**UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

**Instructor: PhD Lê Văn Vinh**

**Project 03**

**PROJECT REPORT**

**K-Nearest Neighbor Algorithm**

**Nguyễn Mạnh Tiến – 17110093**

**Nguyễn Hoàng Trường Minh – 17110067**

**SCHOOL YEAR: 2020 – 2021**

**HO CHI MINH CITY – December 2020**

Instructor's evaluation

*Ho Chi Minh City, January, 2021*

**Instructor**

Reviewer's evaluation

*Ho Chi Minh City, January, 2021*

**Reviewer**

Thank note

Table of Contents

[Instructor's evaluation 2](#_Toc60342642)

[Reviewer's evaluation 3](#_Toc60342643)

[Thank note 4](#_Toc60342644)

[1. Introduction 6](#_Toc60342645)

[1.1. KNN Theory 6](#_Toc60342646)

[1.1.1. Type of algorithm 6](#_Toc60342647)

[1.1.2. Distance measure 6](#_Toc60342648)

[1.1.3. Algorithm steps 6](#_Toc60342649)

[1.1.4. Advantages 7](#_Toc60342650)

[1.1.6. Disadvantages: 7](#_Toc60342651)

[1.1.7. Example with Python 7](#_Toc60342652)

[1.1.8. Evaluation method 9](#_Toc60342653)

# 1. Introduction

## 1.1. KNN Theory

### 1.1.1. Type of algorithm

KNN can be used for both classification and regression predictive problems. KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consisting of training observations (x, y) and would like to capture the relationship between x and y. More formally, our goal is to learn a function h:X→Y so that given an unseen observation x, h(x) can confidently predict the corresponding output y.

### 1.1.2. Distance measure

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen” observation. Similarity is defined according to a distance metric between two data points. The k-nearest-neighbor classifier is commonly based on the Euclidean distance between a test sample and the specified training samples. Let xi be an input sample with p features (xi1, xi2, … xip), n be the total number of input samples (i=1,2, ... n). The Euclidean distance between sample xi and xl is defined as:



Sometimes other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance.

### 1.1.3. Algorithm steps

1. Choose the number K of neighbors
2. Take the K nearest neighbors of the new data point, according to your distance metric
3. Among these K neighbors, count the number of data points to each category
4. Assign the new data point to the category where you counted the most neighbors

### 1.1.4. Advantages

1. The complicated of the algorithm of the training process is 0.
2. It’s very easy to predict the label of the new data test.
3. It’s unnecessary to assume about the distribution of each class.

### 1.1.6. Disadvantages:

1. It will not actual correctly if K is very small.
2. It will take many times to predict the label of the data test if the data set is very large.
3. The bigger K, the larger complexity.

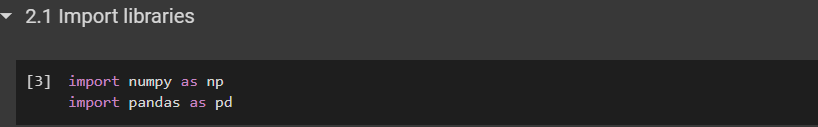
### 1.1.7. Preprocessing with Python

Iris flower dataset is a small data set. This data set consist of information of 3 kind of Iris flowers: Iris-sesota, Iris-virginia and Iris-versicolor. Each kind has 50 flowers with 4 information about: length and width of sepal, length and width of petal. I will put the illustrate picture of 3 kind of Iris flowers below.

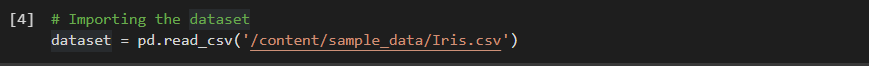


Experiment and coding:

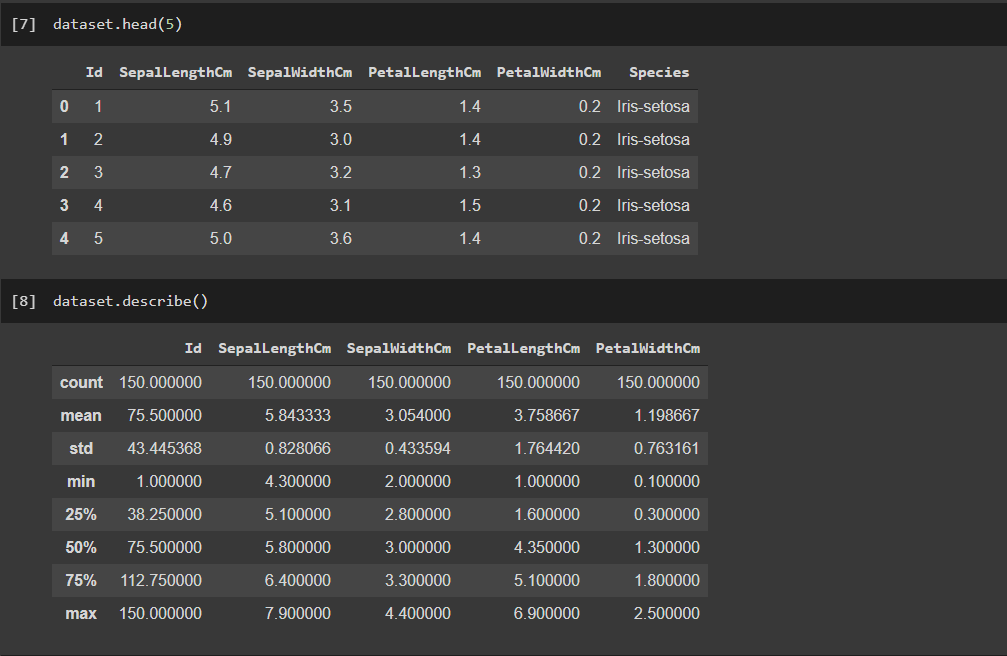
First of all, we have to import some library of Scikit-learn.

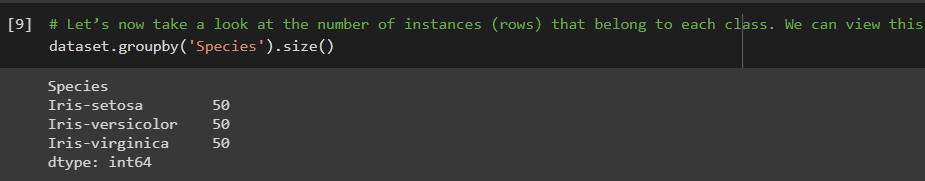


Next, we will load data from csv file

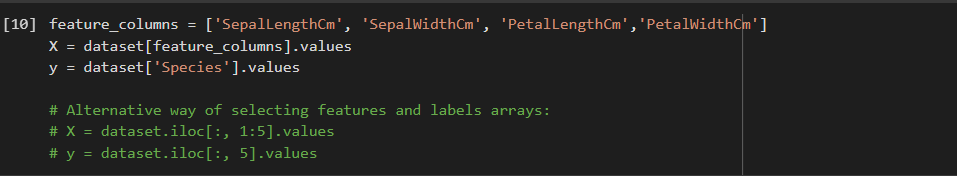


View first 5 rows to verify that data loaded correctly. Then describe the data with python.

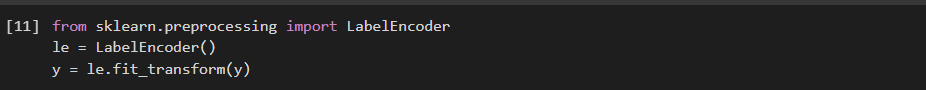




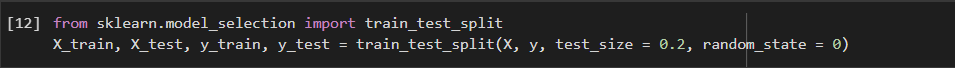
As we can see dataset contain six columns: Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm and Species. The actual features are described by columns 1-4. Last column contains labels of samples. Firstly we need to split data into two arrays: X (features) and y (labels).



As we can see labels are categorical. KNeighborsClassifier does not accept string labels. We need to use LabelEncoder to transform them into numbers. Iris-setosa correspond to 0, Iris-versicolor correspond to 1 and Iris-virginica correspond to 2.



Next step, we have to divide the data set into training set and test set. In this case, I will use 20% data for the test set and 80% data for the training set. In the Scikit-learn library, we already have some functions to help us divide the data set randomly.

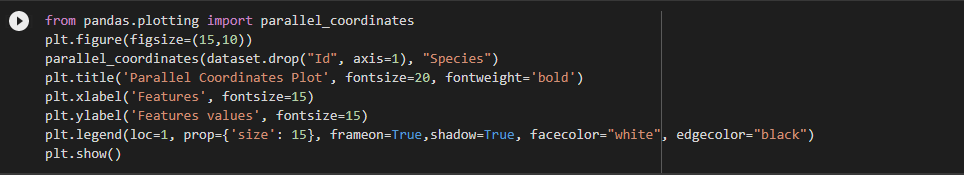


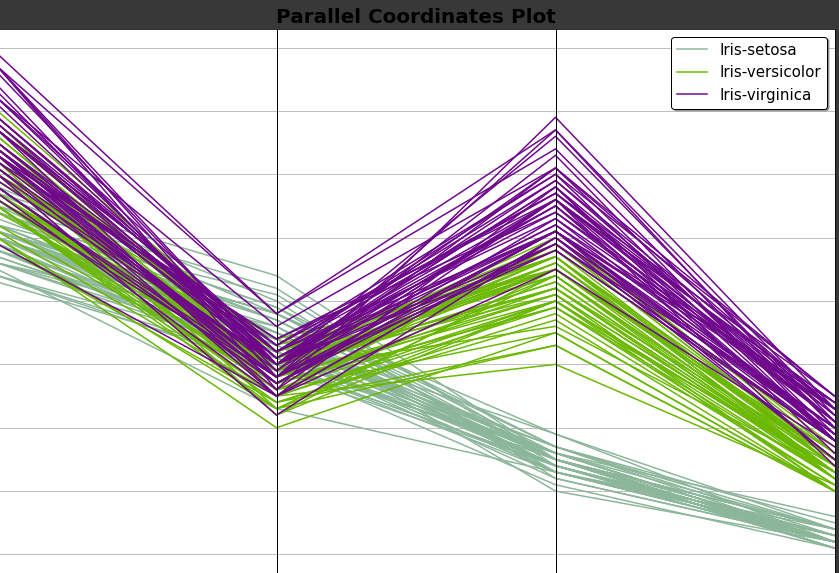
So, we’ll have training size: 120, test size: 30

### 1.1.8. Data visualization

After processing data for training and testing, we’ll try to visualize dataset using python libraries

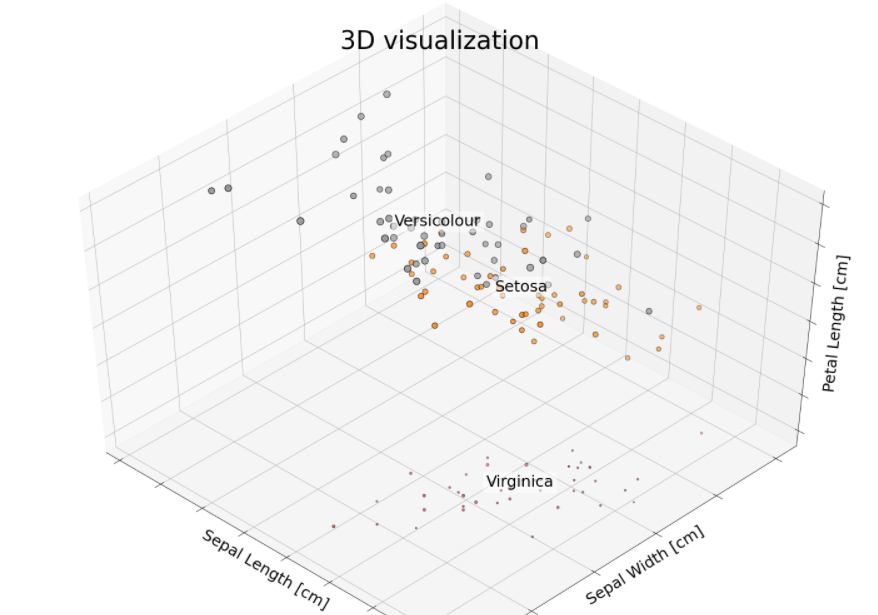
Using parallel coordinates





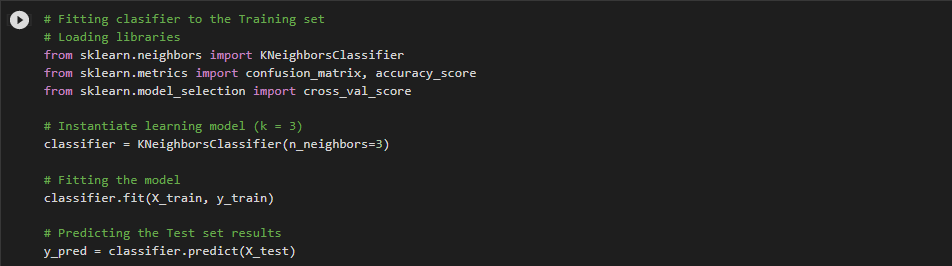
And 3D visualization



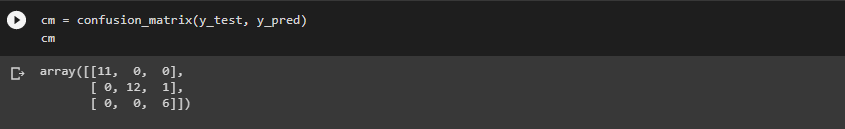


### 1.1.9. Implement KNN algorithm and evaluation method

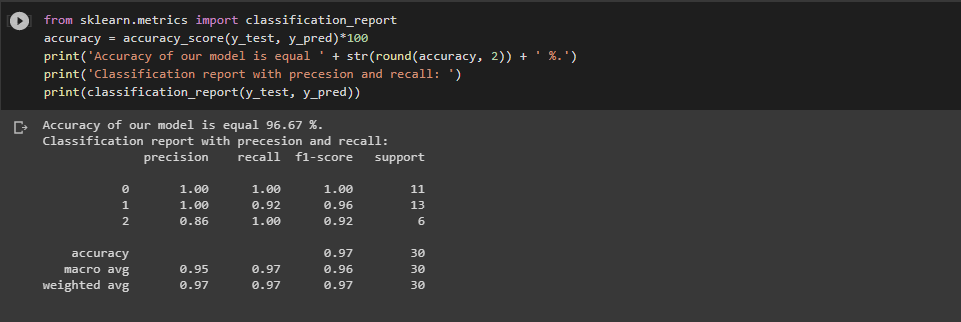
Fit data to the algorithm and make predictions



After applying the algorithm, we’ll have the confusion matrix like:



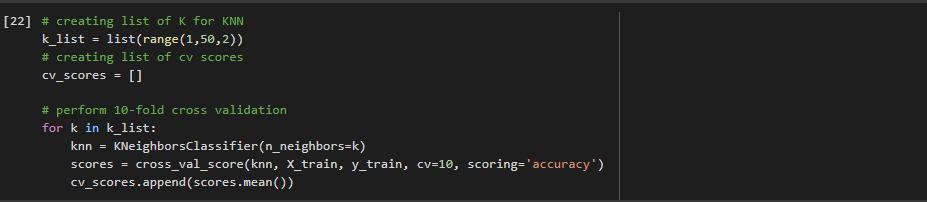
View fully report after predictions:

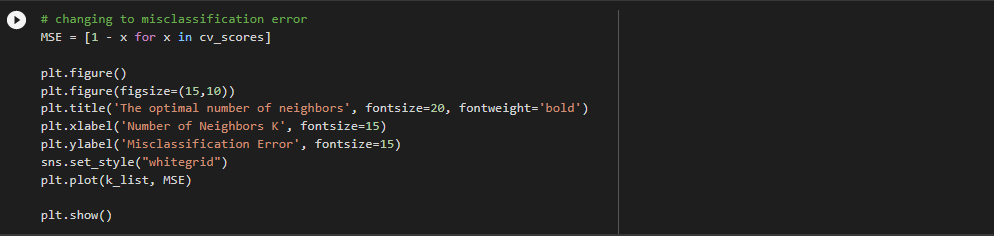


So, we have accuracy about 97% and recall, precision rates for each kind of Iris flower.

### 1.1.10. Finding the best K for the algorithm to acquire the highest accuracy rate

Create a list of K from 1 -> 50 and compare the accuracy for each K number





Getting the best K which causes the minimum MSE (MisClassificationError) from the list

