Lecture 9 – KNN Classifier

Mr. Yousif G. Arshak

University of Zakho

Computer Science Department

yousif.arshak@uoz.edu.krd

OUTLINES

- KNN Classifier
  - Introduction
  - Working of KNN Algorithm
  - Example
  - Discover your data
  - Implementation in Python

Introduction

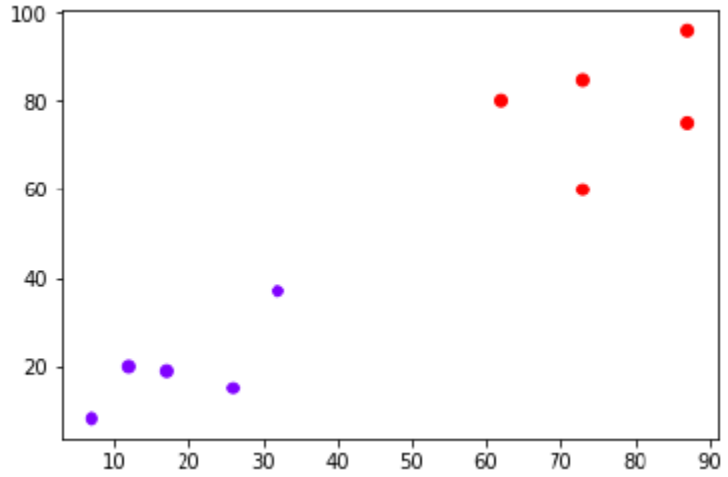
- **K-nearest neighbors (KNN)** algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well:
  - **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
  - **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of KNN Algorithm

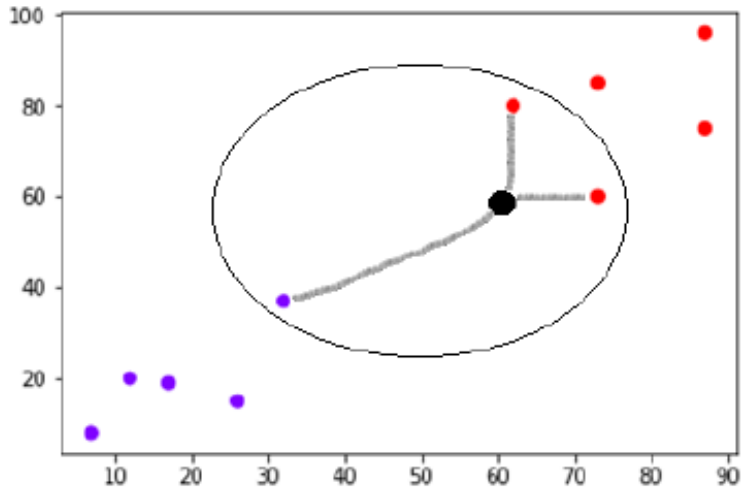
- **Step 1** − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.
- **Step 2** − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.
- **Step 3** − For each point in the test data do the following −
  - **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
  - **3.2** − Now, based on the distance value, sort them in ascending order.
  - **3.3** − Next, it will choose the top K rows from the sorted array.
  - **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.
- **Step 4** − End

Example

- The following is an example to understand the concept of K and working of KNN algorithm −
- Suppose we have a dataset which can be plotted as follows −



- Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

Discover your data

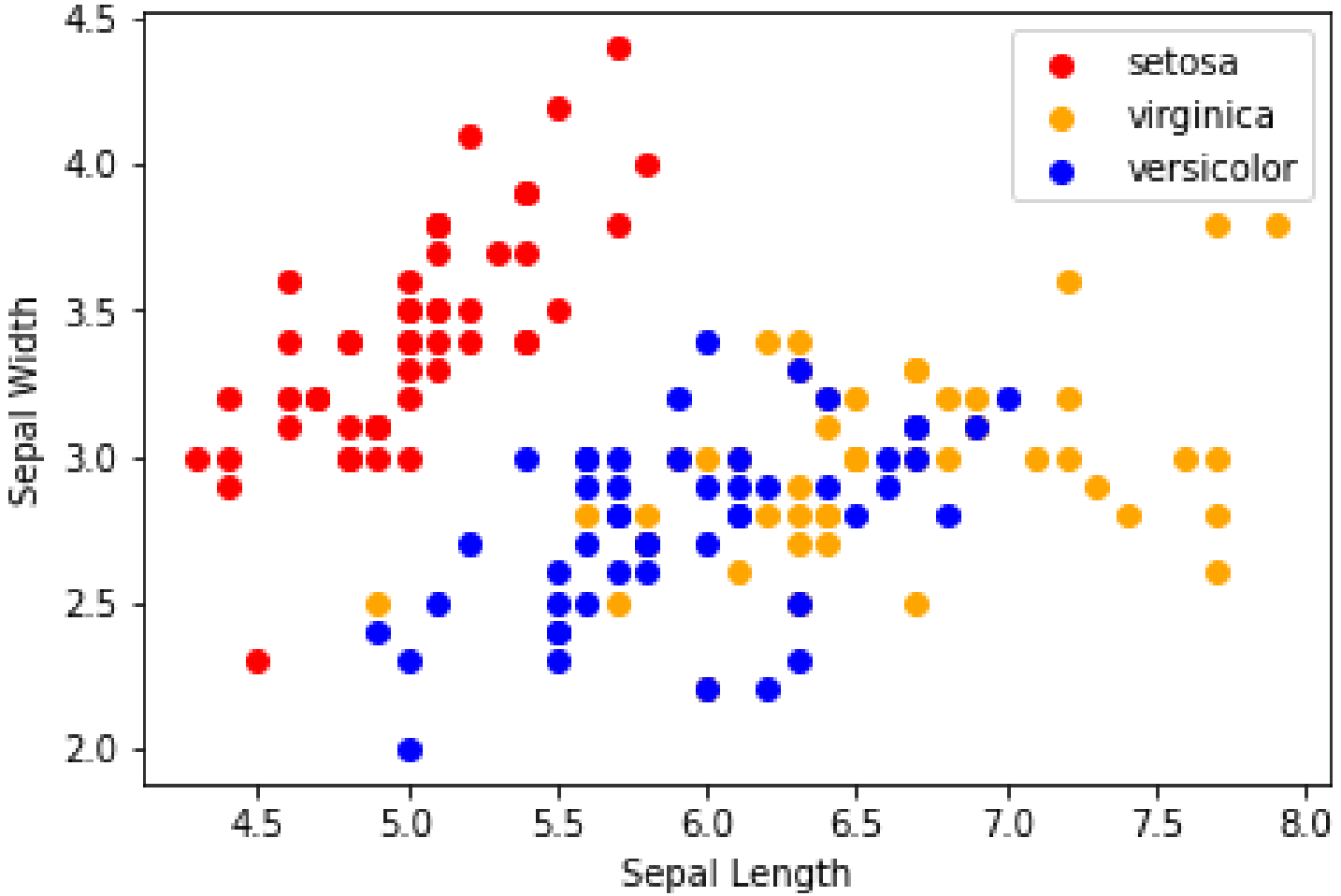
```
#First, start with importing necessary python packages

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# read Iris dataset from csv file from your PC
df = pd.read_csv('iris.csv')
```

Get data from an online server

```
# read from online link
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
headernames = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
ds = pd.read_csv(path, names = headernames)
```

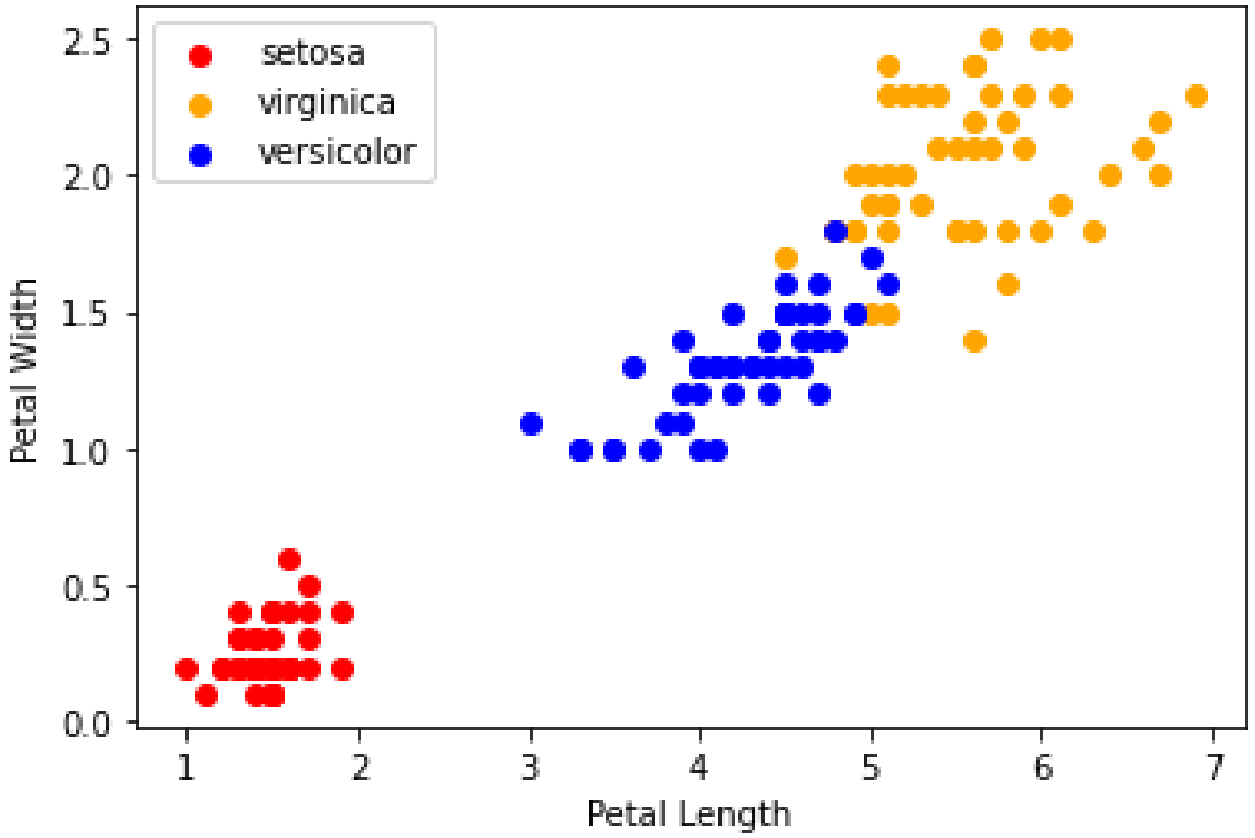
We will use df as our variable

```
#scatter plot for Sepal Length and Width
colors=['red','orange','blue’] # array for colors
species = ['setosa', 'virginica', 'versicolor’] # array for labels
# put data in to scatter plot
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'],x['sepal_width'], c=colors[i], label=species[i])
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.legend()
```


setosa
virginica
versicolor
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Lets use different features to see the difference

```
#scatter plot for Petal Length and Width
colors=['red','orange','blue']
species = ['setosa', 'virginica', 'versicolor']

for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['petal_length'],x['petal_width'], c=colors[i], label=species[i])
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend()
```


setosa
virginica
versicolor
Petal Width
Petal Length
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

Implementation in Python

- As we know K-nearest neighbors (KNN) algorithm can be used for both **classification** as well as **regression**. The following are the recipes in Python to use KNN as **classifier**
- We will use SKLearn Library to implement KNN
- To install SKLearn Library use the bellow command:
  - `pip install -U scikit-learn`

KNN as Classifier

```
from sklearn.datasets import load_iris  # import built-in dataset
from sklearn.neighbors import KNeighborsClassifier # import KNNClassifer
import numpy from sklearn.model_selection # import Model Selection
import train_test_split irisDataset = load_iris() # import train and test data from iris dataset
Data_train, Data_test, Target_train, Target_test = train_test_split(irisDataset["data"], irisDataset["target"],
random_state=0) # divide your data samples into train and test
data_to_predict = numpy.array([[5, 2.9, 1, 0.2]]) # a test sample
knn = KNeighborsClassifier(n_neighbors=7) # set a number for K _ Neighbors
knn.fit(Data_train, Target_train) # train your model
predicted_species = knn.predict(data_to_predict) # test your model with a random data to see the accuracy
print("The most probable species the sample will be is " +
irisDataset["target_names"][predicted_species][0]
            + ". " + "The machine was tested and got {:.2f}% of its predictions
right.".format(knn.score(Data_test, Target_test) * 100))
```

Also try changing the *n_neighbours* parameter values to 19, 25, 31, 43 etc. What happens to the accuracy then?