K-NEAREST NEIGHBOR (KNN)

In

Machine Learning

Subject code: 4IT47

Academic Year: 2022-23

Name: Sanket M. Detroja

ID No.: 19IT413

Faculty Name: Dr. Zankhana Shah

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Birla Vishvakarma Mahavidyalaya Engineering College,

Vallabh Vidyanagar-388120

Chapter 1: What is Machine learning?

Machine learning ML is a type of artificial intelligence AL that allows applications to become more accurate at predicting outcomes without being explicitly programmed to do so. ML uses the past historical data as input to predict the new output values.

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behaviour. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

Machine learning is a discipline of artificial intelligence that provides machines with the ability to automatically learn from data and past experiences while identifying patterns to make predictions with minimal human intervention.

Machine learning methods enable computers to operate automatically without explicit programming. ML applications are with new data, and they can independently learn, grow, develop and adapt.

Machine learning derives insightful information from large volumes of data by leveraging algorithms to identify patterns and learn in an iterative process. ML algorithm use computation methods to learn directly from data instead of relying on any predetermined equation that may serve as a model.

Architecture of Machine Learning:

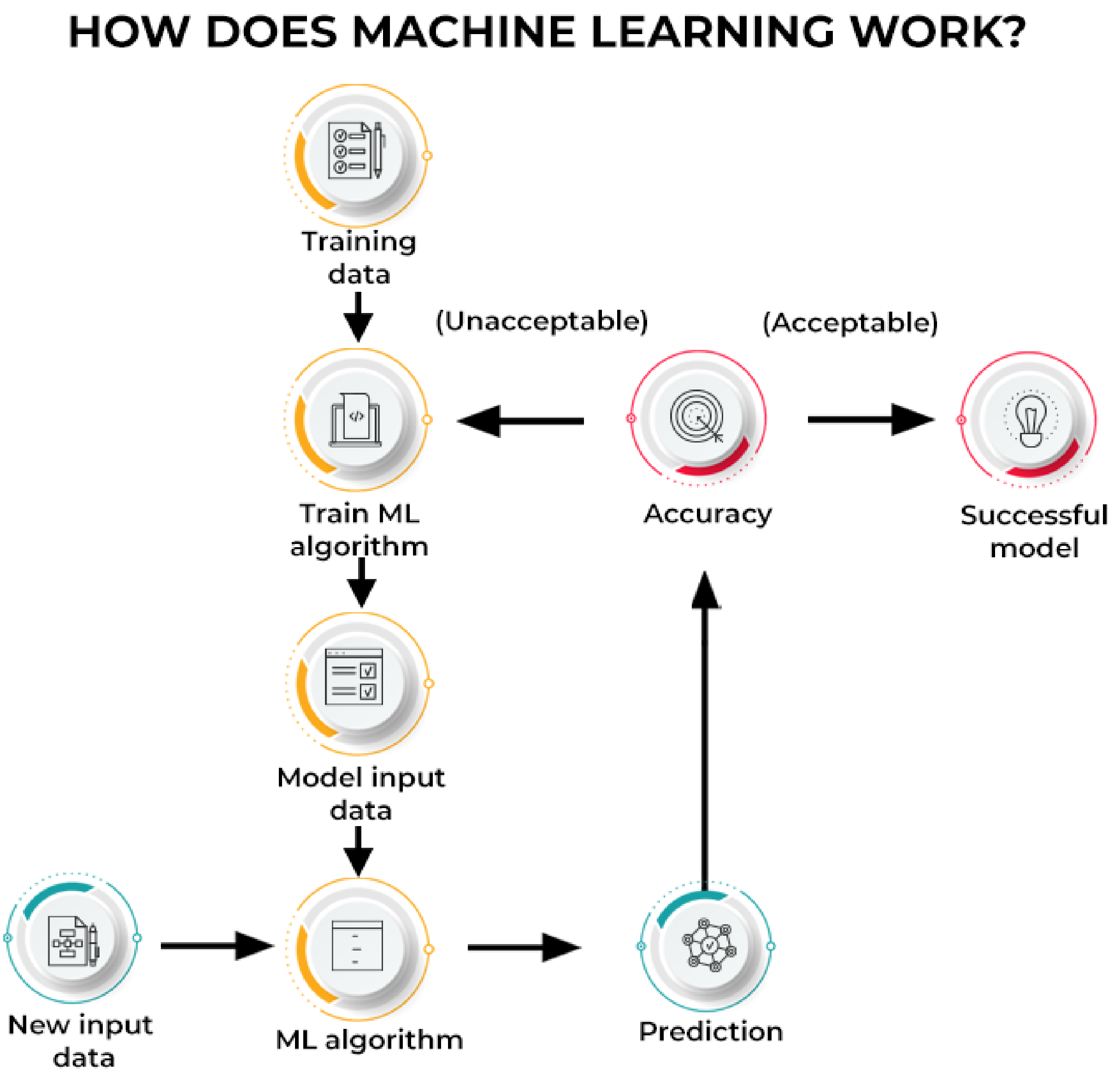


Fig 1 Architecture of ML

Chapter 2: What is K-Nearest Neighbor?

K-Nearest Neighbour is one of the simplest Machine Learning algorithm based on **Supervised Learning** (samples having labelled data) technique. It assumes the similarity between the case or the data and available cases and put the new case into the category that is most similar to the available categories.

KNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it **can be easily** classifies into a well suited category by using KNN. It can be used for regression as well as for classification but **mostly it is used for classification problems**.

KNN is a **non-parametric algorithm** that means it does **not make any assumption** in its own rather it just follows the underlying code. It is also known as **LAZY LEARNER ALGORITHM** because it does not learn from the training data immediately instead it shores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: assume, we have an image of creation that looks similar to hexagon and pentagon, but we want to know either it is a pentagon or hexagon. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the pentagon and hexagon images and based on the most similar features it will put it in either any of the shape category.

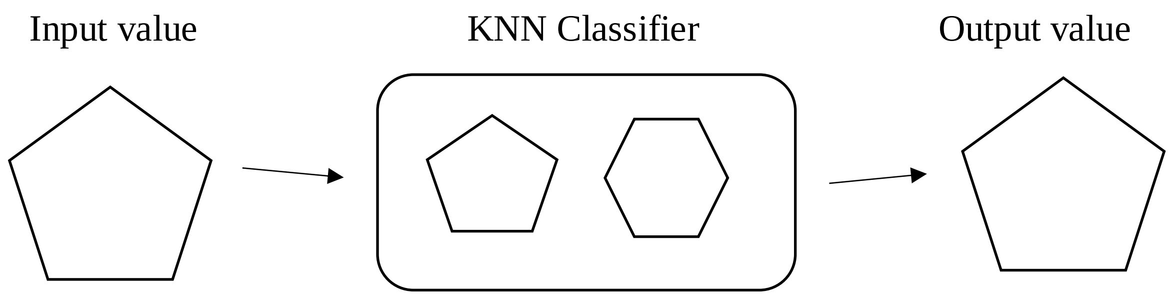


Fig 2 Basic understanding diagram

Chapter 3: About KNN.

3.1 Need of KNN:

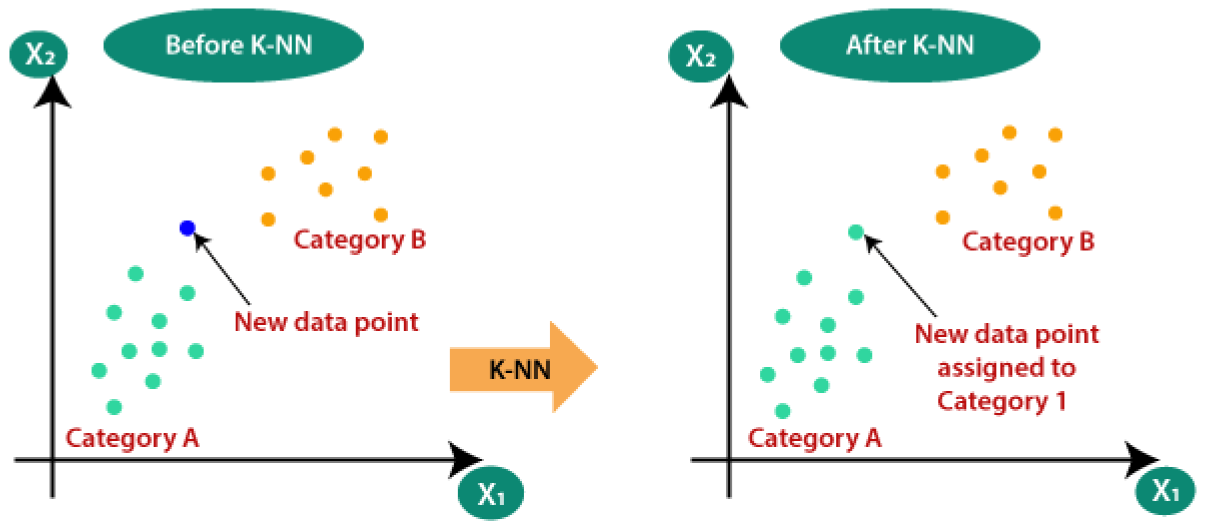


Fig 3 Graphical representation

Suppose we have two categories A green one and B yellow one. Here a new data blue comes in the picture and our model has to kind whether it goes to category A or B and KNN helps to find it easily.

3.2 Architecture

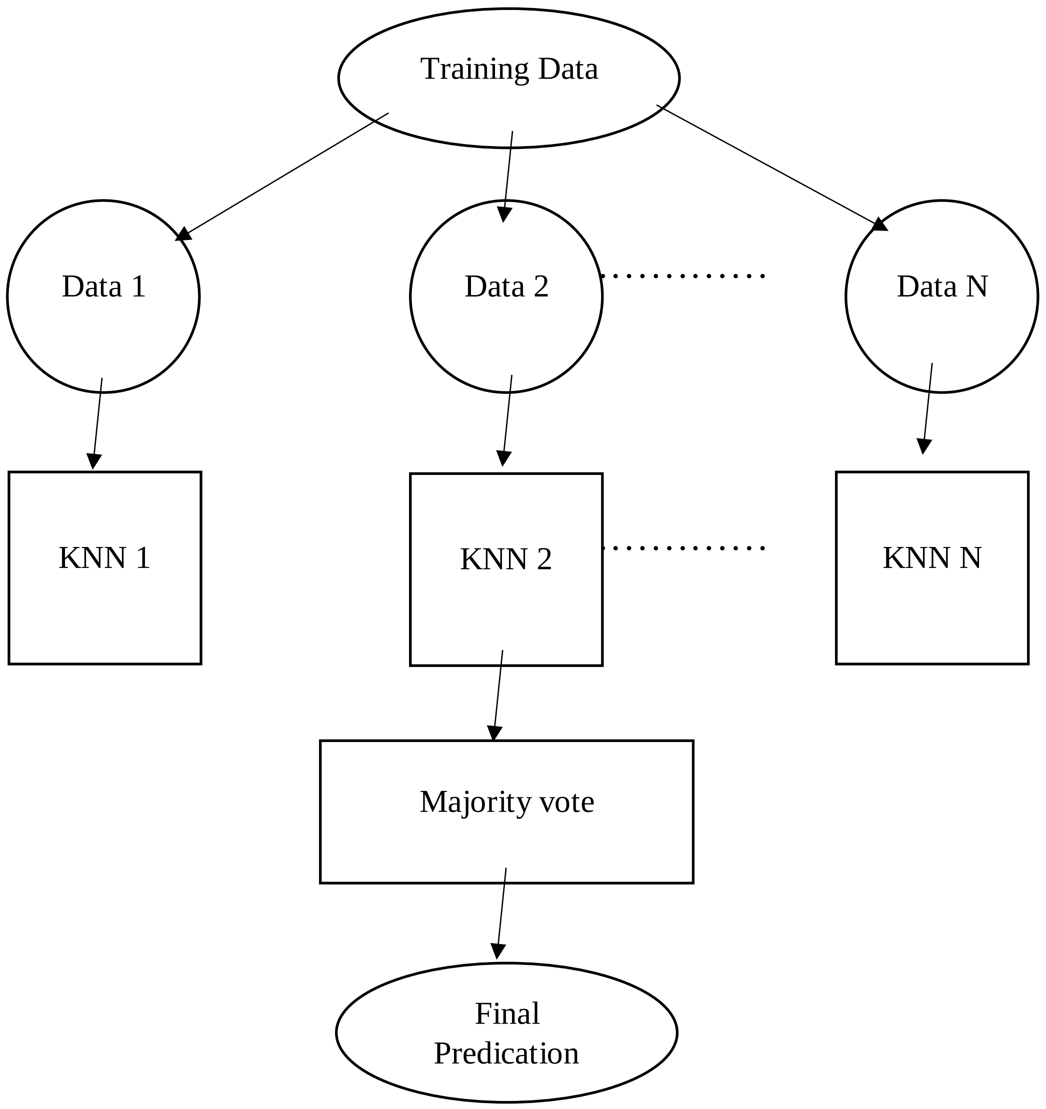


Fig 4 Architecture

Here the different training data must be provided at the initial stage. This data has divided among the small dataset and apply KNN to it and the majority vote has been assigned for the prediction of the training set data and the final output must be the same.

3.3 Application

I. Text mining
II. Agriculture
III. Finance
IV. Medical
V. Facial recognition
VI. Recommendation system

Chapter 4: Mathematical Example.

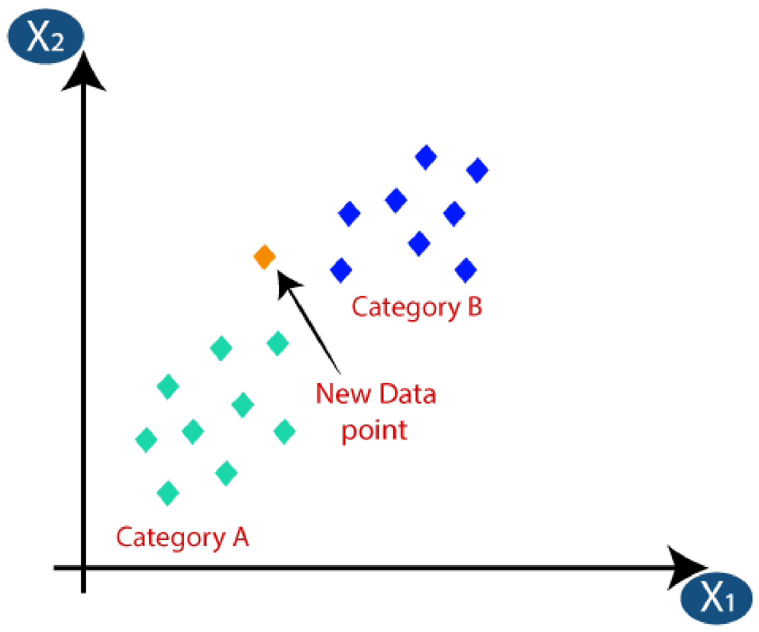


Fig 5 Graphical Representation

Here two categories shown in picture, category A and B and other training data sample is there and we have to identify the categories of that new sample. Following steps must be followed:

Step-1: Select the number K of the neighbors (K means minimum number of vote that has majority side).

How to select K value:

- There is no particular way to determine the best for K, so we need to try some values to find the best out them.
- If we take K **value 1 or 2 can be very noisy** and lead to effects of outliers into the model.
- If we take **k value very high that can lead us to very complex** calculation.
- So the most preferred value **for k is 5**.

Step-2: Calculate the Euclidean distance of k number of neighbors to the training data.

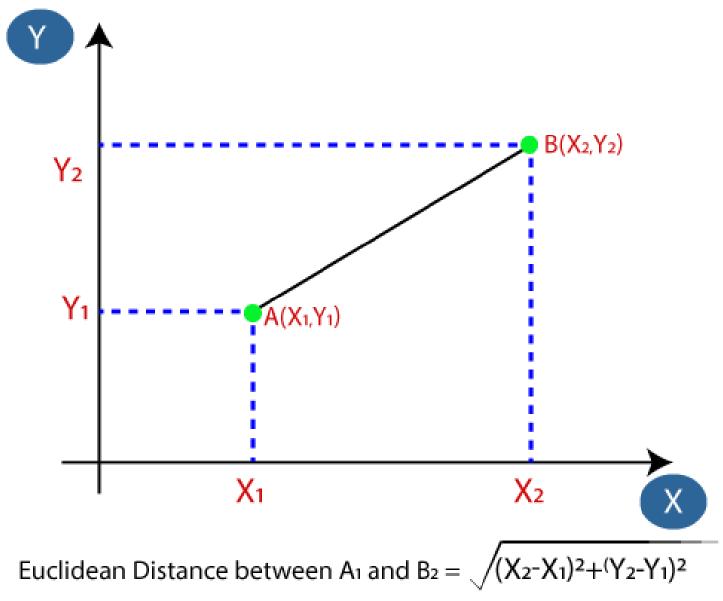


Fig 6 Eclidean Distance Formula

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of data points in each category. (that means which category has maximum number in count k must be assigned to new sample data points).

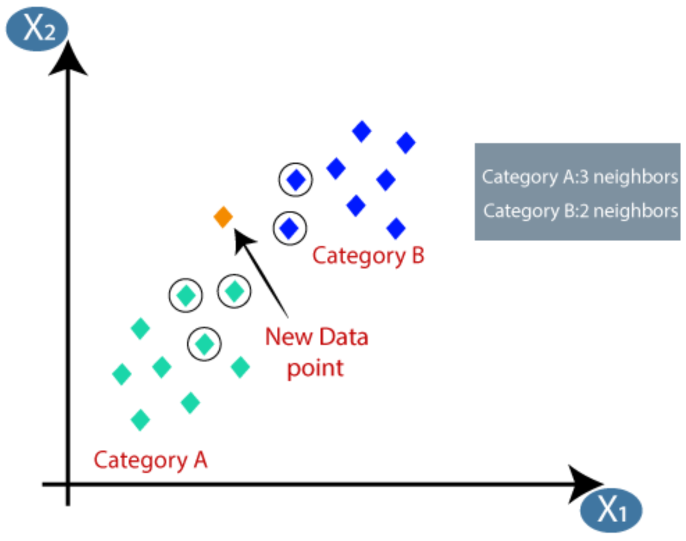


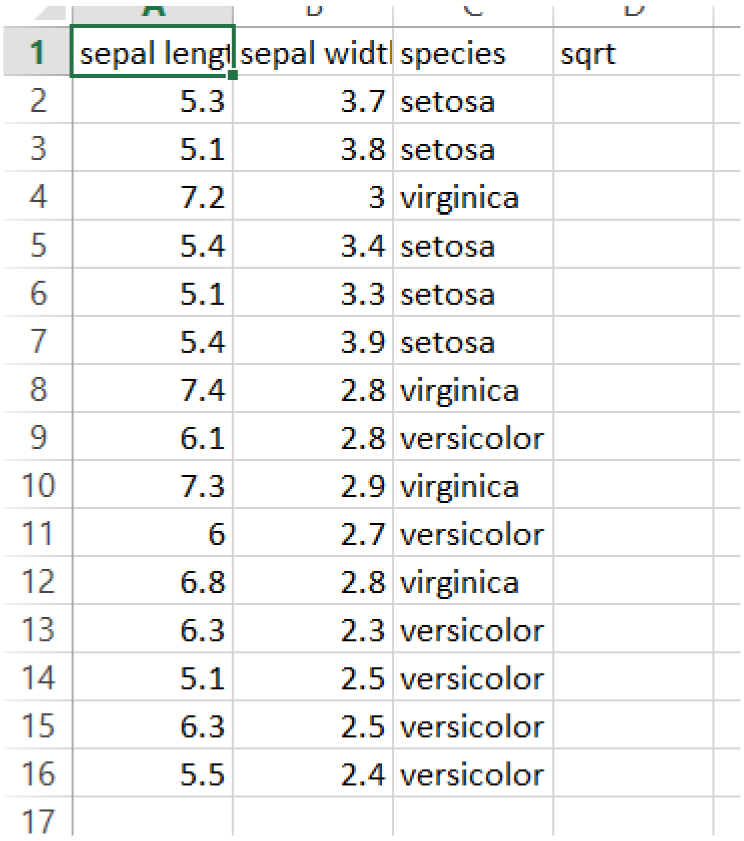
Fig 7 Finding K value

Step-5: Assign the new data points to that category for which the number of the neigbors is maximum. (Here category A has the maximum number of occurrence in k value so its key point has been assigned to new sample data point).

Step-6: Our model is ready.

Chapter 5: KNN with and without sklearn library.

Dataset:

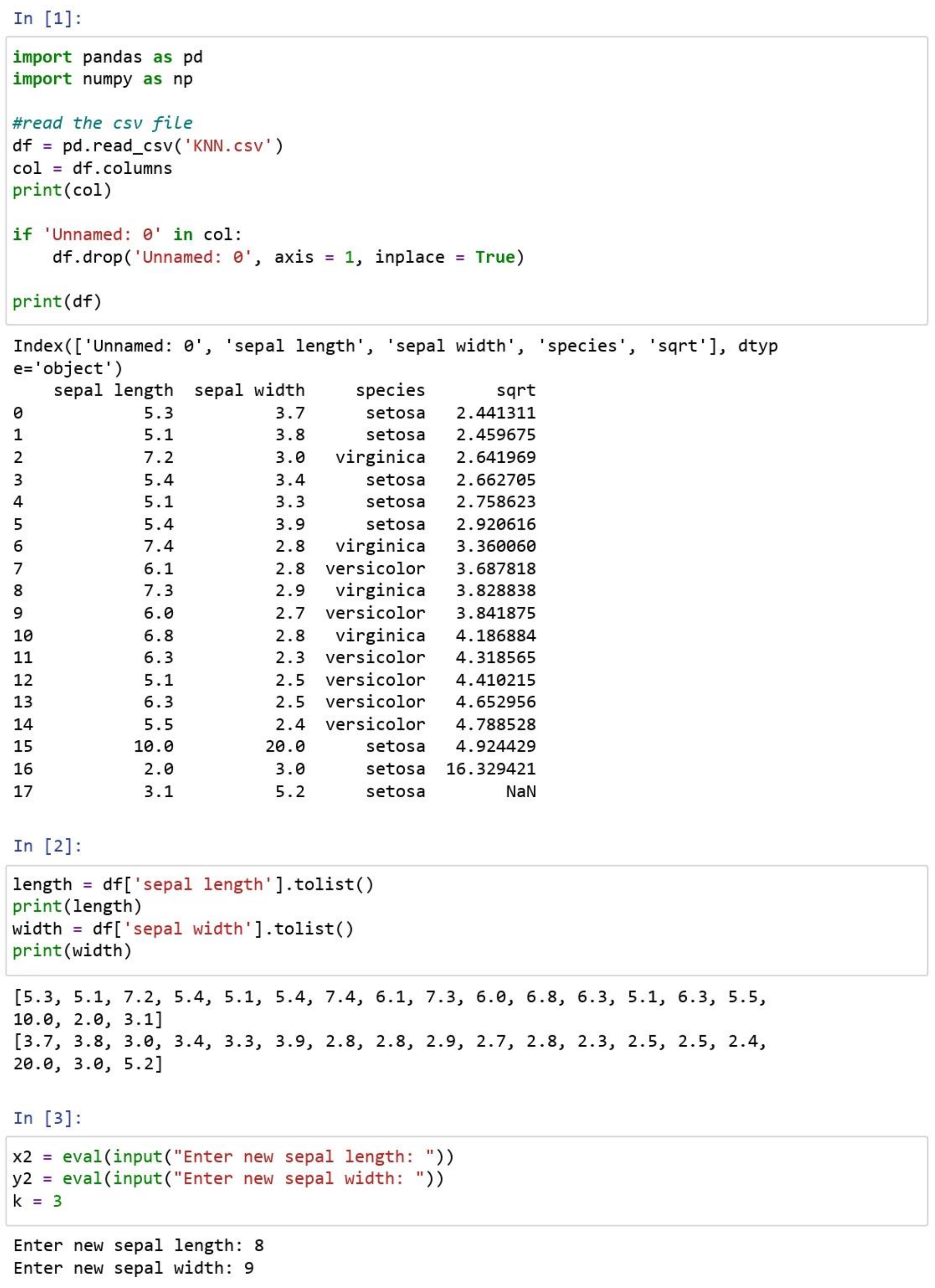


	A	B	C	D
1	sepal lengt	sepal widtl	species	sqrt
2	5.3	3.7	setosa	
3	5.1	3.8	setosa	
4	7.2	3	virginica	
5	5.4	3.4	setosa	
6	5.1	3.3	setosa	
7	5.4	3.9	setosa	
8	7.4	2.8	virginica	
9	6.1	2.8	versicolor	
10	7.3	2.9	virginica	
11	6	2.7	versicolor	
12	6.8	2.8	virginica	
13	6.3	2.3	versicolor	
14	5.1	2.5	versicolor	
15	6.3	2.5	versicolor	
16	5.5	2.4	versicolor	
17				

Fig 8 Dataset

- KNN without sklearn

Code:



```
In [1]:
```

```python
import pandas as pd
import numpy as np

#read the csv file
df = pd.read_csv('KNN.csv')
col = df.columns
print(col)

if 'Unnamed: 0' in col:
    df.drop('Unnamed: 0', axis = 1, inplace = True)

print(df)
```

```
Index(['Unnamed: 0', 'sepal length', 'sepal width', 'species', 'sqrt'], dtyp
e='object')
    sepal length  sepal width     species       sqrt
0            5.3          3.7      setosa   2.441311
1            5.1          3.8      setosa   2.459675
2            7.2          3.0   virginica   2.641969
3            5.4          3.4      setosa   2.662705
4            5.1          3.3      setosa   2.758623
5            5.4          3.9      setosa   2.920616
6            7.4          2.8   virginica   3.360060
7            6.1          2.8  versicolor   3.687818
8            7.3          2.9   virginica   3.828838
9            6.0          2.7  versicolor   3.841875
10           6.8          2.8   virginica   4.186884
11           6.3          2.3  versicolor   4.318565
12           5.1          2.5  versicolor   4.410215
13           6.3          2.5  versicolor   4.652956
14           5.5          2.4  versicolor   4.788528
15          10.0         20.0      setosa   4.924429
16           2.0          3.0      setosa  16.329421
17           3.1          5.2      setosa        NaN
```

```
In [2]:
```

```python
length = df['sepal length'].tolist()
print(length)
width = df['sepal width'].tolist()
print(width)
```

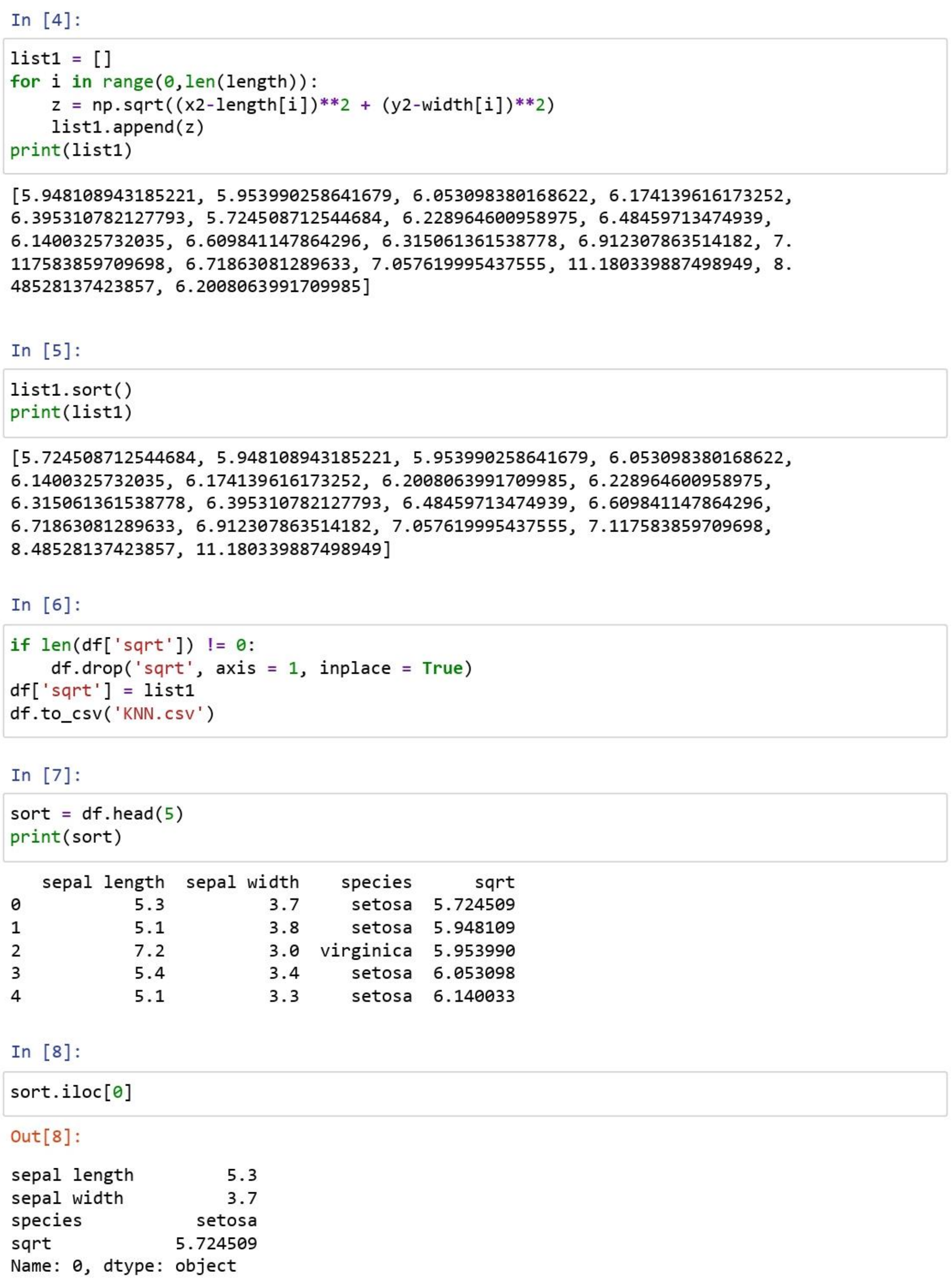
```
[5.3, 5.1, 7.2, 5.4, 5.1, 5.4, 7.4, 6.1, 7.3, 6.0, 6.8, 6.3, 5.1, 6.3, 5.5,
10.0, 2.0, 3.1]
[3.7, 3.8, 3.0, 3.4, 3.3, 3.9, 2.8, 2.8, 2.9, 2.7, 2.8, 2.3, 2.5, 2.5, 2.4,
20.0, 3.0, 5.2]
```

```
In [3]:
```

```python
x2 = eval(input("Enter new sepal length: "))
y2 = eval(input("Enter new sepal width: "))
k = 3
```

```
Enter new sepal length: 8
Enter new sepal width: 9
```

Fig 9 KNN algorithm without sklearn



In [4]:

```
list1 = []
for i in range(0,len(length)):
    z = np.sqrt((x2-length[i])**2 + (y2-width[i])**2)
    list1.append(z)
print(list1)
```

```
[5.948108943185221, 5.953990258641679, 6.053098380168622, 6.174139616173252,
6.395310782127793, 5.724508712544684, 6.228964600958975, 6.48459713474939,
6.1400325732035, 6.609841147864296, 6.315061361538778, 6.912307863514182, 7.
117583859709698, 6.71863081289633, 7.057619995437555, 11.180339887498949, 8.
48528137423857, 6.2008063991709985]
```

In [5]:

```
list1.sort()
print(list1)
```

```
[5.724508712544684, 5.948108943185221, 5.953990258641679, 6.053098380168622,
6.1400325732035, 6.174139616173252, 6.2008063991709985, 6.228964600958975,
6.315061361538778, 6.395310782127793, 6.48459713474939, 6.609841147864296,
6.71863081289633, 6.912307863514182, 7.057619995437555, 7.117583859709698,
8.48528137423857, 11.180339887498949]
```

In [6]:

```
if len(df['sqrt']) != 0:
    df.drop('sqrt', axis = 1, inplace = True)
df['sqrt'] = list1
df.to_csv('KNN.csv')
```

In [7]:

```
sort = df.head(5)
print(sort)
```

```
   sepal length  sepal width    species      sqrt
0           5.3          3.7     setosa  5.724509
1           5.1          3.8     setosa  5.948109
2           7.2          3.0  virginica  5.953990
3           5.4          3.4     setosa  6.053098
4           5.1          3.3     setosa  6.140033
```

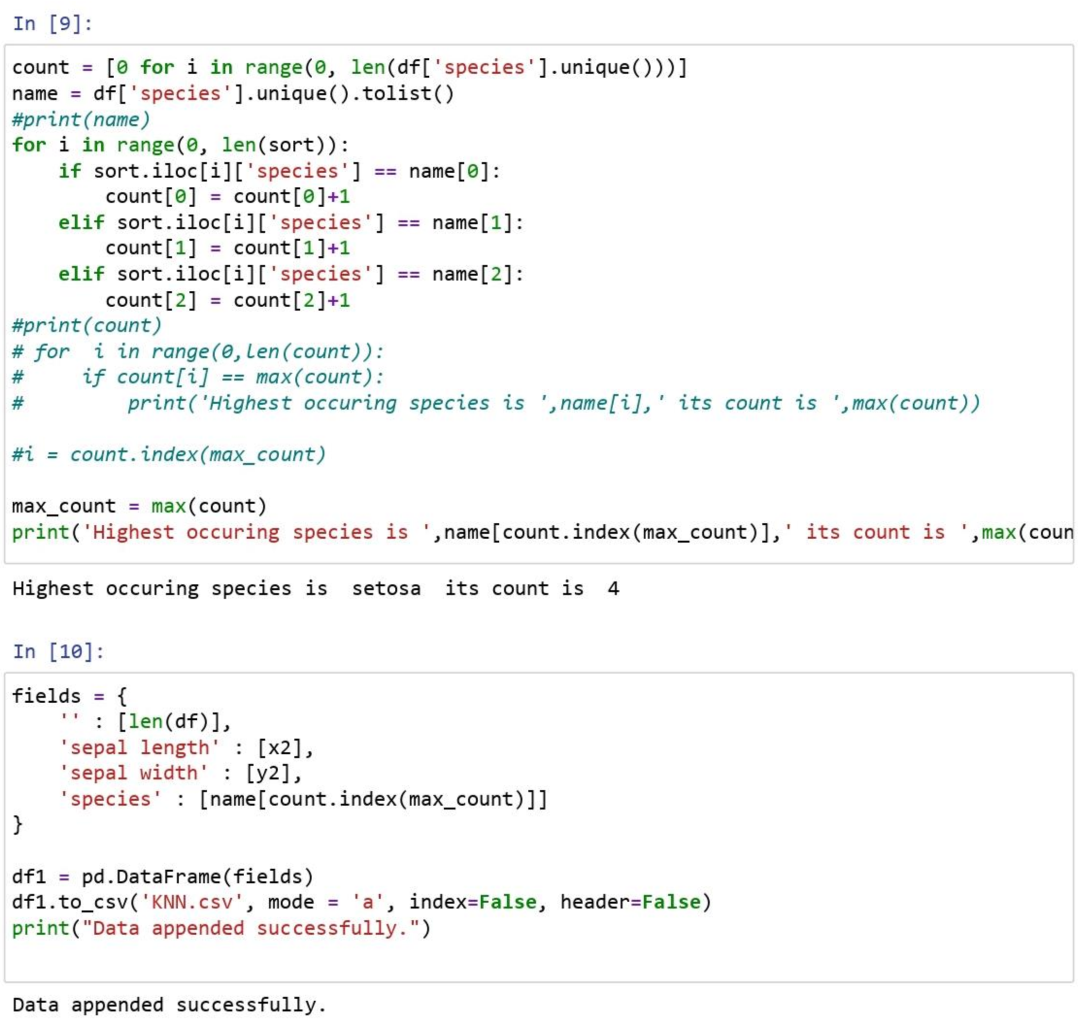
In [8]:

```
sort.iloc[0]
```

Out[8]:

```
sepal length         5.3
sepal width          3.7
species           setosa
sqrt            5.724509
Name: 0, dtype: object
```

Fig 10 KNN algorithm without sklearn

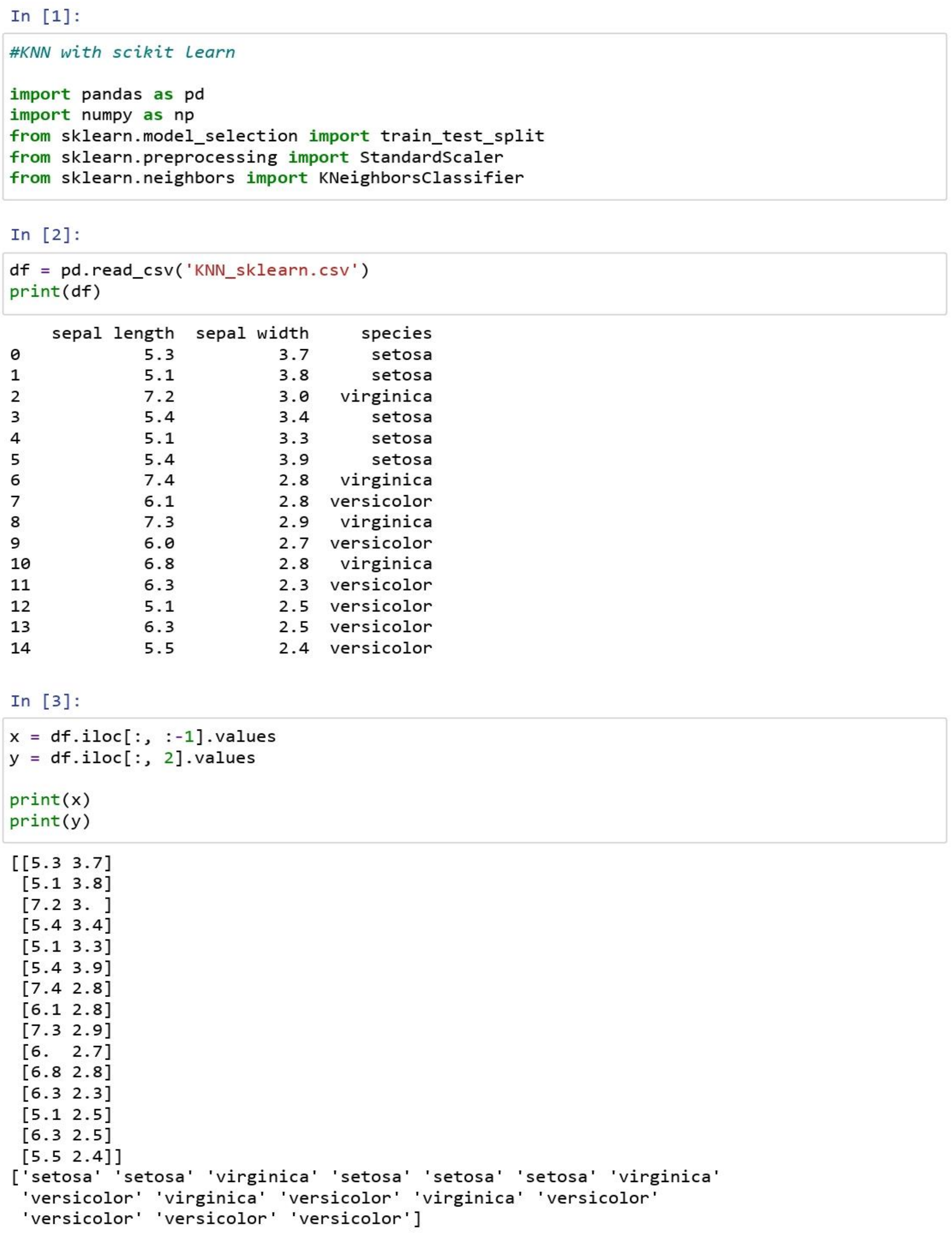


```
In [9]:

count = [0 for i in range(0, len(df['species'].unique()))]
name = df['species'].unique().tolist()
#print(name)
for i in range(0, len(sort)):
    if sort.iloc[i]['species'] == name[0]:
        count[0] = count[0]+1
    elif sort.iloc[i]['species'] == name[1]:
        count[1] = count[1]+1
    elif sort.iloc[i]['species'] == name[2]:
        count[2] = count[2]+1
#print(count)
# for  i in range(0,len(count)):
#     if count[i] == max(count):
#         print('Highest occuring species is ',name[i],' its count is ',max(count))

#i = count.index(max_count)

max_count = max(count)
print('Highest occuring species is ',name[count.index(max_count)],' its count is ',max(coun

Highest occuring species is  setosa  its count is  4

In [10]:

fields = {
    '' : [len(df)],
    'sepal length' : [x2],
    'sepal width' : [y2],
    'species' : [name[count.index(max_count)]]
}

df1 = pd.DataFrame(fields)
df1.to_csv('KNN.csv', mode = 'a', index=False, header=False)
print("Data appended successfully.")

Data appended successfully.
```

Fig 11 KNN algorithm without sklearn

- KNN with sklearn

Code:

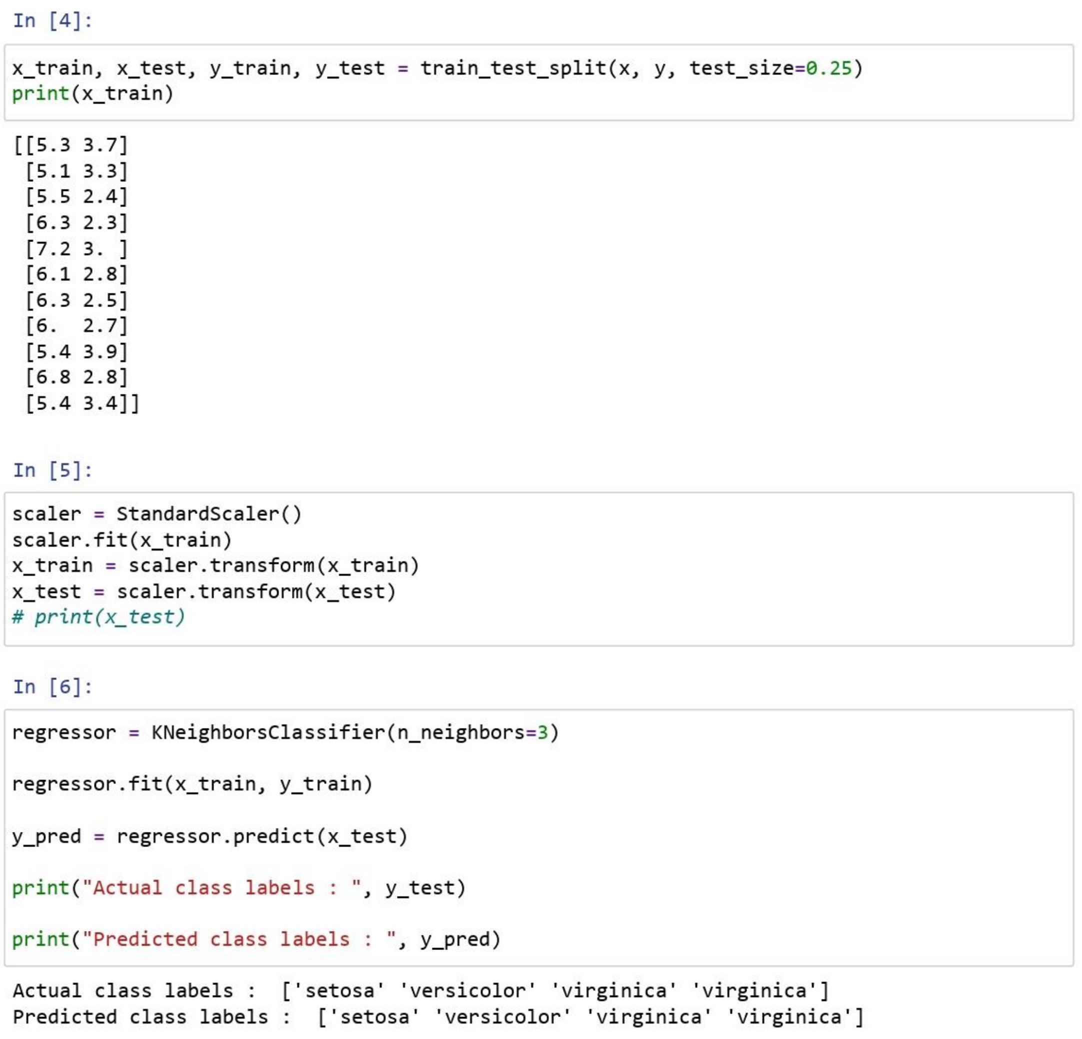


```
In [1]:

#KNN with scikit Learn

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

```
In [2]:

df = pd.read_csv('KNN_sklearn.csv')
print(df)

    sepal length  sepal width     species
0            5.3          3.7      setosa
1            5.1          3.8      setosa
2            7.2          3.0   virginica
3            5.4          3.4      setosa
4            5.1          3.3      setosa
5            5.4          3.9      setosa
6            7.4          2.8   virginica
7            6.1          2.8  versicolor
8            7.3          2.9   virginica
9            6.0          2.7  versicolor
10           6.8          2.8   virginica
11           6.3          2.3  versicolor
12           5.1          2.5  versicolor
13           6.3          2.5  versicolor
14           5.5          2.4  versicolor
```

```
In [3]:

x = df.iloc[:, :-1].values
y = df.iloc[:, 2].values

print(x)
print(y)

[[5.3 3.7]
 [5.1 3.8]
 [7.2 3. ]
 [5.4 3.4]
 [5.1 3.3]
 [5.4 3.9]
 [7.4 2.8]
 [6.1 2.8]
 [7.3 2.9]
 [6.  2.7]
 [6.8 2.8]
 [6.3 2.3]
 [5.1 2.5]
 [6.3 2.5]
 [5.5 2.4]]
['setosa' 'setosa' 'virginica' 'setosa' 'setosa' 'setosa' 'virginica'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor']
```

Fig 12 KNN algorithm with sklearn



```
In [4]:
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
print(x_train)
```

```
[[5.3 3.7]
 [5.1 3.3]
 [5.5 2.4]
 [6.3 2.3]
 [7.2 3. ]
 [6.1 2.8]
 [6.3 2.5]
 [6.  2.7]
 [5.4 3.9]
 [6.8 2.8]
 [5.4 3.4]]
```

```
In [5]:
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
# print(x_test)
```

```
In [6]:
regressor = KNeighborsClassifier(n_neighbors=3)

regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)

print("Actual class labels : ", y_test)

print("Predicted class labels : ", y_pred)
```

```
Actual class labels :  ['setosa' 'versicolor' 'virginica' 'virginica']
Predicted class labels :  ['setosa' 'versicolor' 'virginica' 'virginica']
```

Fig 13 KNN algorithm with sklearn

Download this file and code with dataset from below QR code: