**ML BASICS**

**Numpy**-is a numerical python, quite a powerful tool for numerical operations in Python.

import numpy as np

 importing a library means loading it into the memory and then it’s there for you to work with.

Helps to wIork with array and create a n dimensional array.

**Indexing and slicing.**

array[0:5] will return the first five elements

array[0:5,4] will return the first five elements in column 4

 array[:2] to get elements from the beginning until index 2 (not including index 2)

array[2:] to return from the 2nd index until the end of the array.

array[:,1] will return the elements at index 1 on all rows.

array[1]=4 will assign the value 4 to the element on index 1; you can do it to multiple values: array[1,5]=10

**Sorting and Reshaping**

array.sort() can be used to sort your NumPy array

array.T will transpose an array

array.reshape(x,y) would reshape your array to the size you set with x and y.

array.resize((x,y)) will change the array shape to x and y and fill new values with zeros.

## Combining and Splitting

Row=0 col=1

np.concatenate((array1,array2),axis=1) will add array 2 as columns to the end of array 1. np.split(array,2) will spilt the array into two sub-arrays

## Adding and Removing Elements

* np.append(array,values) will append values to end of array.
* np.insert(array, 3, values)will insert values into array before index 3
* np.delete(array, 4, axis=0)will delete row on index 4 of array
* np.delete(array, 5, axis=1) will delete column on index 5 of array

## Descriptive Statistics

* np.mean(array,axis=0) will return mean along specific axis (0 or 1)
* array.sum() will return the sum of the array
* array.min()will return the minimum value of the array
* array.max(axis=0)will return the maximum value of specific axis
* np.var(array)will return the variance of the array
* np.std(array,axis=1)will return the standard deviation of specific axis
* array.corrcoef()will return the correlation coefficient of the array
* numpy.median(array) will return the median of the array elements

## Doing Math with NumPy

Any tutorial to NumPy would not be complete without the numerical and mathematical operations you can do with NumPy! Let’s go over them:

np.add(array ,1) will add 1 to each element in the array and np.add(array1,array2) will add array 2 to array 1. The same is true to np.subtract(), np.multiply(), np.divide() and np.power() — all these commands would work in exactly the same way as described above.

You can also get NumPy to return different values from the array, like:

* np.sqrt(array) will return the square root of each element in the array
* np.sin(array) will return the sine of each element in the array
* np.log(array) will return the natural log of each element in the array
* np.abs(arr) will return the absolute value of each element in the array
* np.array\_equal(arr1,arr2) will return True if the arrays have the same elements and shape

**PANDAS**

it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical

# **Working with Pandas**

## Loading and Saving Data with Pandas

import pandas as pd

pd.read\_filetype() replace “filetype” with the actual, well, filetype (like CSV).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

pd.DataFrame()

You can also save a data frame you’re working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

df.to\_filetype(filename)

## Viewing and Inspecting Data

first n rows with df.head(n) or the last n rows with df.tail(n).

 df.shape would give you the number of rows and columns.

df.info() would give you the index, datatype and memory information

df.describe() which inputs summary statistics for numerical columns.

* df.mean()Returns the mean of all columns
* df.corr()Returns the correlation between columns in a data frame
* df.count()Returns the number of non-null values in each data frame column
* df.max()Returns the highest value in each column
* df.min()Returns the lowest value in each column
* df.median()Returns the median of each column
* df.std()Returns the standard deviation of each column

## Selection of Data

(df[col])  return column with label col as Series

(df[[col1, col2]]) returns columns as a new DataFrame.

ILOC-gets rows (or columns) at particular positions in the index

LOC-gets rows (or columns) with particular labels from the index.

 In order to select the first row you can use df.iloc[0,:]

## Filter, Sort and Groupby

sort values in a certain column in an ascending order using df.sort\_values(col1) ;

 descending order using df.sort\_values(col2,ascending=False).  sort values by col1 in ascending order then col2 in descending order by using df.sort\_values([col1,col2],ascending=[True,False]).

Data cleaning

pd.isnull() which checks for null Values, and returns a boolean array (an array of true for missing values and false for non-missing values).

In order to get a sum of null/missing values, run pd.isnull().sum()

 pd.notnull() is the opposite of pd.isnull().

After you get a list of missing values you can get rid of them, or drop them by using df.dropna() to drop the rows or df.dropna(axis=1) to drop the columns.

A different approach would be to fill the missing values with other values by using df.fillna(x) which fills the missing values with x .

s.replace(1,'one') would replace all values equal to 1 with 'one'.

It’s possible to do it for multiple values: s.replace([1,3],['one','three']) would replace all 1 with 'one' and 3 with 'three'.

You can also rename specific columns by running: df.rename(columns={'old\_name': 'new\_ name'}) or use

 df.set\_index('column\_one') to change the index of the data frame.

<https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673>

**Confusion Matrix**

The confusion matrix of a classifier summarizes the TP, TN, FP, FN measures of performance of our model.

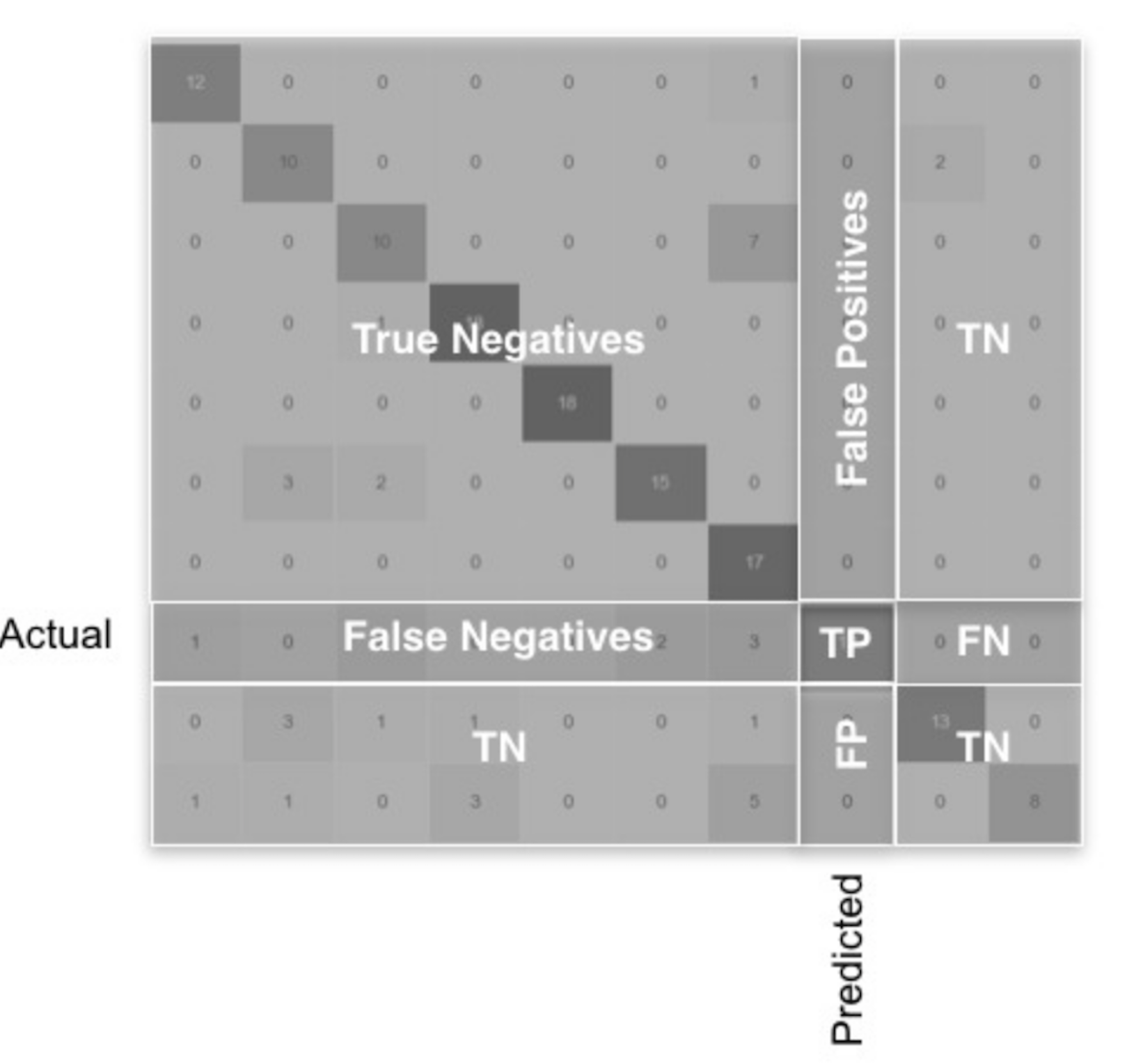
Actual

Predicted

|  |  |  |  |
| --- | --- | --- | --- |
| TOTAL | True(COL1) | False(COL2) |  |
| Positive(ROW1) | True Positive  (Truly classified as positive) | False Positive  (Falsely classified as positive) | Precision=TP/ROW1 |
| Negative(ROW2) | False negative  (Falsely classified as negative) | True negative(EXCH)  (Truly classified as negative) |  |
|  | TPR,Recall,sensitivity  =TP/COL1 | FPR,Fall-out  =FP/COL2 |  |
|  | FNR,Miss rate  =FN/Col1 | Specifity,TNR,Selectivity  =TN/COL2 |  |

* TP=The actual value was positive(TRUE) and the model predicted a positive value
* TN-The actual value was negative(FALSE) and the model predicted a negative value
* FP-TYPE1-The actual value was negative (False)but the model predicted a positive value
* FN-TYPE2-The actual value was positive(True) but the model predicted a negative value

1. Accuracy (all **correct** / all) = TP + TN / TP + TN + FP + FN
2. Misclassification (all **incorrect** / all) = FP + FN / TP + TN + FP + FN
3. **Precision** (**true** positives / **predicted** positives) = TP / TP + FP. *Precision tells us how many of the correctly* ***predicted*** *cases actually turned out to be positive.*
4. Sensitivity aka Recall (**true** positives / all **actual** positives) = TP / TP + FN
5. Specificity (**true** negatives / all **actual** negatives) =TN / TN + FP. *Recall tells us how many of the actual positive cases we were able to predict correctly with our model.*
6. *F1-SCORE*- =2\*R\*P/R+P.



FP = cnf\_matrix.sum(axis=0) - np.diag(cnf\_matrix)   
FN = cnf\_matrix.sum(axis=1) - np.diag(cnf\_matrix)  
TP = np.diag(cnf\_matrix)  
TN = cnf\_matrix.sum() - (FP + FN + TP)

<https://towardsdatascience.com/multi-class-classification-extracting-performance-metrics-from-the-confusion-matrix-b379b427a872#:~:text=The%20confusion%20matrix%20of%20a,%2C%20FPR%2C%20FNR%20and%20accuracy>.