

SCRIPTING AND PROGRAMMING LABORATORY FOR DATA ANALYSIS



Machine Learning in python: scikit-learn

Fabio Rigamonti, 4th Floor on the left corner, frigamonti@uninsubria.it

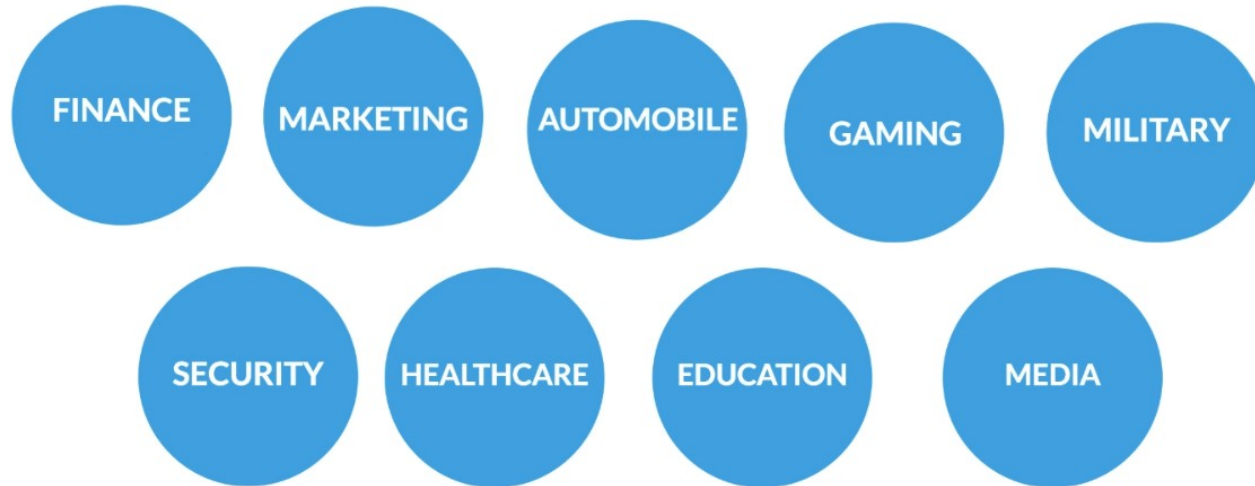


Lecture outline

- Brief introduction to machine learning (supervised, classification).
- Scikit-Learn for:
 - Data handling and reduction (standardization, encoding, train/test splitting).
 - Classification problems (KNN, decision tree, perceptron).
 - Test performance.

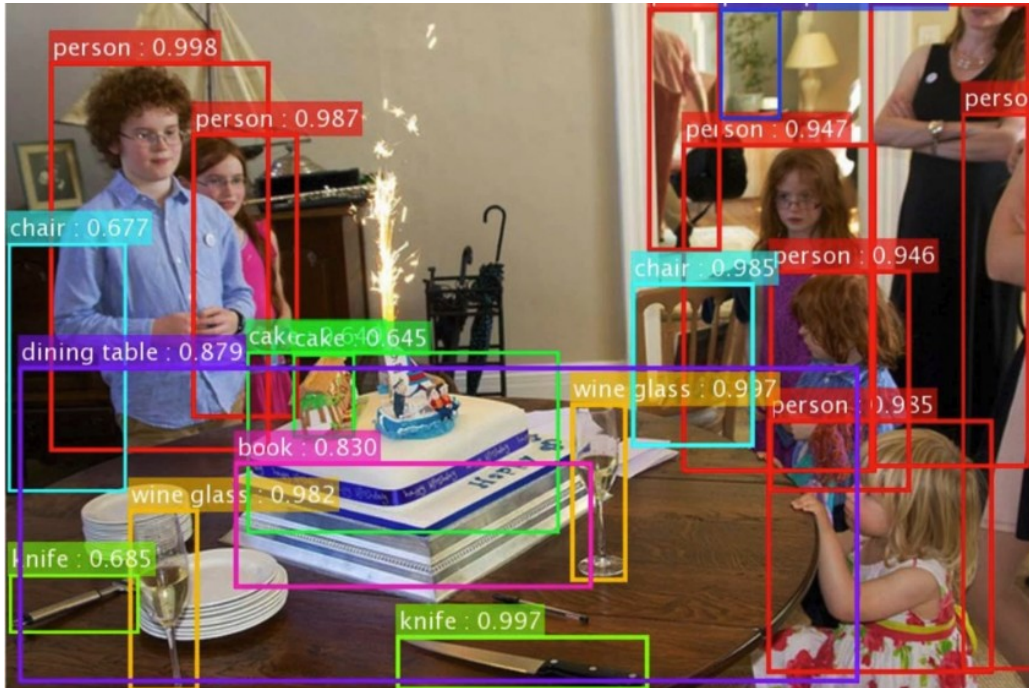
What is Machine Learning?

AI (artificial intelligence) is a general term including several tasks applied in different fields of the market.



Machine Learning is a subfield of AI (Artificial intelligence)

What is Machine Learning?



Have a look at:

TENSORFLOW and **PYTORCH** .

What is Machine Learning?

Machine Learning is a sub-field of Artificial Intelligence.

The main task is fitting a model to data, but the model computed starting from the data themselves.



What is Machine Learning?

Machine Learning is a sub-field of Artificial Intelligence.



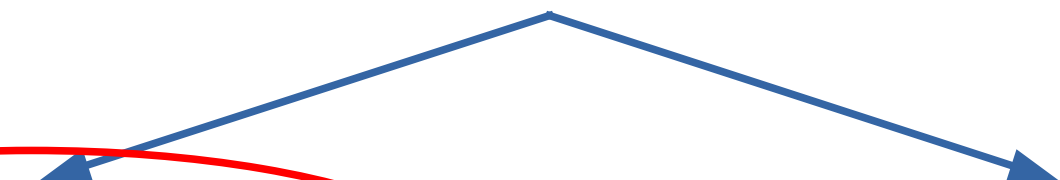
```
graph TD; A[Machine Learning is a sub-field of Artificial Intelligence.] --> B[Supervised learning: you give your model both the labels (x) and the targets (y).]; A --> C[Unsupervised learning: you give your model only the labels (x) and it tries to find patterns and unknown relations.];
```

Supervised learning: you give your model both the labels (\mathbf{x}) and the targets (y).

Unsupervised learning: you give your model only the labels (\mathbf{x}) and it tries to find patterns and unknown relations.

What is Machine Learning?

Machine Learning is a sub-field of Artificial Intelligence.



```
graph TD; A[Machine Learning is a sub-field of Artificial Intelligence.] --> B[Supervised learning: you give to the algorithm both the labels (x) and the targets (y).]; A --> C[Unsupervised learning: you give your model only the labels (x) and it tries to find patterns and unknown relations.];
```

Supervised learning: you give to the algorithm both the labels (x) and the targets (y).

Unsupervised learning: you give your model only the labels (x) and it tries to find patterns and unknown relations.


What is Machine Learning? Useful terminology

Dataset: a collection of data (i.e. the x and y of a standard fit).

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

What is Machine Learning? Useful terminology

Labels (i.e. columns,
the x of the fit)

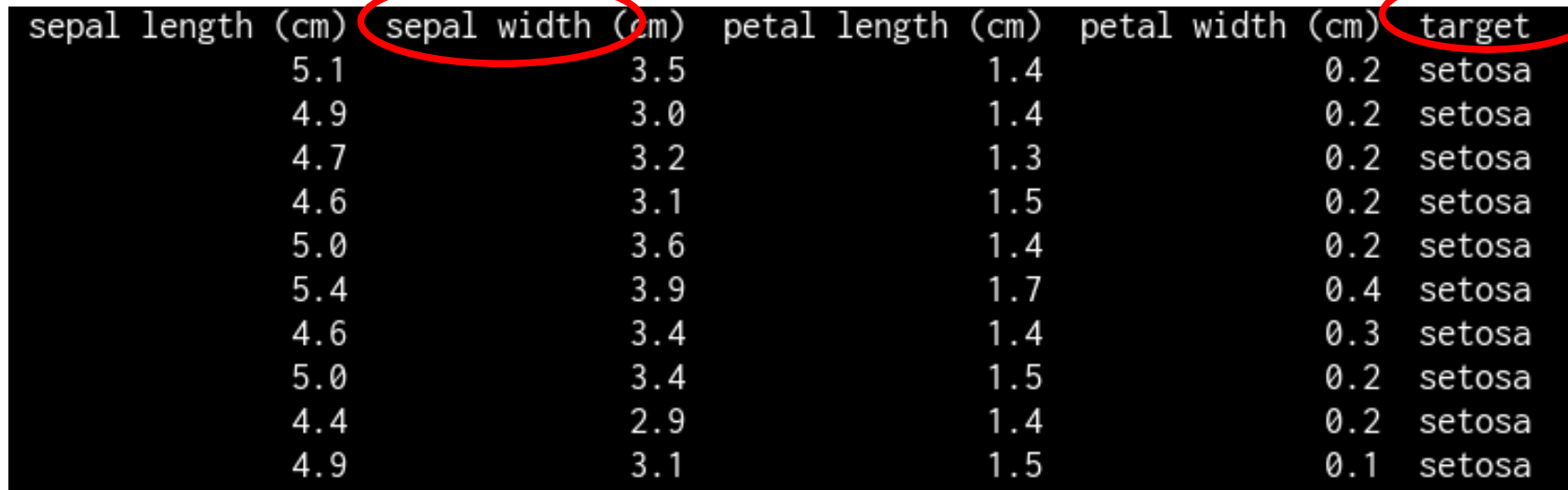


sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

What is Machine Learning? Useful terminology

Labels (i.e. columns,
the x of the fit)

Target (what you want to
predict/**learn**, the y of the fit)

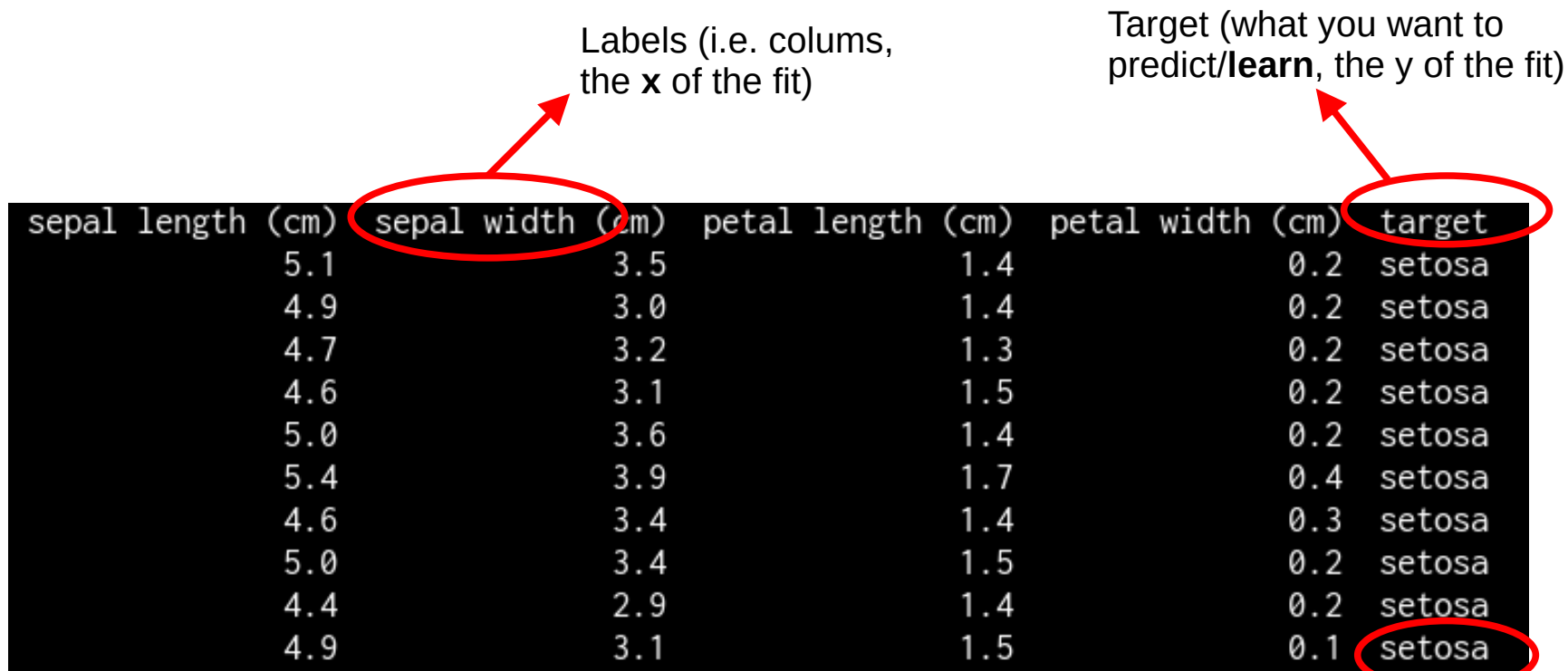


sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

What is Machine Learning? Useful terminology

Labels (i.e. columns,
the **x** of the fit)

Target (what you want to
predict/**learn**, the **y** of the fit)



sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

All data must be real numbers:
ENCODING (setosa → 0.0,
versicolor → 1.0, virginica → 2.0)

What is Machine Learning? Useful terminology

Labels (i.e. columns, the x of the fit)

Target (what you want to predict/**learn**, the y of the fit)

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

A line in your dataset is called pattern

All data must be real numbers:
ENCODING (setosa → 0.0, versicolor → 1.0, virginica → 2.0)

What is Machine Learning? Training

Given your model $F(\mathbf{x}, \mathbf{w})$, you define the loss function $L(\mathbf{x}, \mathbf{w}, y) \sim |F(\mathbf{x}, \mathbf{w}) - y|$.

You search (**gradient descend**) the weights/parameters \mathbf{w} that minimize your loss function.

ML models train/learn by seeing different examples.

The model usually is a non-linear combination of you data points \mathbf{x} and you weights \mathbf{w} .

$$F(\mathbf{x}, \mathbf{w}) \sim \sum_j \phi(w_j \times x_j) \quad \phi(z) \text{ is a non-linear } \textit{activation function}$$



What is Machine Learning? Classification examples

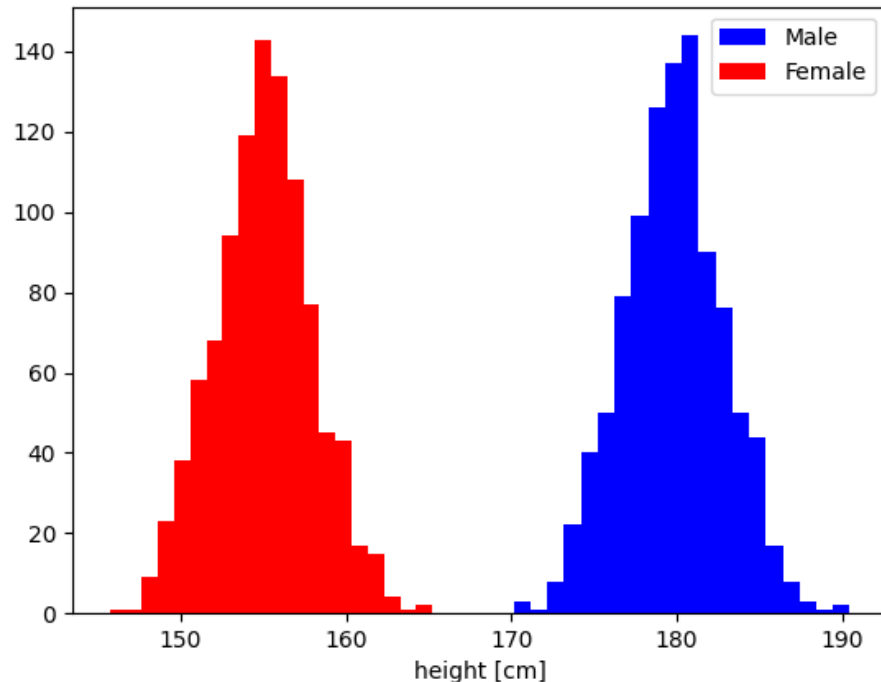
Classification is the task of assigning a label (class) to a particular pattern present in your dataset. Your targets/classes are a discrete number.

Regression: your targets is a continuous variable. This is more similar to a standard fit.

For ML regression and classification are treated similarly. You can think of regression as a classification problem with a huge number of classes.

What is Machine Learning? Classification examples

In physics the model is usually known from theory. Here the model is computed solely from the data, and it has no *theoretical* meaning.



I want to distinguish between male and female based on their height.

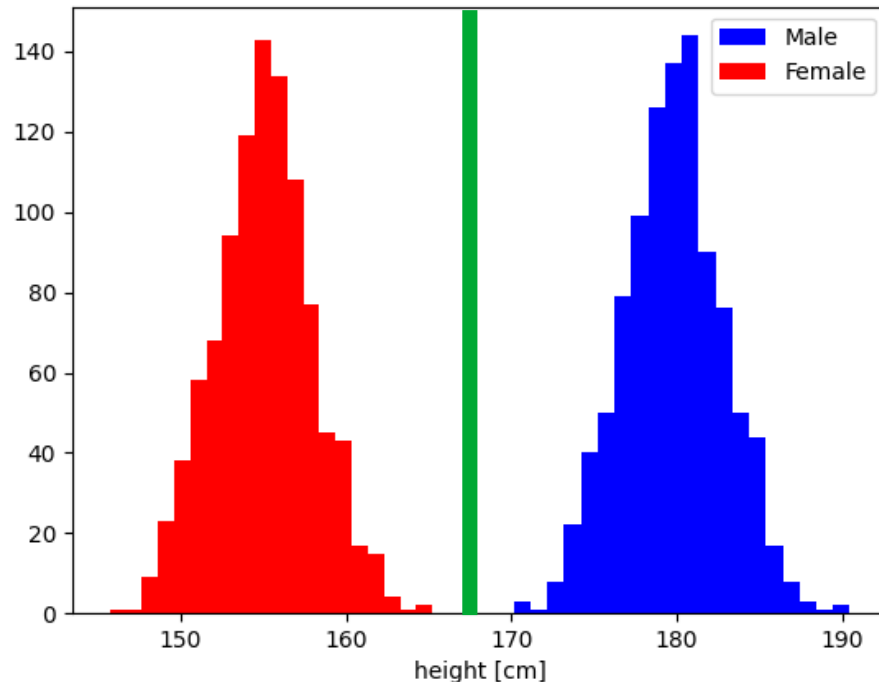
We do not have any theoretical model on that.

Do you have any idea looking at the data?

What is Machine Learning?

Training:

In physics the model is usually known from theory. Here the model is computed solely from the data, and it has no theoretical meaning.



Male: $h > 168$ cm

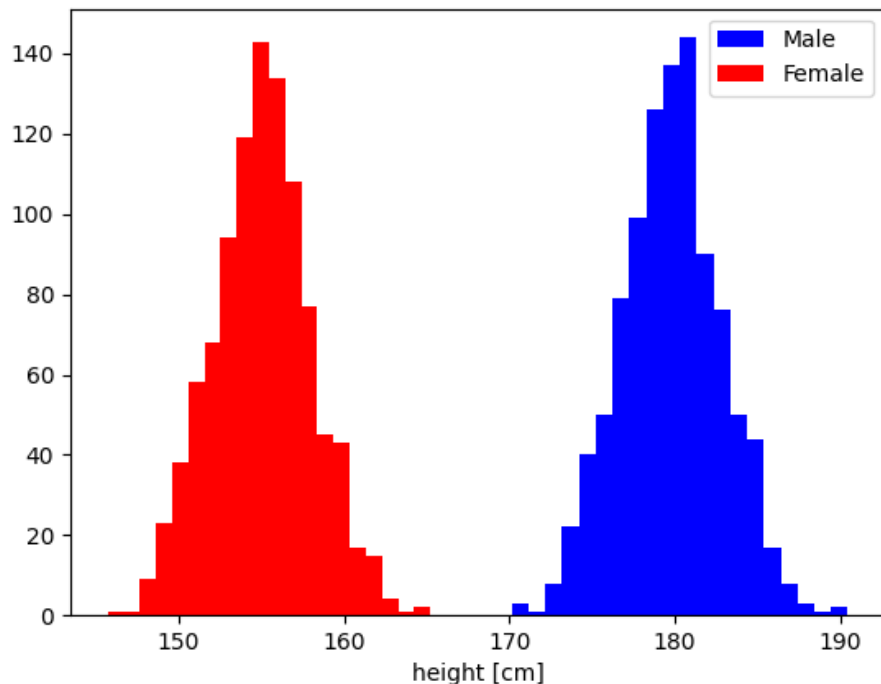
Female: $h < 168$ cm

Note that this is a linear model

What is Machine Learning?

Training:

In physics the model is usually known from theory. Here the model is computed solely from the data, and it has no theoretical meaning.



$$\langle h_{male} \rangle = \frac{\sum_i^N h_{i;male}}{N}$$

$$\langle h_{female} \rangle = \frac{\sum_i^N h_{i;female}}{N}$$

$$|h' - \langle h_{male} \rangle| < |h' - \langle h_{female} \rangle|$$



What is Machine Learning? Workflow

Preprocessing of the data:

Make sure that your dataset is not biased. Rescale using standardization or min/max scaler. Encode non-numeric variables.



What is Machine Learning? Workflow

Preprocessing of the data.

Train test splitting:

Divide your data into training set (75%) and test set (25%). The selection must not be biased (i.e. random selection), remember that your ML model **learn** by seeing many different examples.



What is Machine Learning? Workflow

Preprocessing of the data.

Train test splitting.

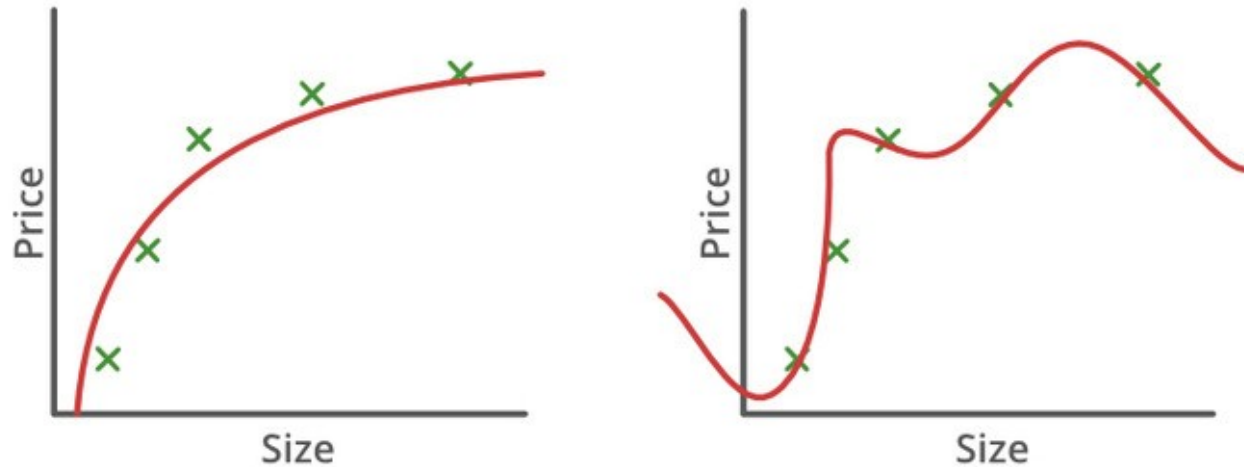
Training:

Your model sees all the examples in the training set and starts to recognise patterns and relations. Training is an iterative task in which your model parameters are updated until a “good enough” fit to the data is reached.

AVOID OVERFITTING

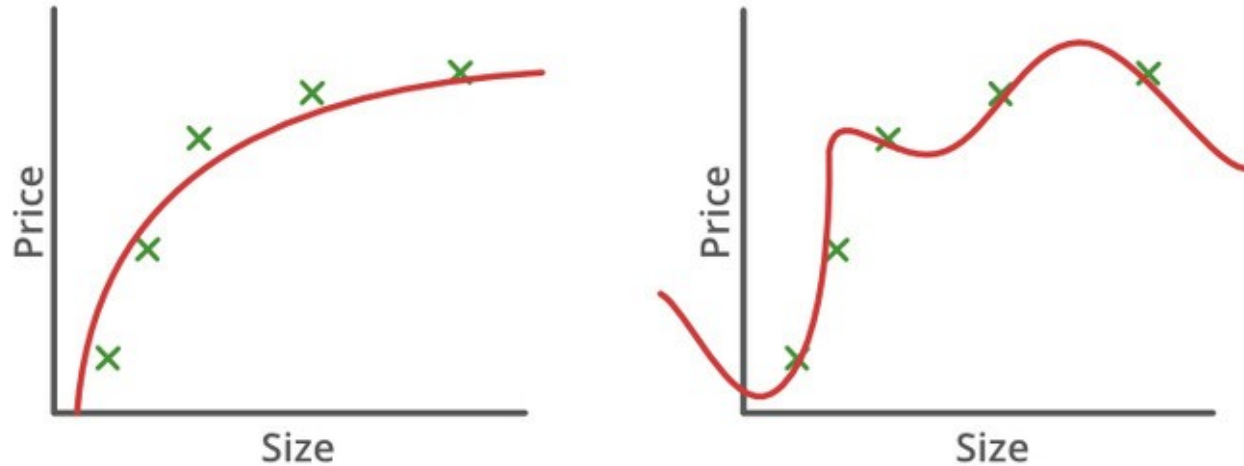
Excursus: Overfitting

Overfitting is when your model has too much flexible (usually it has too many free parameters) and it learns too well your training data.



Excursus: Overfitting

Overfitting is when your model has too much flexible (usually it has too many free parameters) and it learns too well your training data.



Apply your trained model on data that it has never seen. Is it able to **generalise**?



What is Machine Learning? Workflow

Preprocessing of the data.

Train test splitting.

Training.

Test :

Once you trained your model, you should check its performance (different metrics availables) on never seen data.

good performance on train + bad performance on test
=
overfitting

good performance on train + good performance on test
=
go to production



What is Machine Learning? Workflow

Preprocessing of the data

Train test splitting

Training

Test

All automatic in python: Scikit learn (sklearn)



Scikit Learn: Overview

Python library based on numpy and scipy. It works well in combination with the most popular python libraries for visualisation (matplotlib and seaborn) and data manipulation (pandas/numpy/scipy).

- **supervised learning (classification and regression)**
- unsupervised learning
- **scoring evaluation**
- **dataset transformation and loading**

and much more....

Scikit Learn: Overview

It is the most complete “catalogue” of ML methods. For example:

1. Supervised learning

- ▶ 1.1. Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- ▶ 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees
- ▶ 1.11. Ensemble methods
- ▶ 1.12. Multiclass and multioutput algorithms
- ▶ 1.13. Feature selection
- ▶ 1.14. Semi-supervised learning

sklearn

Scikit Learn: Data Loading

The module `sklearn.datasets` useful for working with small *toy* datasets commonly used in the AI community.

```
sklearn.datasets.load_iris(*, return_X_y=False, as_frame=False)
```

Return dictionary-like object with the different attributes:

- data: ndarray (the **x** of the dataset), if `as_frame = True` it will be a pandas DataFrame.
- target: ndarray (the classification target), if `as_frame=True`, it will be a pandas Series.
- feature_names: list, the names of the target columns
- target_names: list, the names of the target classes
- frame: DataFrame, present only if `as_frame=True`

[Docs](#)



Scikit Learn: Dataset Transformations

A module of *transformers* for:

- **cleaning (i.e. standardization, encoding, etc.)**
- reducing dimensionality (PCA)
- missing data generation

Scikit Learn: Dataset Transformations

A module of *transformers* for:

- **cleaning** (i.e. standardization, encoding, etc.) *sklearn.preprocessing*
- reducing dimensionality (**PCA**) *sklearn.decomposition.PCA*
- missing data generation



It is useful while dealing with dataset with huge number of features (i.e. columns) for dimensionality reduction and hence speed-up.

Scikit Learn: Dataset Transformations

Sklearn.preprocessing.StandardScaler()

$$x' = \frac{x - \mu}{\sigma}$$

```
from sklearn import preprocessing
import numpy as np
X_train = np.array([[ 1., -1.,  2.],
                    [ 2.,  0.,  0.],
                    [ 0.,  1., -1.]])
scaler = preprocessing.StandardScaler(with_mean=True,with_std=True)
|
scaler.fit(X_train)

X_scaled = scaler.transform(X_train)

X_scaled

array([[ 0.          , -1.22474487,  1.33630621],
       [ 1.22474487,  0.          , -0.26726124],
       [-1.22474487,  1.22474487, -1.06904497]])
```

with_mean = False put mean to zero

with_std = False put std to zero

Scikit Learn: Dataset Transformations

Initialization of the methods

Fit of the parameters (mean and std in this case).

Application of the method to the data.

```
from sklearn import preprocessing
import numpy as np
X_train = np.array([[ 1., -1.,  2.],
                    [ 2.,  0.,  0.],
                    [ 0.,  1., -1.]])

scaler = preprocessing.StandardScaler(with_mean=True, with_std=True)
|
scaler.fit(X_train)

X_scaled = scaler.transform(X_train)

X_scaled
array([[ 0.          , -1.22474487,  1.33630621],
       [ 1.22474487,  0.          , -0.26726124],
       [-1.22474487,  1.22474487, -1.06904497]])
```

This is the standard workflow in sklearn

```
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X_train)
X_scaled
array([[ 0.          , -1.22474487,  1.33630621],
       [ 1.22474487,  0.          , -0.26726124],
       [-1.22474487,  1.22474487, -1.06904497]])
```

```
X_train_2 = scaler.inverse_transform(X_scaled)
X_train_2
array([[ 1.00000000e+00, -1.00000000e+00,  2.00000000e+00],
       [ 2.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 1.11022302e-16,  1.00000000e+00, -1.00000000e+00]])
```

Scikit Learn: Dataset Transformations

Sklearn.preprocessing.MinMaxScaler()

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
X_train = np.array([[ 1., -1.,  2.],
                    [ 2.,  0.,  0.],
                    [ 0.,  1., -1.]])
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))

X_scaled = scaler.fit_transform(X_train)
|
X_scaled
array([[0.5       ,  0.       ,  1.         ],
       [1.        ,  0.5      ,  0.33333333],
       [0.        ,  1.        ,  0.         ]])
```

Scaling is done by column!

Scikit Learn: Dataset Transformations

Sklearn.preprocessing.MinMaxScaler()

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
X_train = np.array([[ 1., -1.,  2.],  
                    [ 2.,  0.,  0.],  
                    [ 0.,  1., -1.]])  
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
```

```
X_scaled = scaler.fit_transform(X_train)
```

```
|  
X_scaled
```

```
array([[0.5, 0., 1.],  
       [1., 0.5, 0.33333333],  
       [0., 1., 0.]])
```

► Defines the output range.

Scikit Learn: Train-Test splitting

Sklearn.model_selection.train_test_split

Train test splitting is necessary for correctly evaluate the performance of your model and **avoiding overfitting**.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm

X, y = datasets.load_iris(return_X_y=True)
X.shape, y.shape
```

```
((150, 4), (150,))
```

```
X_train, X_test, y_train, y_test = train_test_split(\
    X, y, test_size=0.4, random_state=0,\
    shuffle=True)
```

```
X_train.shape
```

```
(90, 4)
```

```
X_test.shape
```

```
(60, 4)
```

Random_state is for reproducibility, do not fix it unless you are debugging.

test_size fraction of data to use as test.

Shuffle=True for reordering all data. Initial order is often biased.

Scikit Learn: Training

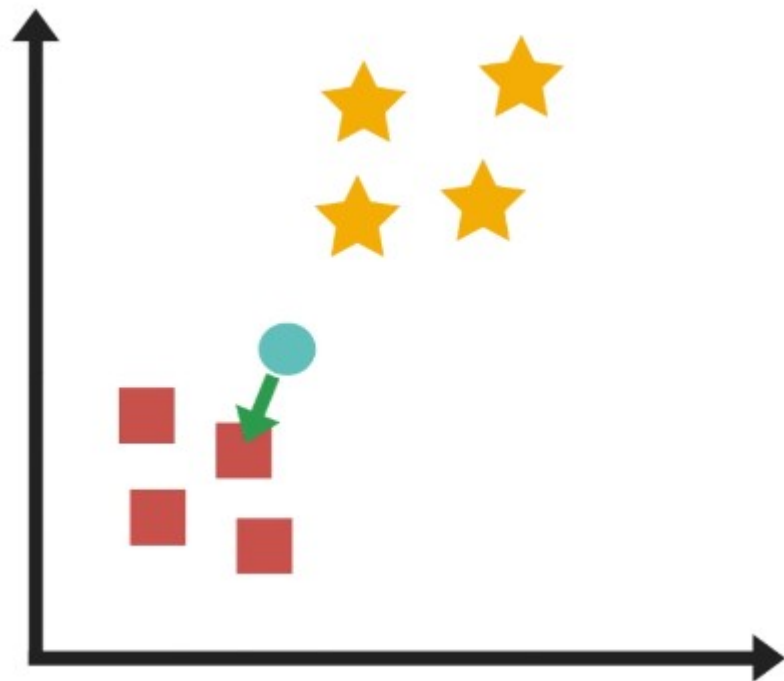
K-Nearest Neighbor (K-NN)

Lazy model.

Given a new pattern x , the NN assign to x the class of the **nearest pattern in the training set**.

You need a **notion of distance** (i.e. euclidean distance, ...).

You have to store in memory all the training set.
If your training set has dimension N for every new pattern you need to evaluate N distances.
Computational expensive for large datasets.



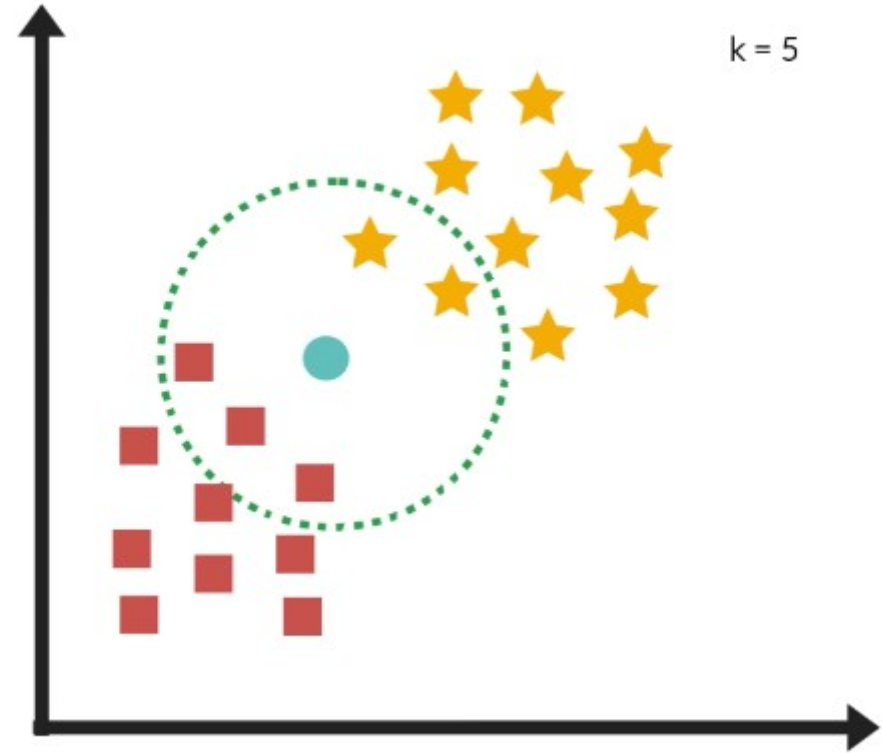
Nearest Neighbor does not learn, it memorises!

Scikit Learn: Training

K-Nearest Neighbor (K-NN)

KNN search among the K nearest pattern.

K is an **hyperparameter**. The best choice depends on you data. Usually $K \sim 5$ gives good results.



Scikit Learn: Training

K-Nearest Neighbor (K-NN)

Sklearn.neighbors.KNeighborsClassifier()

$$(|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

```
from sklearn.neighbors import KNeighborsClassifier

X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]

KNN = KNeighborsClassifier(n_neighbors=3,
                           p=2,
                           metric='minkowski',
                           n_jobs=1)

KNN.fit(X,y)

KNN.predict([[1.1]])

array([0])
```

Number of k nearest neighbor to consider

Exponent of the minkowskian distance

Type of distance to consider

Number of CPUs, -1 use all the availables

KNN

Scikit Learn: Training

K-Nearest Neighbor (K-NN)

Sklearn.neighbors.KNeighborsClassifier()

```
from sklearn.neighbors import KNeighborsClassifier

X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]

KNN = KNeighborsClassifier(n_neighbors=3,
                           p=2,
                           metric='minkowski',
                           n_jobs=1)

KNN.fit(X,y)

KNN.predict([[1.1]])

array([0])
```

$$(|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

Standardization and/or MinMax scaling
is fundamental.

Sklearn.neighbors.RadiusNeighborsClassifier()

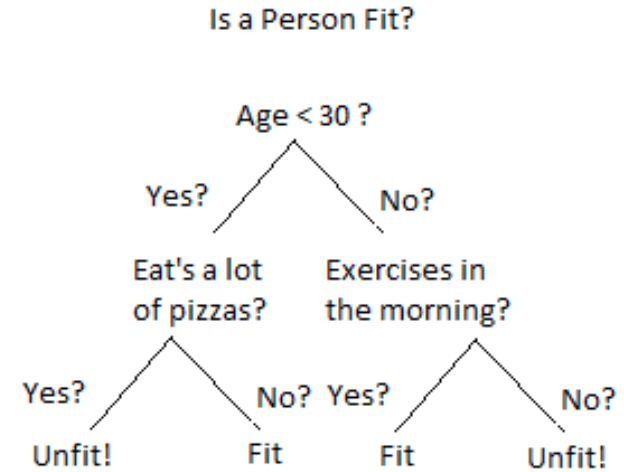
Scikit Learn: Training

Decision Trees

Non parametric learning methods that predicts the value of a target variable by learning simple decision rules inferred from the data features

The deeper the tree, the more complex the decision rules and the fitter the model.
(**Overfitting**)

They do **not need data preparation**.



How the trees decide the decision rules? Maximise the information gain

Scikit Learn: Training

Decision Trees

`sklearn.tree.DecisionTreeClassifier()`

```
from sklearn import tree
X = [[0, 0], [1, 1]]
Y = [0, 1]

tt = tree.DecisionTreeClassifier(max_depth=None,
                                min_samples_split=2,
                                min_samples_leaf=1)

tt = tt.fit(X, Y)

tt.predict([[2., 2.]])

array([1])
```

Is the maximum depth of the tree, if None goes on until all leaves are pure.

Minimum number of samples required to split an internal node.

The minimum number of samples required to be at a leaf node.

Decision Tree

Tuning of the hyperparameters.

Scikit Learn: Training

-x: features

-y: target

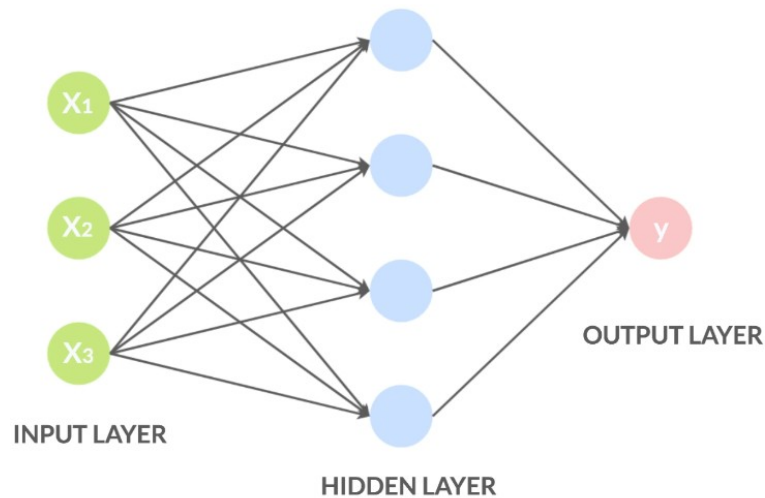
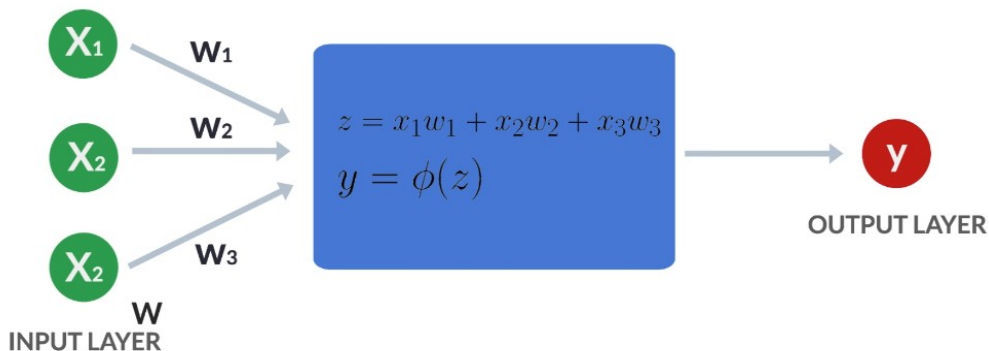
-w: weights (free parameters)

- ϕ : activation function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

$$\phi(z) = \max(0, z)$$

Multi-Layer Perceptron MLP



Scikit Learn: Training

Multi-Layer Perceptron MLP

sklearn.neural_network.MLPClassifier()

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

X, y = make_classification(n_samples=100, random_state=1)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    stratify=y,
                                                    random_state=1)

MLP = MLPClassifier(random_state=1,
                    hidden_layer_sizes=100,
                    max_iter=300, verbose=False,
                    )

MLP.fit(X_train, y_train)

MLP.predict(X_test[:5, :])

array([1, 0, 1, 0, 1])
```

```
MLP.predict_proba(X_test[:5, :])

array([[0.03838405, 0.96161595],
       [0.99738911, 0.00261089],
       [0.05044737, 0.94955263],
       [0.64599075, 0.35400925],
       [0.00753975, 0.99246025]])
```

Random_state is only for reproducibility

hidden_layer_sizes is the number of hidden layer

max_iter: maximum number of iteration for training

Verbose: print diagnostic on screen, always put to true

MLP

More parameters on documentation, check them for fine tuning .

Scikit Learn: Score

Many metrics available, the most useful one, for classification, is the **confusion matrix**.

You want high numbers on the main diagonal.

You want the same performance on train and test.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Scikit Learn: Score

`sklearn.metrics.confusion_matrix()`

```
from sklearn.metrics import confusion_matrix  
  
y_true = [1, 0, 1, 1, 0, 1]  
y_pred = [0, 0, 1, 1, 0, 1]  
  
confusion_matrix(y_true, y_pred)  
  
array([[2, 0],  
       [1, 3]])
```

2: True positive
3: True negative

0: False positive
1: False negative

Compare your model prediction
with the targets.

Scikit Learn: Score

`sklearn.metrics.accuracy_score()`

The number of predicted labels that match exactly the true label.

```
from sklearn.metrics import accuracy_score
y_pred = [0, 2, 1, 3]
y_true = [0, 1, 2, 3]
accuracy_score(y_true, y_pred)
```

0.5

```
accuracy_score(y_true, y_pred, normalize=False)
```

2

Gives the accuracy as a fraction



Scikit Learn: Training

Multi-Layer Perceptron MLP

sklearn.neural_network.MLPClassifier()

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

X, y = make_classification(n_samples=100, random_state=1)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    stratify=y,
                                                    random_state=1)

MLP = MLPClassifier(random_state=1,
                    hidden_layer_sizes=100,
                    max_iter=300, verbose=False,
                    )

MLP.fit(X_train, y_train)

MLP.predict(X_test[:5, :])

array([1, 0, 1, 0, 1])
```

```
MLP.predict_proba(X_test[:5, :])

array([[0.03838405, 0.96161595],
       [0.99738911, 0.00261089],
       [0.05044737, 0.94955263],
       [0.64599075, 0.35400925],
       [0.00753975, 0.99246025]])
```

Random_state is only for reproducibility

hidden_layer_sizes is the number of hidden layer

max_iter: maximum number of iteration for training

Verbose: print diagnostic on screen, always put to true

MLP

More parameters on documentation, check them for fine tuning .