Tien Rast

Unit 4

Individual Project

In this project I chose Prostate Cancer dataset to apply K-nearest neighbor algorithm. This dataset includes 100 observations and 10 variables.

The variables include: ID, radius, texture,area, smoothness, compactness, diagnosis\_result, symmetry, and fractal dimension

First, I will import the data and necessary functions to apply KNN algorithm, then take a look at the top 10 rows of the dataset

Code:

import pandas as pd

import os

import numpy as np

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

from sklearn.preprocessing import MinMaxScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

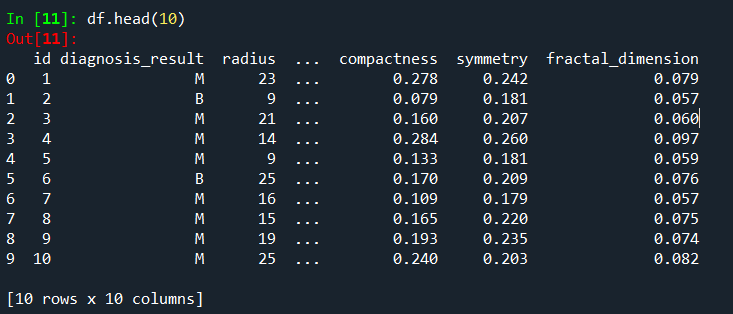
from sklearn.metrics import accuracy\_score

df = pd.read\_csv('C:/Users/tiera/Downloads/Prostate\_Cancer.csv')

df.head(10)

print (df)

Result:

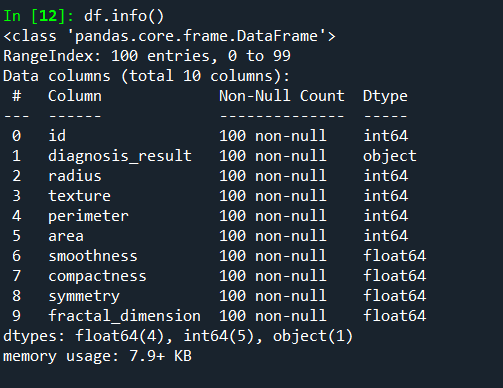


Then let’s take a look to see what type of data in the dataset an if there is any unknown data:

Code:

Df.info()

Result

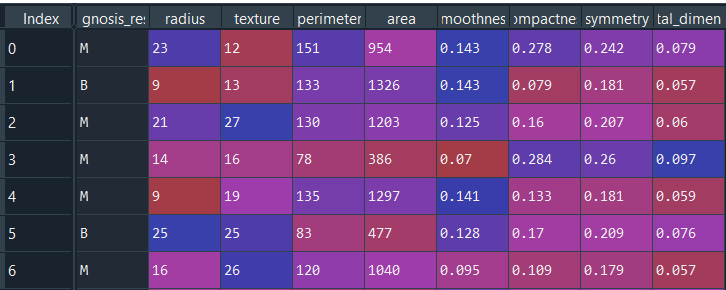


There is no unknown data, 1 categorical and the rest is numberical data.

The ID column is not necessery to be included in the analysis. So that, I will drop it from the dataset.

Code:

df.drop(['id'],axis =1, inplace=True)

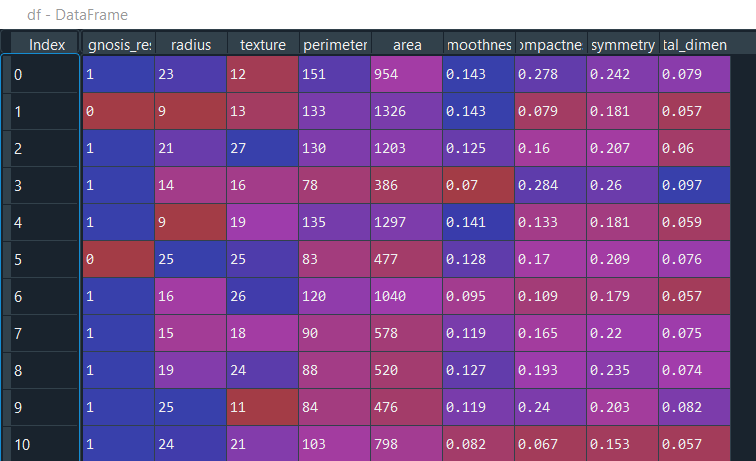


Diagnosis\_result is the outcome, and we want it to be numerical. I will convert M to 1 and 0 for the rest.

Code:

df.diagnosis\_result = [1 if each == 'M' else 0 for each in df.diagnosis\_result]

Result:



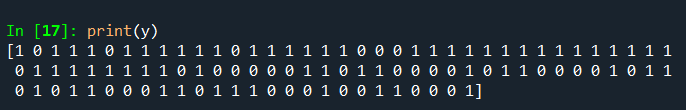
Now I will assign x and y values for test-train data split. Let’s y be our outcome which is the diagnosis results and x will be all the variables: radius, texture, area, and so on.

Code:

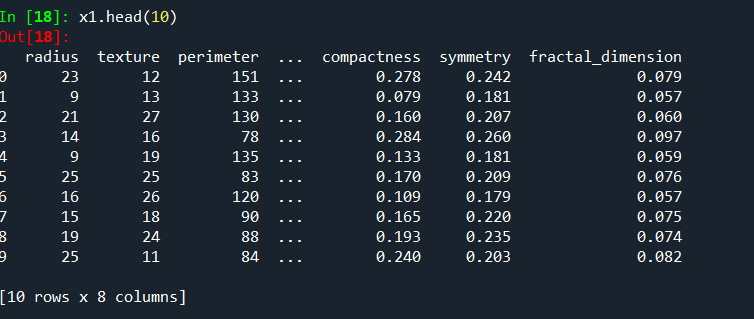
y= df.diagnosis\_result.values

x1 = df.drop(['diagnosis\_result'],axis=1)

Let’s take a look at y:



And x1:



Now we will transform our variable data x1;so that they can be all on the same scale between 0 and 1. This process is called Normalization and I will use MinMaxScaler function in Python to do it.

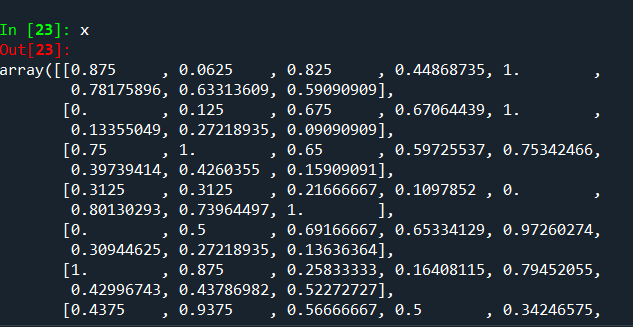
Code:

scaler = MinMaxScaler(feature\_range = (0,1))

x = scaler.fit\_transform(x1)

x

Result:



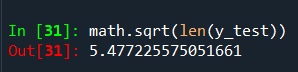
As we can see, all the independent variables were put in an array and their values are between 0 and 1. This process helps us to obtain a balance data to work with.

Now we will split the data in to 2 parts: train and test by using the function below:

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,random\_state = 0, test\_size = 0.3)

We will train 70 percent of the data and leave 30 percent to test.

Now we will choose K nearest number by square root the length of y test and the result is: 5.4777



So k will be 5

Next, I will define the model using k=5, p=2 (there only 2 values: 1 or 0 for the output), and Euclidean as the metric

Code:

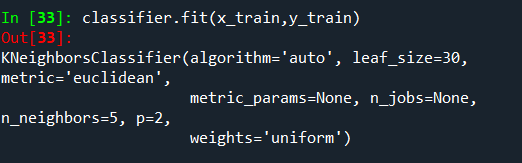
classifier = KNeighborsClassifier(n\_neighbors=5, p=2, metric = 'euclidean')

Then we fit the data into KNN model

Code:

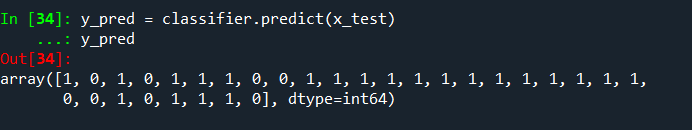
Classifier.fit(x\_train,y\_train)

Output:

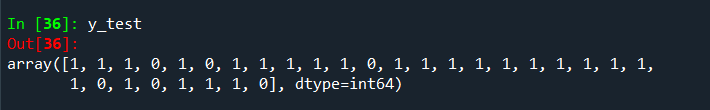


So how do we know that our model fit well with the data?

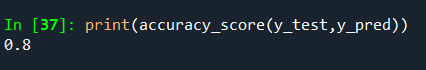
Let predict the test result:



How is that compared with the real y\_test?



Its hard to see by eyes, I will check the accuracy score:



80% accuracy is pretty good.