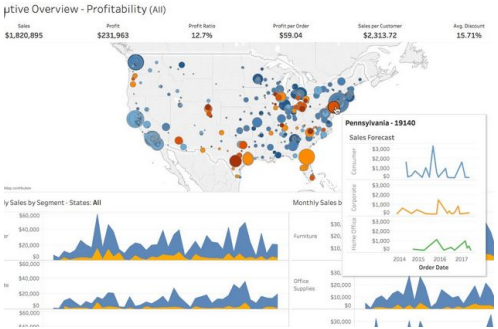


others

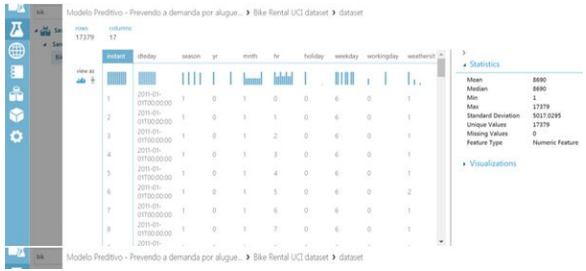
[Power Bi]



[Tableau]



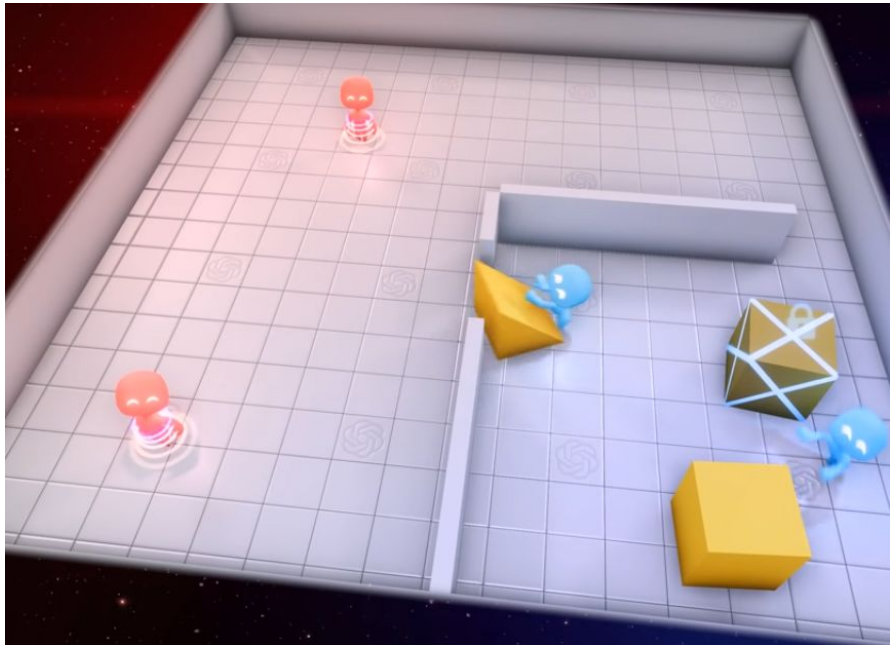
[Modelo Preditivo - Parte 1 - Prevendo a demanda por aluguel de bicicletas usando R e Azure Machine Learning!]



open IA

instituição sem fins lucrativos de pesquisa em inteligência artificial (IA), que tem como objetivo promover e desenvolver IA amigável, de tal forma a beneficiar a humanidade como um todo

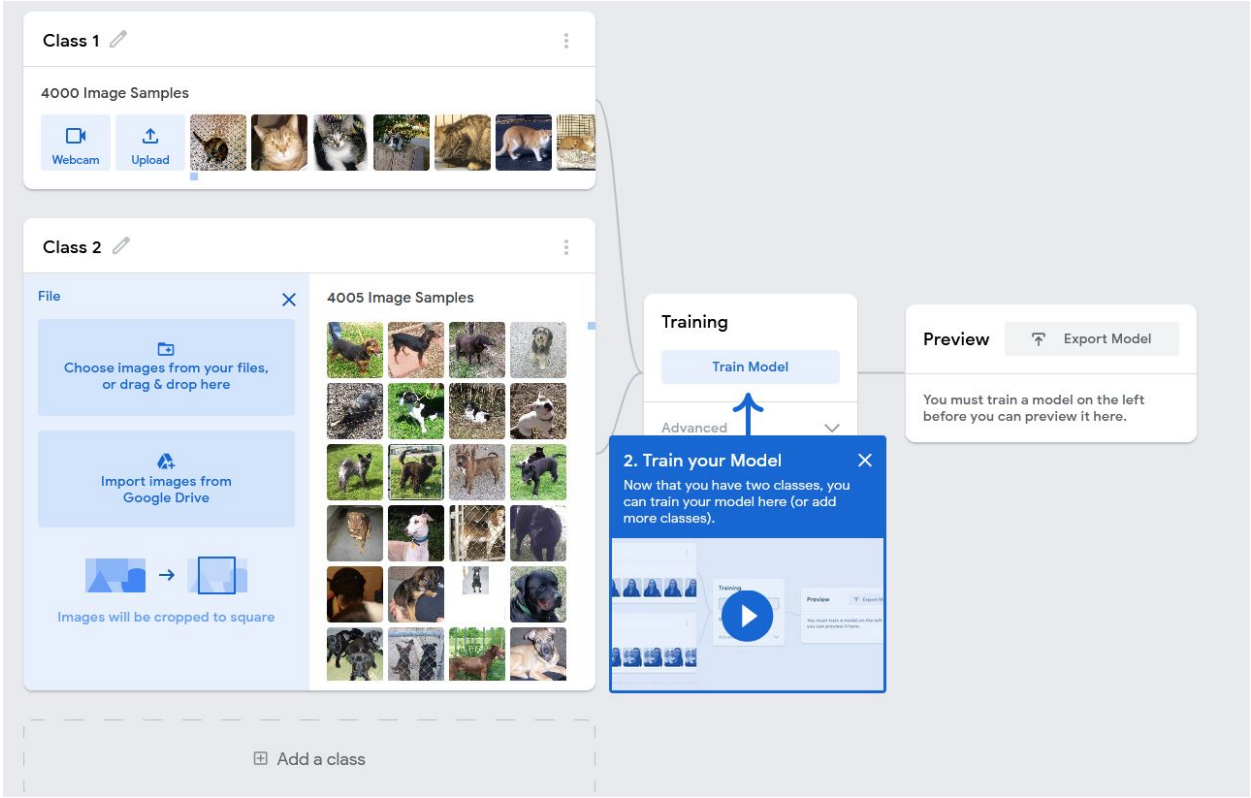
Inteligência Artificial brincando de Pique-Esconde



Teachable Machine

https://teachablemachine.withgoogle.com/

https://www.youtube.com/watch?v=T2qQGqZxkD0


Class 1
4000 Image Samples
Webcam
Upload
Class 2
File
Choose images from your files, or drag & drop here
Import images from Google Drive
Images will be cropped to square
4005 Image Samples
Add a class
Training
Train Model
Advanced
2. Train your Model
Now that you have two classes, you can train your model here (or add more classes).
Preview
Export Model
You must train a model on the left before you can preview it here.

155 min depois...

Training

Train Model

Advanced

Epochs: 50

Batch Size: 16

Learning Rate:

0.001

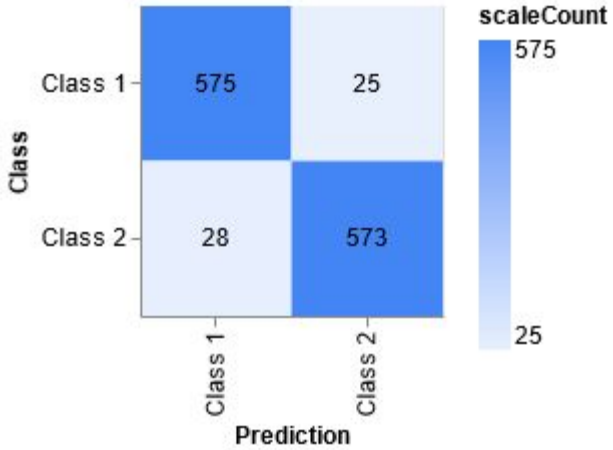
Reset Defaults

Under the hood

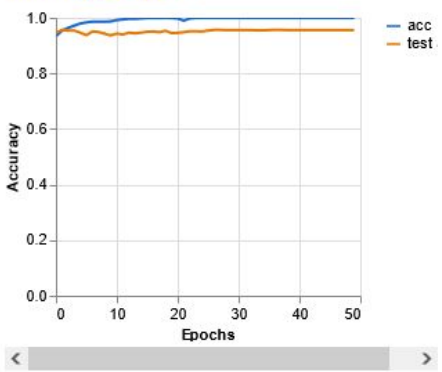
Accuracy per class

CLASS	ACCURACY	# SAMPLES
Class 1	0.96	600
Class 2	0.95	601

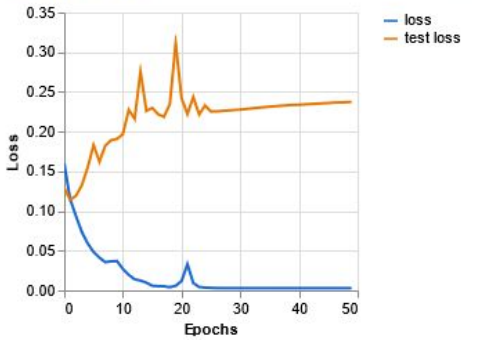
Confusion Matrix



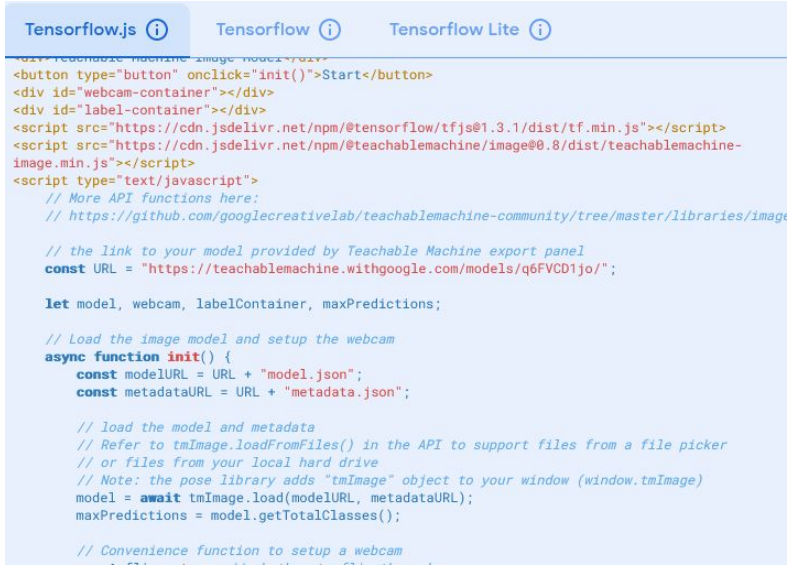
Accuracy per epoch



Loss per epoch

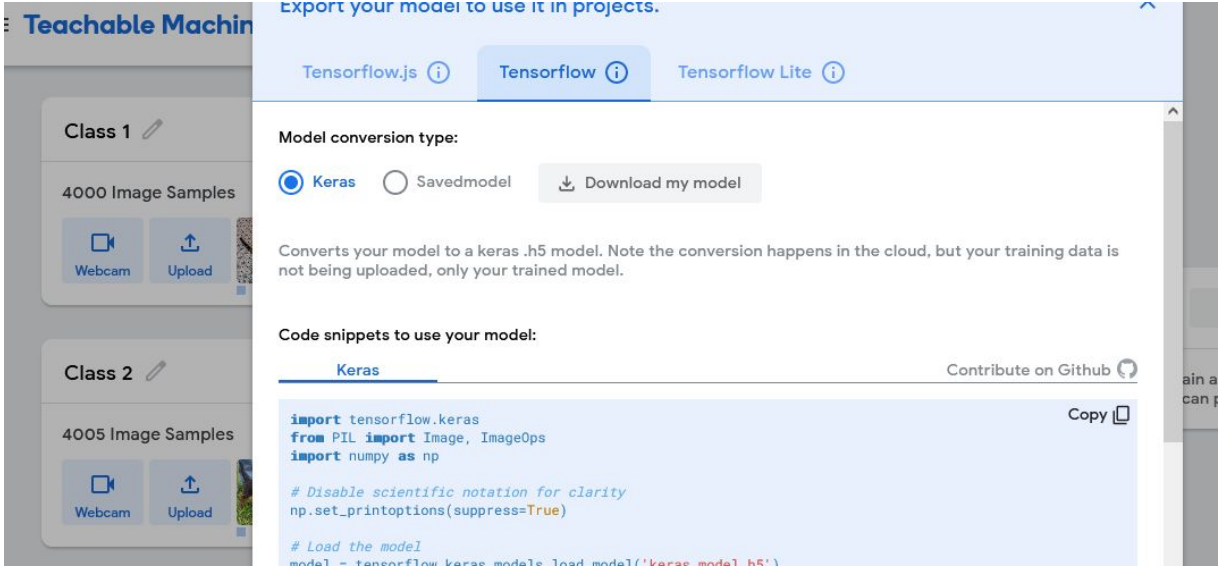


modelo gerado



https://teachablemachine.withgoogle.com/models/q6FVCD1jo/

modelo gerado



[modelo - teachable machine]

[colab - teachable machine]

FIM
Teachable Machine

obrigado

prof.jefer@gmail.com

datasets

[7 datasets gratuitos para inspirar seu portfólio de Ciência de Dados]

[UCI ML - iris dataset]

fique de olho

pytorch

openvino

Caffe

Tensorflow - keras

aprenda+

colab intro redes neurais

colab - intro deep learning - boston housing price - handwrite

colab - intro scikit - iris

Prevendo aluguel de bicicleta

Prevendo valor - aluguel RJ

Prevendo valor passagem - NZ

Prevendo passagens ORANGE

orange no python

https://colab.research.google.com/drive/1zbdbTeDWmBzpEA9VpDhWddW0IFZAUDYp?usp=sharing

```
!pip install Orange3
import Orange

data = Orange.data.Table("housing")
learner = Orange.regression.linear.LinearRegressionLearner()
model = learner(data)

print ("pred obs")
for d in data[:3]:
   print ("%.1f %.1f" % (model(d), d.get_class()))
```

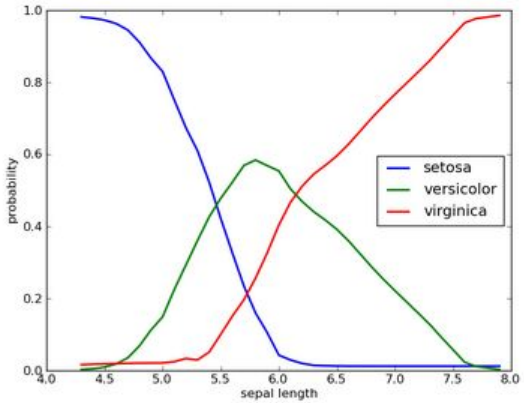
```
pred obs
30.0 24.0
25.0 21.6
30.6 34.7
```

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/distance.html

```
!pip install Orange3
import Orange
from Orange.data import Table
from Orange.distance import Euclidean
iris = Table('iris')
dist_matrix = Euclidean(iris)
# Distance between first two examples
dist_matrix.X[0, 1]
```

0.5385164807134628

```
iris = Orange.data.Table("iris.tab")
nb = Orange.classification.bayes.NaiveLearner(iris)

sepal_length, probabilities = zip(*nb.conditional_distributions[0].items())
p_setosa, p_versicolor, p_virginica = zip(*probabilities)

pylab.xlabel("sepal length")
pylab.ylabel("probability")
pylab.plot(sepal_length, p_setosa, label="setosa", linewidth=2)
pylab.plot(sepal_length, p_versicolor, label="versicolor", linewidth=2)
pylab.plot(sepal_length, p_virginica, label="virginica", linewidth=2)

pylab.legend(loc="best")
pylab.savefig("bayes-iris.png")
```



```
data = Orange.data.Table("housing.tab")
tree = Orange.regression.tree.TreeLearner(data, m_pruning=2., min_instances=20)
print tree.to_string()
```

The script outputs the tree:

```
RM<=6.941: 19.9
RM>6.941
|    RM<=7.437
|    |    CRIM>7.393: 14.4
|    |    CRIM<=7.393
|    |    |    DIS<=1.886: 45.7
|    |    |    DIS>1.886: 32.7
|    RM>7.437
|    |    TAX<=534.500: 45.9
|    |    TAX>534.500: 21.9
```


```
data = Orange.data.Table("housing.tab")
tree = Orange.regression.tree.TreeLearner(data, m_pruning=2., min_instances=20)
print tree.to_string()
```

The script outputs the tree:

```
RM<=6.941: 19.9
RM>6.941
|    RM<=7.437
|    |    CRIM>7.393: 14.4
|    |    CRIM<=7.393
|    |    |    DIS<=1.886: 45.7
|    |    |    DIS>1.886: 32.7
|    RM>7.437
|    |    TAX<=534.500: 45.9
|    |    TAX>534.500: 21.9
```


https://docs.biolab.si/orange/2/tutorial/rst/regression.html

```
>>> from Orange import data, distance
>>> from Orange.clustering import hierarchical
>>> data = data.Table('iris')
>>> dist_matrix = distance.Euclidean(data)
>>> hierar = hierarchical.HierarchicalClustering(n_clusters=3)
>>> hierar.linkage = hierarchical.AVERAGE
>>> hierar.fit(dist_matrix)
>>> hierar.labels
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  0.
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  2.,  0.,  2.
        2.,  2.,  0.,  2.,  2.,  2.,  2.,  2.,  2.,  0.,  0.,  2.
        2.,  2.,  0.,  2.,  0.,  2.,  0.,  2.,  2.,  0.,  0.,  2.
        2.,  2.,  2.,  0.,  2.,  2.,  2.,  2.,  0.,  2.,  2.,  2.
        2.,  2.,  2.,  0.,  2.,  2.,  0.])
```

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/clustering.hierarchical.html#example

Linear Regression

```
>>> from Orange.regression.linear import LinearRegressionLearner
>>> mpg = Orange.data.Table('auto-mpg')
>>> mean_ = LinearRegressionLearner()
>>> model = mean_(mpg[40:110])
>>> print(model)
LinearModel LinearRegression(copy_X=True, fit_intercept=True, normal
>>> mpg[20]
Value('mpg', 25.0)
>>> model(mpg[0])
Value('mpg', 24.6)
```

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/regression.html#linear-regression

cross validation

```
data = Orange.data.Table("titanic")
lr = Orange.classification.LogisticRegressionLearner()
res = Orange.evaluation.CrossValidation(data, [lr], k=5)
print("Accuracy: %.3f" % Orange.evaluation.scoring.CA(res)[0])
print("AUC:      %.3f" % Orange.evaluation.scoring.AUC(res)[0])
```

```
Accuracy: 0.779
AUC:      0.704
```

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/tutorial/classification.html#learners-and-classifiers

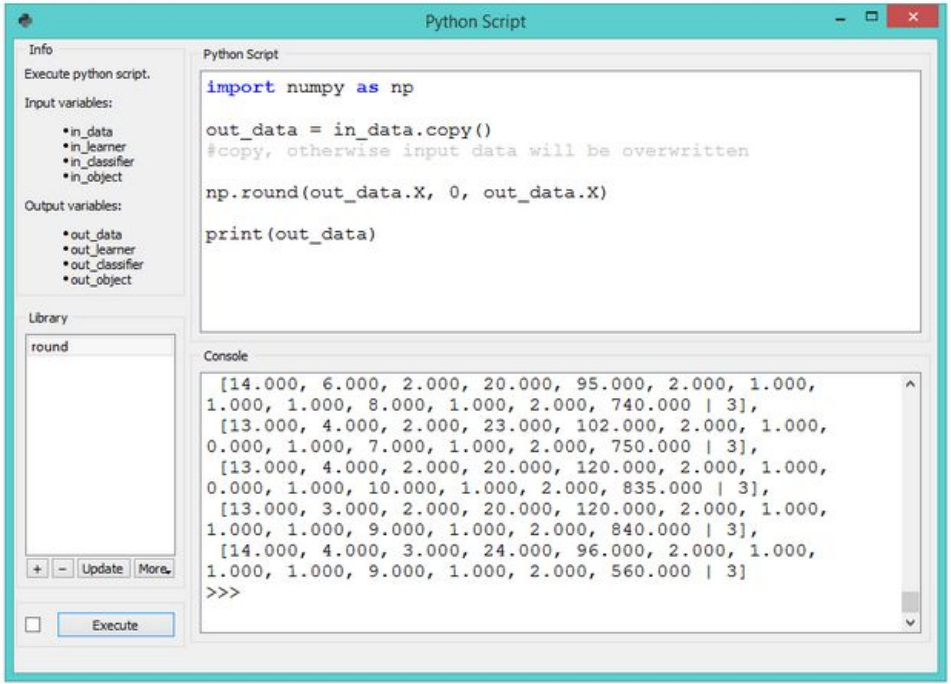
classification tree

```
>>> import Orange
>>> iris = Orange.data.Table('iris')
>>> tr = Orange.classification.TreeLearner()
>>> classifier = tr(data)
>>> printed_tree = classifier.print_tree()
>>> for i in printed_tree.split('\n'):
>>>     print(i)
[50.  0.  0.] petal length ≤ 1.9
[ 0. 50. 50.] petal length > 1.9
[ 0. 49.  5.]     petal width ≤ 1.7
[ 0. 47.  1.]         petal length ≤ 4.9
   [0. 2. 4.]         petal length > 4.9
   [0. 0. 3.]             petal width ≤ 1.5
   [0. 2. 1.]             petal width > 1.5
   [0. 2. 0.]                 sepal length ≤ 6.7
   [0. 0. 1.]                 sepal length > 6.7
[ 0.  1. 45.]     petal width > 1.7
```

https://orange3.readthedocs.io/projects/orange-data-mining-library/en/latest/reference/classification.html#classification-tree

python no orange

```
import numpy as np
out_data = in_data.copy()
#copy, otherwise input data will be overwritten
np.round(out_data.X, 0, out_data.X)
```



IA [údica

atividades

Shy Panda

Ocean IA