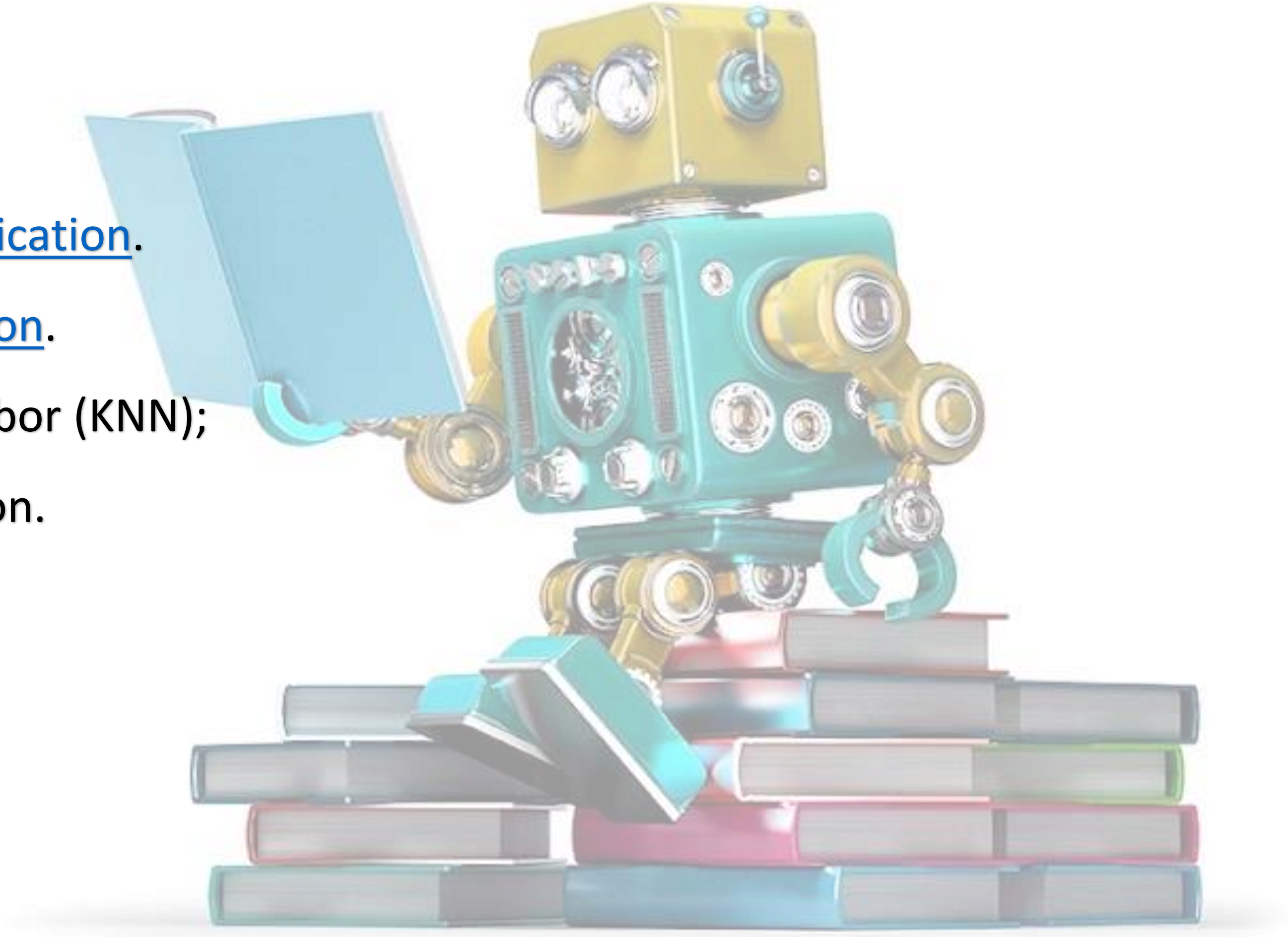
Python: Logistic regression and k-Nearest Neighbor algorithms

Eduardo Destefani Stefanato[1]*
Vitor Souza Premoli Pinto de Oliveira[1]*

[1]Universidade Federal do Espírito Santo

Artificial Intelligence Applied to Images

- Objectives;
- Machine learning;
    - Learning process;
        - Iris data set application.
    - Algorithm Classification.
        - K-Nearest Neighbor (KNN);
        - Logistic regression.
    - Cross-validation;
        - Metrics.
- Conclusion;
    - Cross validate.
- References.



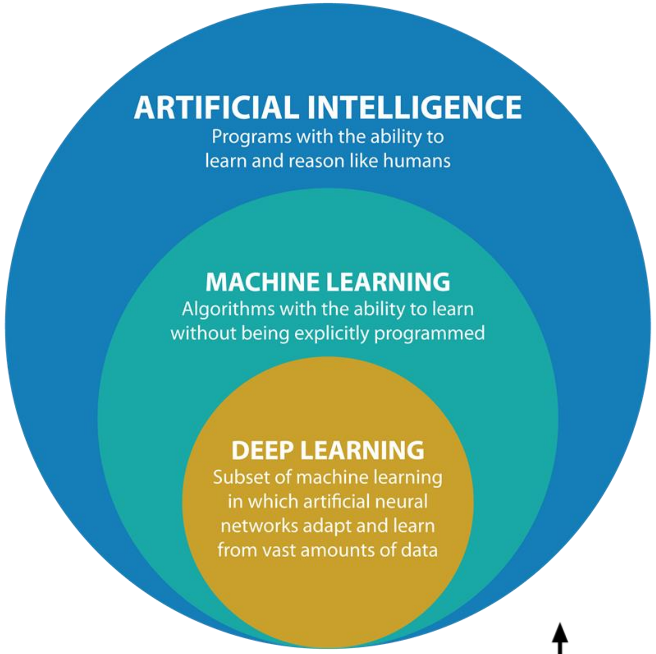
Objectives

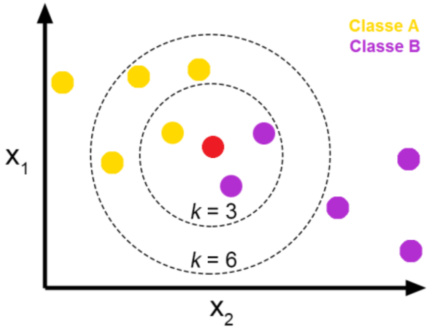
1. Introduce the idea of **Machine Learning**;

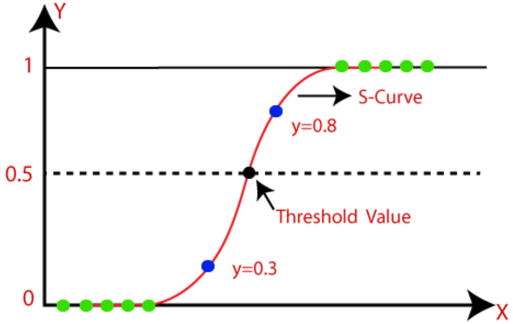
2. Show the mathematical logic behind two algorithms used in classification (**KNN** and **Logistic regression**);

3. Evaluate (through **cross-validation**) the two models with different algorithms in different **metrics**;

4. After made the evaluation, show which model had better performance in each metric.







MACHINE LEARNING

Learning process

Machine learning is a subfield of artificial intelligence, the study and construction of algorithms that can learn from data and make predictions. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to adapt intelligently.

Machine learning methods

Machine learning elements

Are divided into:

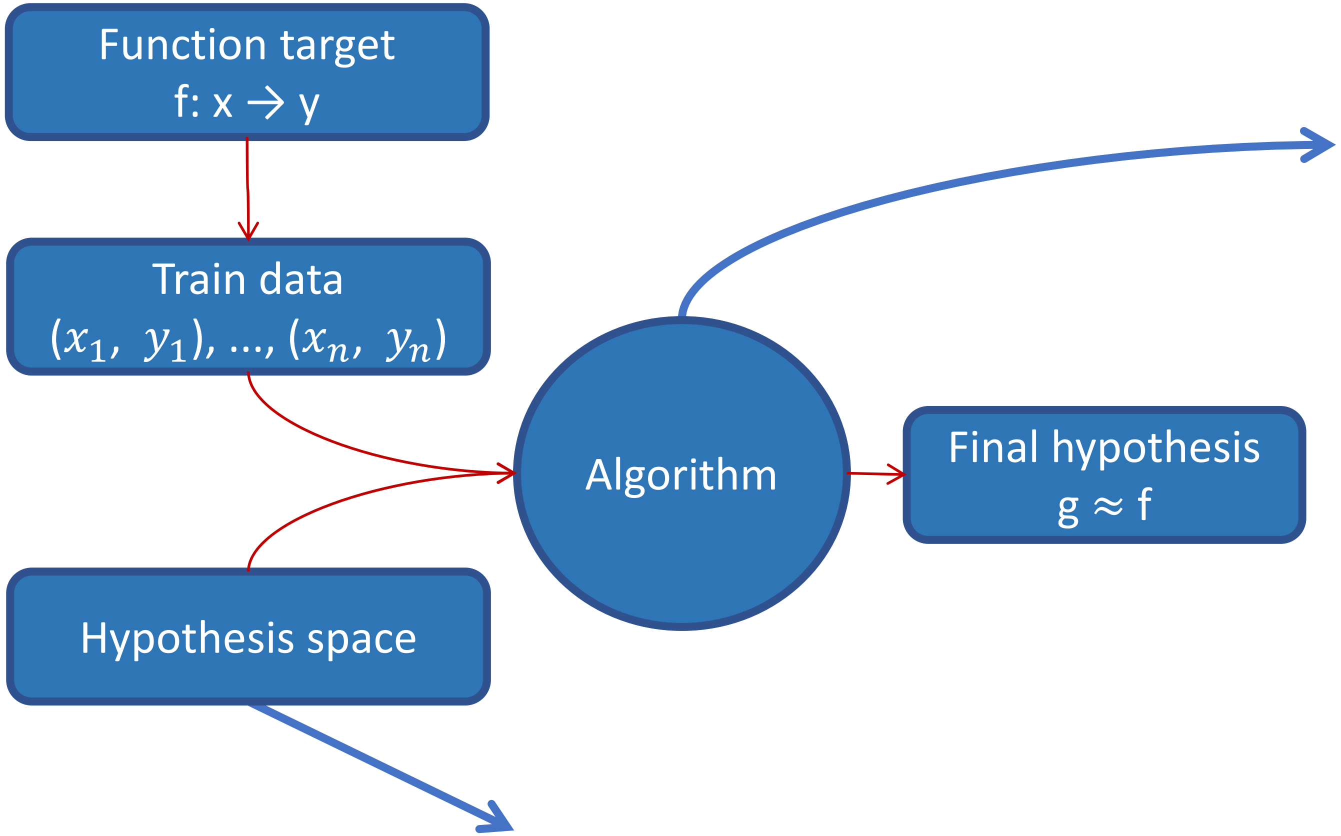
For this chapter,
we have to:

Supervised Learning

Unsupervised Learning

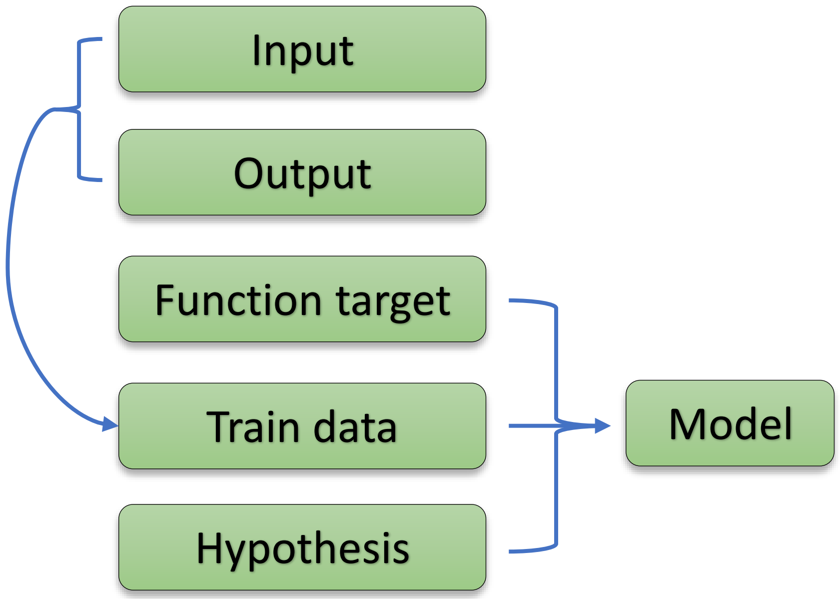
Reinforcement Learning

Learning process of Machine Learning



No model is 100% effective, the algorithm determines based on the **cost function** which model minimizes the errors

N possible functions (g) for the algorithm to determine the most efficient one for the model based on the predictor and target variables



Extracting the dataset

```
# library data sets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# primal data
data = pd.read_csv('data/Iris.csv').copy(); data

data['Species'] = [i.split('-')[1] for i in data['Species']]; data = data.drop(columns=['Id']); data
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

The rows being the samples and the columns being: Sepal length (cm), Sepal width (cm), Petal length (cm) and Petal width (cm).

Predictive Model

Input — x Length and width data of petals and sepals

Output — y Decision→ Iris-setosa, Iris-virginica or Iris-versicolor

Function Target — F = x→ y {Species profiles}
{Unknown mathematical function}

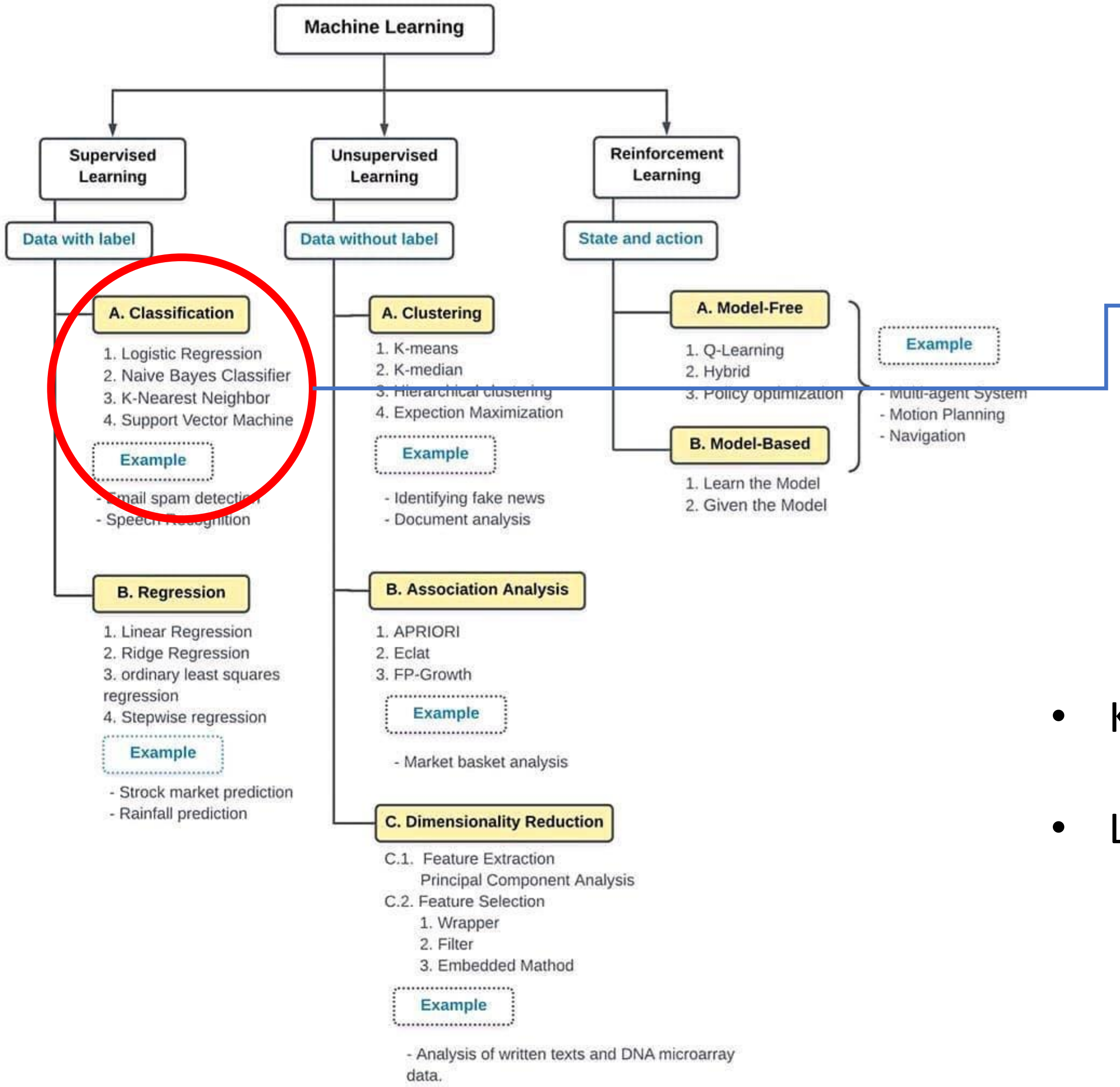
Train Data —

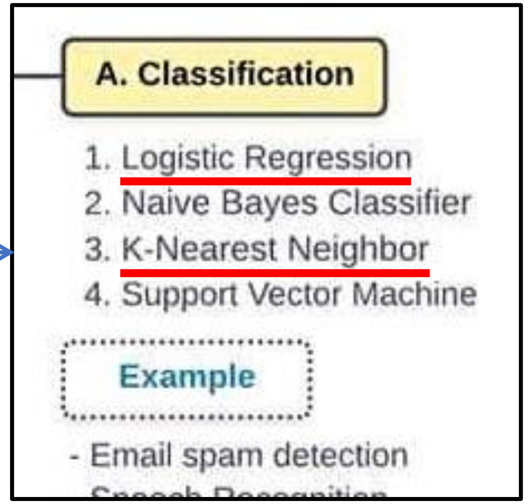
$$\begin{bmatrix} X_{00} & \cdots & X_{03} \\ \vdots & \ddots & \vdots \\ X_{149\,0} & \cdots & X_{149\,3} \end{bmatrix} \begin{bmatrix} Y_1 \\ \vdots \\ Y_{149} \end{bmatrix}$$

Where: X_{ij} = Width and Length data of Petals and Sepals
Where: Y_i = Specie related to X_{ij} values

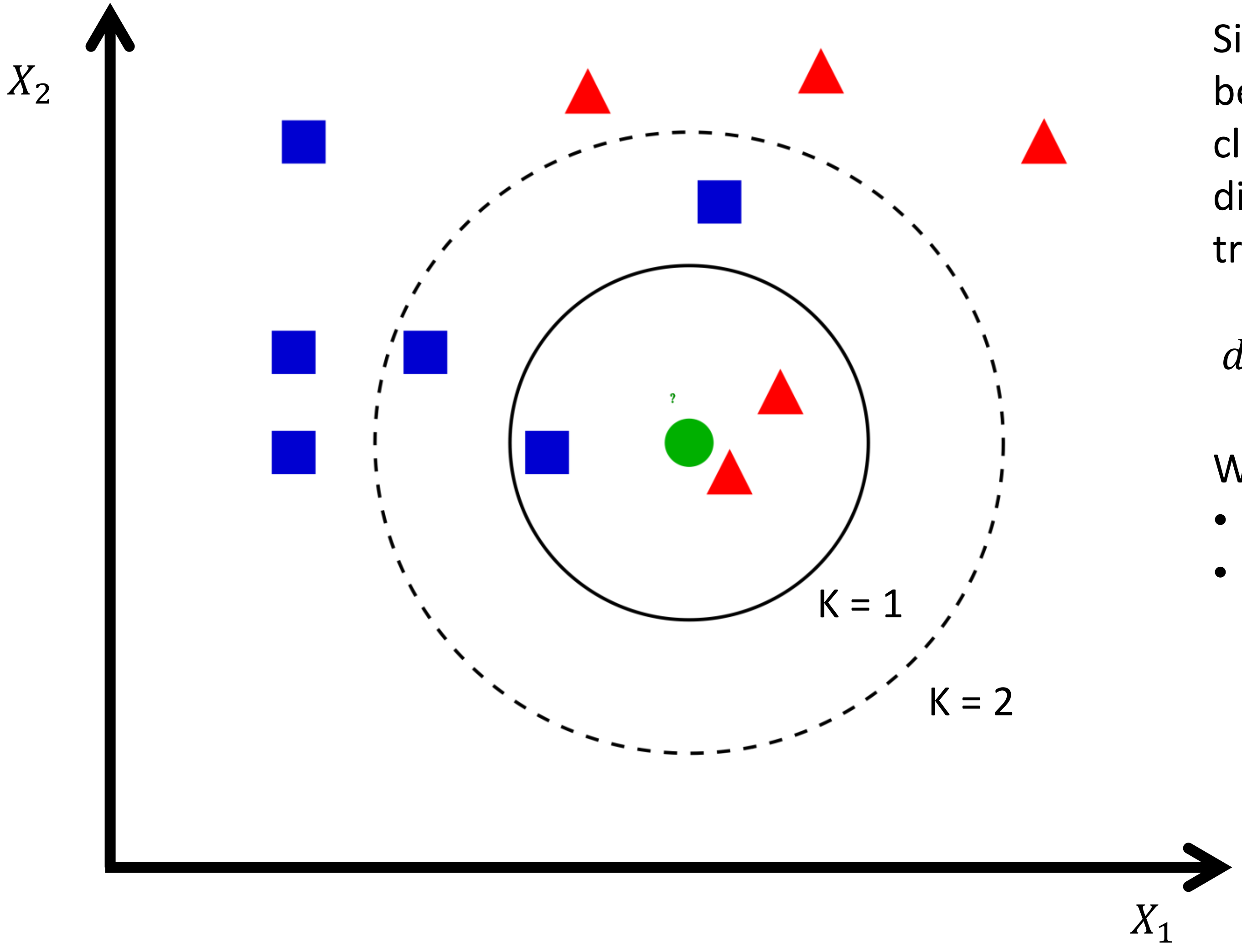
Hypothesis — G = x→ y Function to be discovered by the algorithm

Supervised Learning: Algorithms Classification





- K-Nearest Neighbor → is a instance-based algorithm
- Logistic Regression → is a regression algorithm



Similarity is defined according to a distance metric between two data points. The k-nearest-neighbor classifier is commonly based on the Euclidean distance between a test sample and the specified training samples.

$$d(X_i, X_l) = \sqrt{(X_{i1} - X_{l1})^2 + \cdots + (X_{ip} - X_{lp})^2}$$

Where:

- *i* = numbers data;
- *p* = *n* features (dimensions).

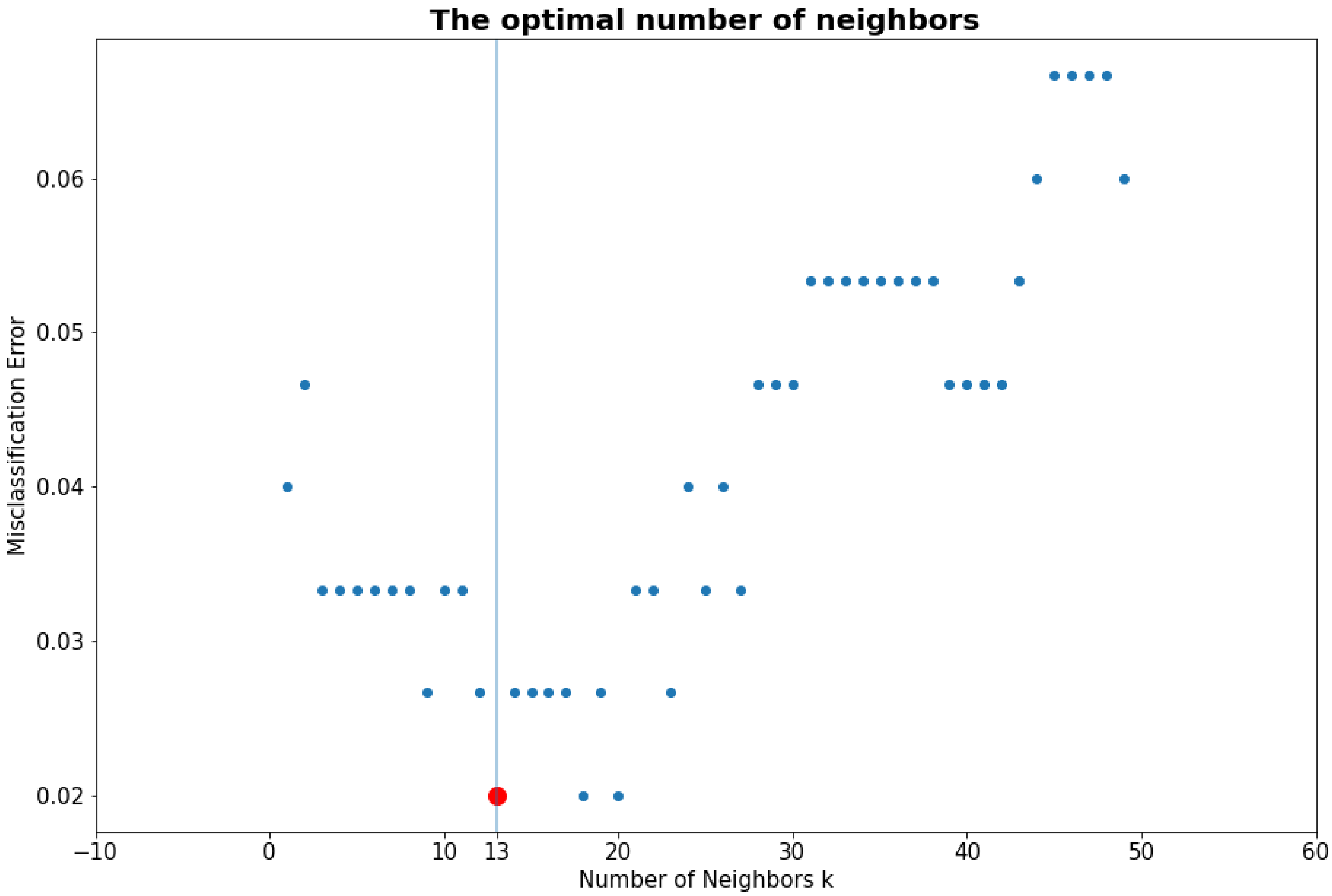
Choosing the **value of k is critical**.

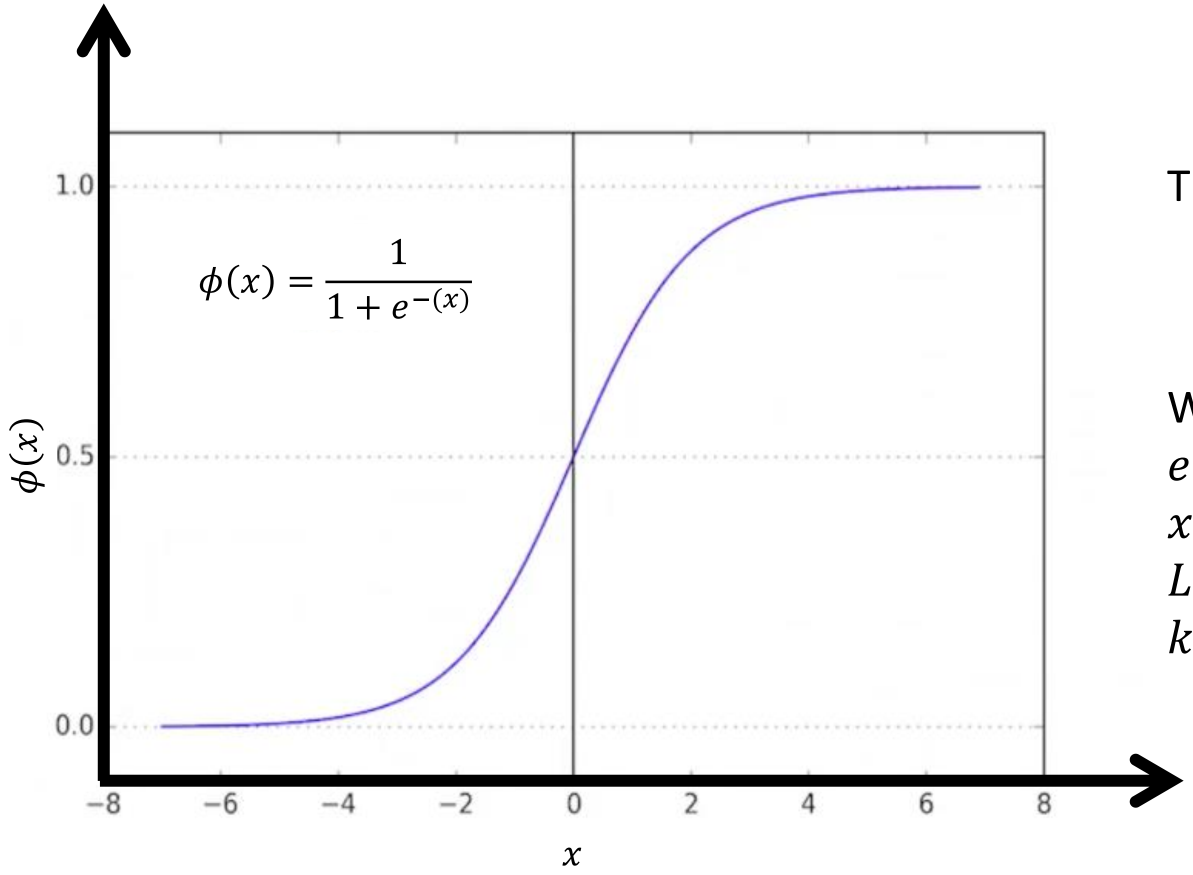
- A too small k results in a solution that does not;
- tolerates noise (I create islands of data);
- A too large k goes against the KNN philosophy (I don't create a good neighborhood);
- Generic rule for choosing k:

$$k = \sqrt{n}$$

```
The optimal number of neighbors with accuracy is 13.
The optimal number of neighbors with precision is 13.
The optimal number of neighbors with recall is 13.
The optimal number of neighbors with f1 is 13.
```

K-NN (K-Nearest Neighbor)





The model uses as link function:

$$\phi(x) = \frac{L}{1 + e^{-k(x - x_o)}} \quad \text{(Logistic Function)}$$

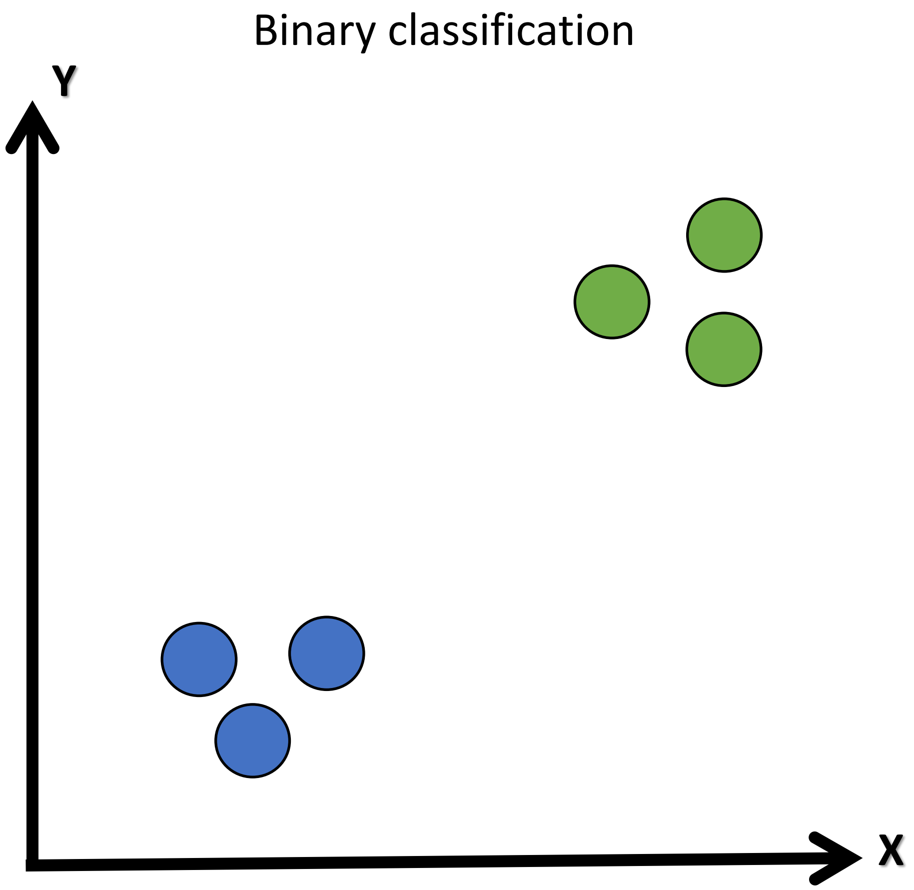
Where:

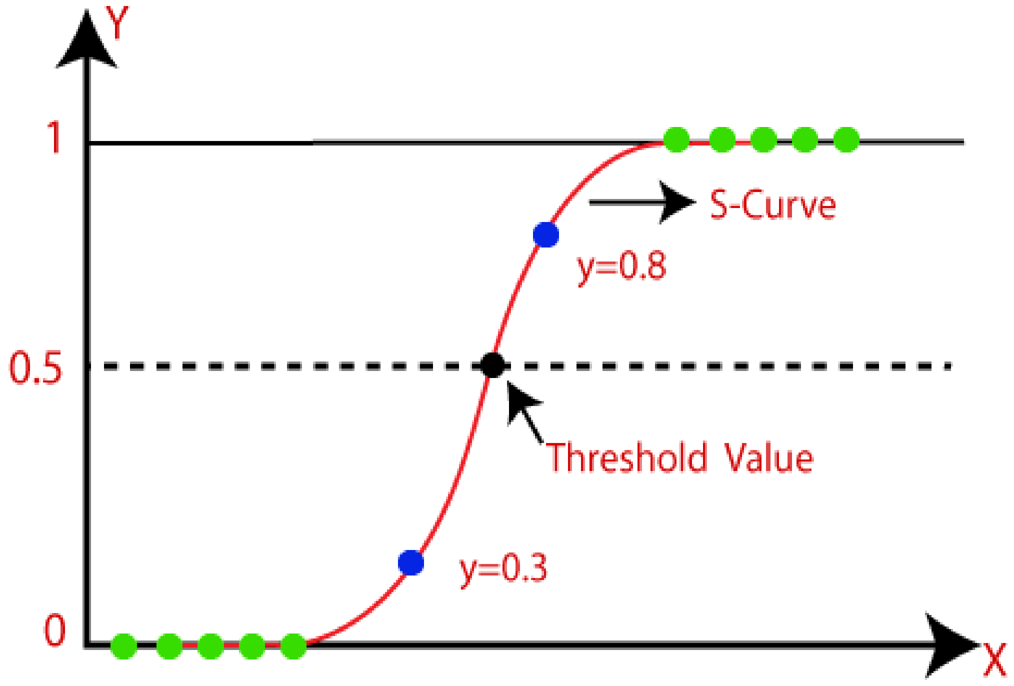
e = Euler's number

x_o = z value at the midpoint of the sigmoid curve,

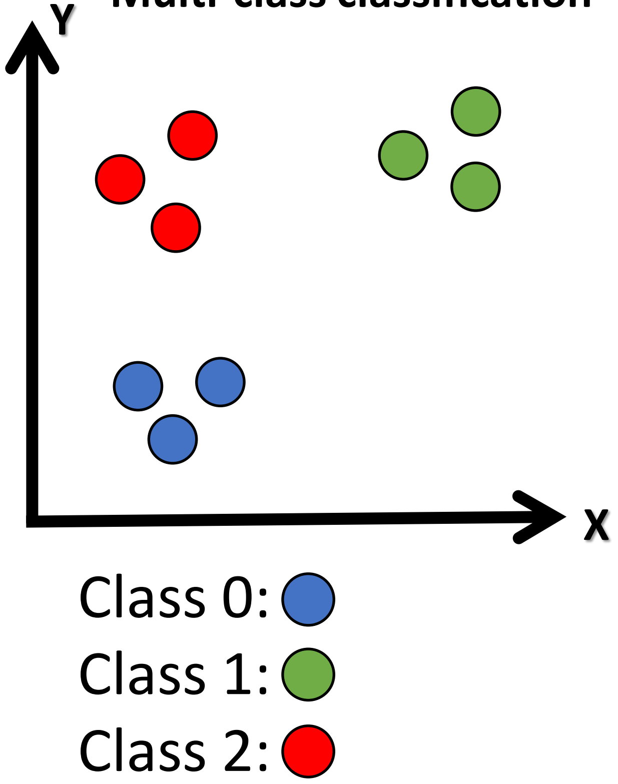
L = maximum curve value,

k = curve's declivity


Binary classification
Y
X


Y
1
0.5
0
X
S-Curve
y=0.8
Threshold Value
y=0.3

Multi-class classification

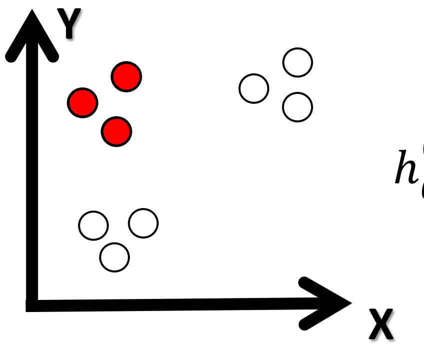


On a new input x, to make a prediction, pick the class i that maximizes.

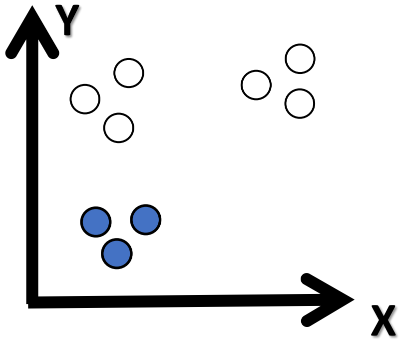
$$\max_i \; h_\theta^{(i)}(x)$$

Train a logistic regression classifier $h_\theta^{(i)}$ for each class i to predict the probability that y=i

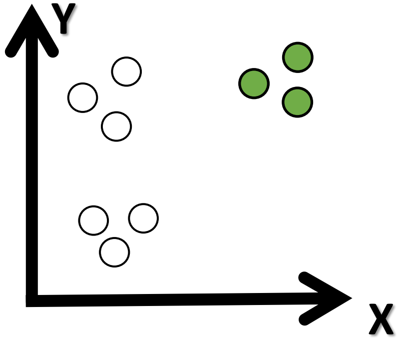
$$h_\theta^{(i)} = P(y - i|\text{x};\theta) \qquad (i = 0, 1, 2, \ldots)$$



$$h_\theta^{(1)} = P(y - 0|\text{x};\theta)$$



$$h_\theta^{(2)} = P(y - 1|\text{x};\theta)$$

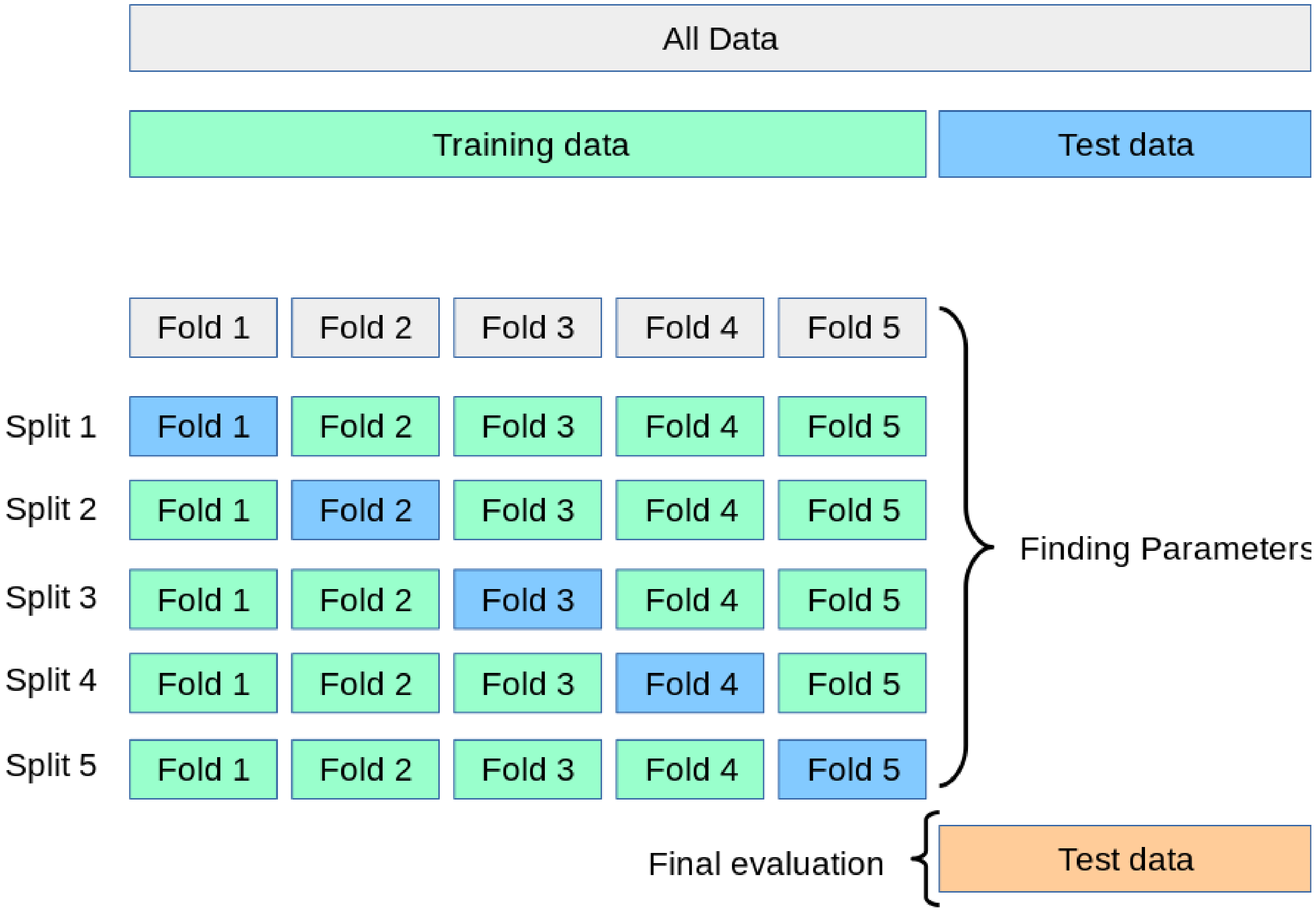


$$h_\theta^{(3)} = P(y - 2|\text{x};\theta)$$

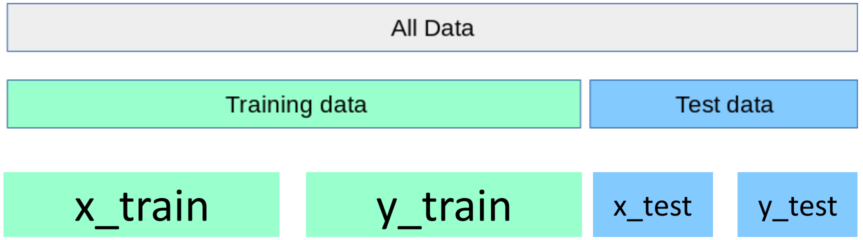
CROSS-VALIDATION

Metrics

What's Cross-validation?



Ability to evaluate the generalizability of a machine learning model from a dataset. Estimate how accurate the model is in practice.



The hold-out method is good to use when you have a very large dataset, you're on a time crunch, or you are starting to build an initial model in your data science project. But does not deliver the real potential of the model.

Metrics of cross validate

1. KNN

```
for v,i in enumerate(scores): #Cross_validate
    if v>1:
        print('{}: ({} +/- {}) %'.format(i, round(np.mean(scores[i])*100), round(np.std(scores[i])*100)))
```

```
test_accuracy: (94 +/- 5) %
test_precision_macro: (95 +/- 4) %
test_recall_macro: (94 +/- 5) %
test_f1_macro: (94 +/- 5) %
```

2. Logistic Regression

```
for v,i in enumerate(scores_LR): #Cross_validate
    if v>1:
        print('{}: ({} +/- {}) %'.format(i, round(np.mean(scores_LR[i])*100), round(np.std(scores_LR[i])*100)))
```

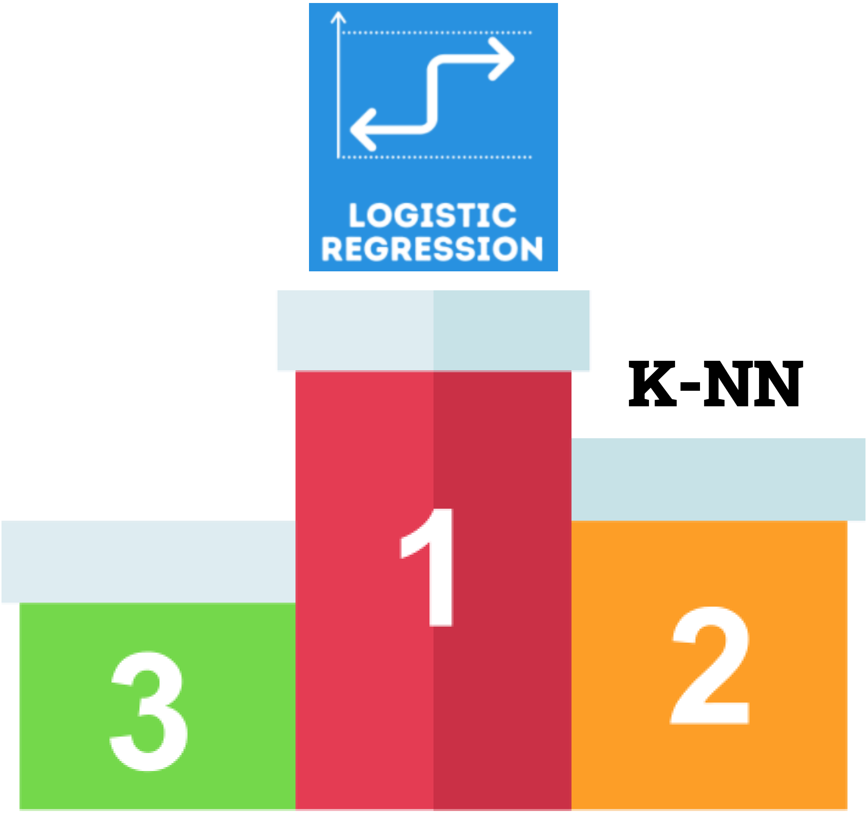
```
test_accuracy: (97 +/- 3) %
test_precision_macro: (98 +/- 3) %
test_recall_macro: (97 +/- 3) %
test_f1_macro: (97 +/- 3) %
```

CONCLUSION

Cross-validation

Best performance

We can see that by applying "Cross Validation" for different metrics. The Logistic Regression algorithm performed better on all scores.



Python: Logistic regression and k-Nearest Neighbor algorithms

Eduardo Destefani Stefanato[1]*
Vitor Souza Premoli Pinto de Oliveira[1]*

[1]Universidade Federal do Espírito Santo

Artificial Intelligence Applied to Images

REFERENCES

Matplotlib. **Matplotlib: Python plotting — Matplotlib 3.4.2 documentation.** Available in: https://matplotlib.org/. Access in: 22 Jul. 2021.

Numpy. **NumPy Reference — NumPy v1.21 Manual.** Available in: https://numpy.org/doc/stable/reference/. Access in: 22 Jul. 2021.

NumFocus. **Pandas.** Available in: https://pandas.pydata.org/. Access in: 29 Jul. 2021.

Kaggle. **Iris Species.** Available in: https://www.kaggle.com/uciml/íris/. Access in: 08 Aug. 2021.

Sktlearn. **Nearest Neighbors Classification.** Available in: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification. Access in: 08 Aug. 2021.

Sktlearn. **LogisticRegression.** Available in: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn-linear-model-logisticregression. Access in: 08 Aug. 2021.