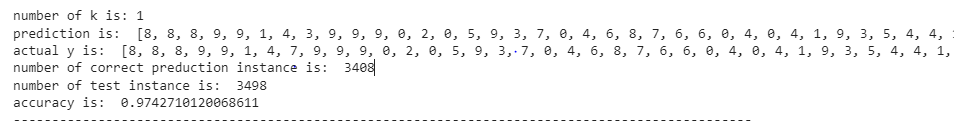
**KNN from scratch decumentation:**

1. Reading training data “data was 17 col and 7494 row”
2. Check for null “data didn’t have nulls” and see the corr between features
3. Splitting data into inputs(x\_train ) and outputs(y\_train)
4. Apply normalization to x\_train using this “x\_train=x\_train-mean/std”
5. Read test data “data was 17 col and 3498 row”
6. Check for null “data didn’t have nulls”
7. Splitting data into inputs(x\_test) and outputs(y\_test)
8. Apply normalization to x\_test using means and stds of x\_train this “x\_test=x\_test-mean/std”
9. Change x\_train, y\_train, x\_test and y\_test from dataframe to numpy array
10. Create knn class that take the number of k I want and class have 3 functions

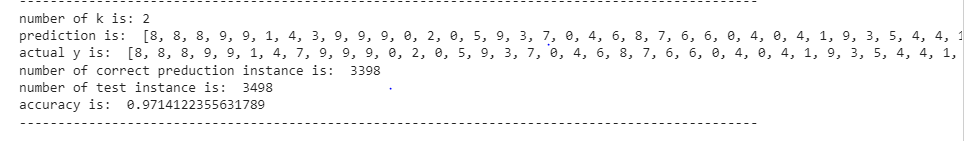
* Equlidean\_distance : take x,y and return distances between them”
* Fit : take x and y and assigning to his class att return NA “used in prediction function”
* Predict : take one argument, inside it handling ties and return list of prediction output

1. Create a for loop to take 9 object from class knn with different k in range from (1,9)

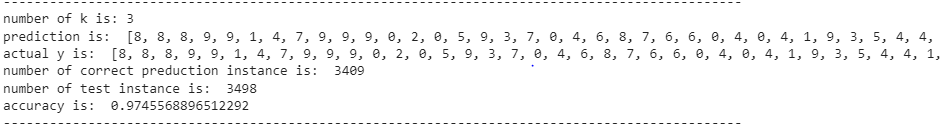
* Report for the first object with k=1



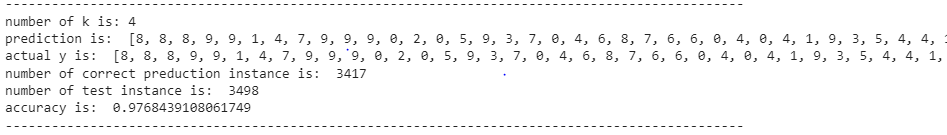
* Report for the sec object with k=2



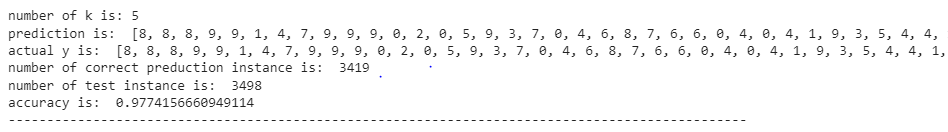
* Report for the 3th object with k=3



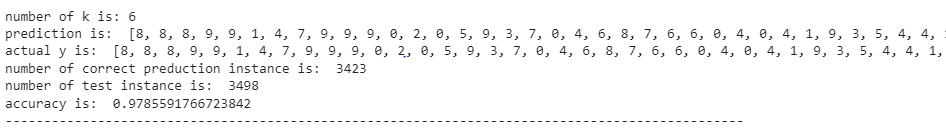
* Report for the fourth object with k=4



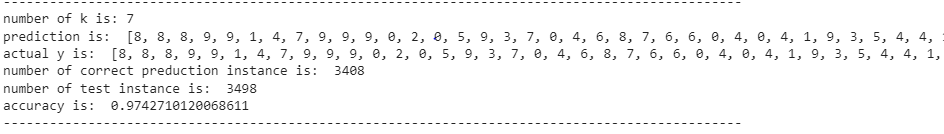
* Report for the 5th object with k=5



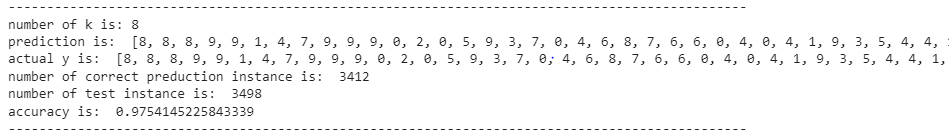
* Report for the 6th object with k=6



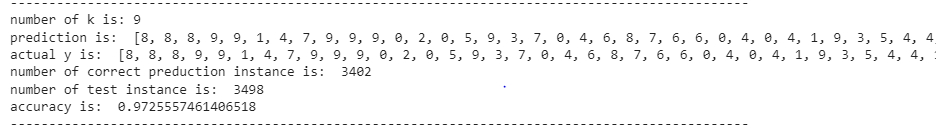
* Report for the seven object with k=7



* Report for the 8th object with k=8



* Report for the 9th object with k=9



1. **Finally,** the better before tie is in k=3 **”3406,** **0.9736992567181246” but** after tie is in k=6 **”** **3423,** **0.9785591766723842”**

**Used libraries:**

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sn

import warnings

warnings.filterwarnings('ignore')

import math

from google.colab import drive