<https://www.tutorialspoint.com/artificial_intelligence_with_python/index.htm>

SIMILAR A REGRESION, PERO NO NECESARIAMENTE LINEAL

In classification problem, we have the categorized output such as “Black” or “white” or “Teaching” and “Non-Teaching”.

If we want to check whether the image is of a car or not.

For checking this, we will build a training dataset having the two classes related to “car” and “no car”.

Then we need to train the model by using the training samples.

The classification models are mainly used in face recognition, spam identification, etc.

Case:[Breast Cancer Wisconsin Diagnostic Database.](http://scikit-learn.org/stable/datasets/index.html#breast-cancer-wisconsin-diagnostic-database)

569 instances with 30 attributes each, that label the cancer as malign or benign.

import sklearn

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

#target data

target\_names   = data['target\_names']  #nombre de columnas de target

target\_data    = data['target']        #array de datos de target data

# 2 columns for target  malignant / bening

# si la columna tiene 0 es falso

# Puede ser 0,0 ni maligno ni benigno

# Puede ser 1,0 / 0,1, si es 1,1 es clasificado como benigno y maligno

print(label\_names)

#arrays

feature\_names = data['feature\_names']  #nombres de features

features      = data['data']           #array de datos de features

print(feature\_names) #mean radius / mean texture / mean perimeter / mean area ...

The command below will show that they are mapped to binary values 0 and 1.

Here 0 represents malignant cancer and 1 represents benign cancer.

print(target\_names[0])

0

Informacion de quienes con cancer maligno en labels[1]

Informacion de quienes con cancer benigno en labels[0]

Para conocer cual es el nombre del feature[0]

print(feature\_names[0])

mean radius

Para ver los datos del feature[0], del feature ‘mean radius’

print(features[0])

[ **1.79900000e+01** 1.03800000e+01 1.22800000e+02 1.00100000e+03

1.18400000e-01 2.77600000e-01 3.00100000e-01 1.47100000e-01

2.41900000e-01 7.87100000e-02 1.09500000e+00 9.05300000e-01

8.58900000e+00 1.53400000e+02 6.39900000e-03 4.90400000e-02

5.37300000e-02 1.58700000e-02 3.00300000e-02 6.19300000e-03

2.53800000e+01 1.73300000e+01 1.84600000e+02 2.01900000e+03

1.62200000e-01 6.65600000e-01 7.11900000e-01 2.65400000e-01

4.60100000e-01 1.18900000e-01]

From the above output, we can see that the first data instance is a malignant tumor

Mean radius = 1.7990000e+01.

The command below will split the data into training (60%) and test data (40%).

from sklearn.model\_selection import train\_test\_split

train, test, train\_labels, test\_labels = train\_test\_split(features,labels,test\_size = 0.40, random\_state = 42)

To build the model we will to use Naïve Bayes algorithm. import the GaussianNB modu

from sklearn.naive\_bayes import GaussianNB

#initialize model

Model\_gnb = GaussianNB()

#train the model by fitting it to the data with Model\_gnb.fit()

model = Model\_gnb.fit(train, train\_labels)

With the train data we train the model to learn when the cancer is malign or benign,

With the test data we validate if the data trained is enough to predict, and how much is the accuracy.

NewData = Model\_gnb.predict(test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(test\_labels, NewData))

0.951754385965

The result shows that the NaïveBayes classifier is 95.17% accurate.

As it is very accuracy when we have a new item we can use a prediction model to now with 0.95% of accuracy if it has benign or malign cancer, according to the features:

‘mean radius' 'mean texture' 'mean perimeter' 'mean area'

'mean smoothness' 'mean compactness' 'mean concavity'

'mean concave points' 'mean symmetry' 'mean fractal dimension'

'radius error' 'texture error' 'perimeter error' 'area error'

'smoothness error' 'compactness error' 'concavity error'

'concave points error' 'symmetry error' 'fractal dimension error'

'worst radius' 'worst texture' 'worst perimeter' 'worst area'

'worst smoothness' 'worst compactness' 'worst concavity'

'worst concave points' 'worst symmetry' 'worst fractal dimension'

Array with data of the second row, which has all of the values of the features

Paciente1 = features[2] #valores segunda columna

temp = Paciente1.reshape(1,-1) #reshape para 1 fila

resultado = Model\_gnb.predict(temp)

print(resultado)

Resultado : 0

Es de agregar “,”

Paciente1=np.array([[3.969e+01,2.125e+01,4.300e+02,1.203e+03,1.096e-01,1.599e-01,1.974e-01,1.279e-01,2.069e-01,5.999e-02,7.456e-01,7.869e-01,4.585e+00,9.403e+01,6.150e-03,4.006e-02,3.832e-02,2.058e-02,2.250e-02,4.571e-03,2.357e+01,2.553e+01,1.525e+02,1.709e+03,1.444e-01,4.245e-01,4.504e-01,2.430e-01,3.613e-01,8.758e-02]])

NewData = Model\_gnb.predict(Paciente1)

print(NewData)

=== = = = =

import sklearn

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

#arrays

#target data

target\_names   = data['target\_names']  #nombre de columnas de target

target\_data    = data['target']        #array de datos de target data

# 2 columns for target  malignant / bening

# si la columna tiene 0 es falso

# Puede ser 0,0   ni maligno ni benigno

# Puede ser 1,0 / 0,1, si es 1,1 es clasificado como benigno y maligno

#arrays

feature\_names = data['feature\_names']  #nombres de features

features      = data['data']           #array de datos de features

print(feature\_names) #mean radius / mean texture / mean perimeter / mean area ...

from sklearn.model\_selection import train\_test\_split

#train\_labels train data

#test\_labels test data

train, test, train\_labels, test\_labels = train\_test\_split(features,target\_data,test\_size = 0.40, random\_state = 42)

from sklearn.naive\_bayes import GaussianNB

#initialize model

Model\_gnb = GaussianNB()

#train the model by fitting it to the data with Model\_gnb.fit()

model = Model\_gnb.fit(train, train\_labels)

NewData = Model\_gnb.predict(test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(test\_labels, NewData))

= = = = = =

import sklearn

from sklearn import svm, datasets

iris = datasets.load\_iris()

features = iris.data[:, :2]

target\_data = iris.target  # por ahora no usamos target

from sklearn.model\_selection import train\_test\_split

train, test, train\_labels, test\_labels = train\_test\_split(features,target\_data,test\_size = 0.40, random\_state = 42)

from sklearn.naive\_bayes import GaussianNB

#initialize model

Model\_gnb = GaussianNB()

#train the model by fitting it to the data with Model\_gnb.fit()

model = Model\_gnb.fit(train, train\_labels)

#CON LA DATA DE TEST SE CREA EL ARRAY DE PREDCCIONES

ArrPredicciones = Model\_gnb.predict(test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(test\_labels, ArrPredicciones))