UNIVERSITI SAINS MALAYSIA

A Comparative Study Among KNN and Decision Tree Models on Iris Flower Dataset

Presented by: Group no.7

Merna Mohamed Elalfy
Islam Said Elshafie
Aya Khaled Salah Sadek
Ahmed Samir Ziada
Yasmin Samy Abdullah Elsaeed

Presented to: Ts. Dr. Khaw Khai Wah

Data programming and predictive analytics for Business

“Second Project”

Table of Contents

1. Introduction

Machine learning is a subdivision of Artificial Intelligence. It is used in many application areas in many fields such as business, medicine, sports, geography, environment, healthcare, traffic predictions, Education, Entertainment, and online shopping. as it provides them with solutions for the issue of having large amount of data and the need of selecting the important information from the raw data from the large sets of knowledge and the inception of databases in large number. Machine learning working process is to train the data and the algorithm develops some rules, based on that learning, evaluation is done with the test data to generate results without human intervention. This study is a comparative study focuses on comparing between two machine learning models "KNN (K-Nearest Neighbors), Decision Tree" by applying them on Iris flower dataset.

1.1 KNN Algorithm

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN 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 classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems [1].

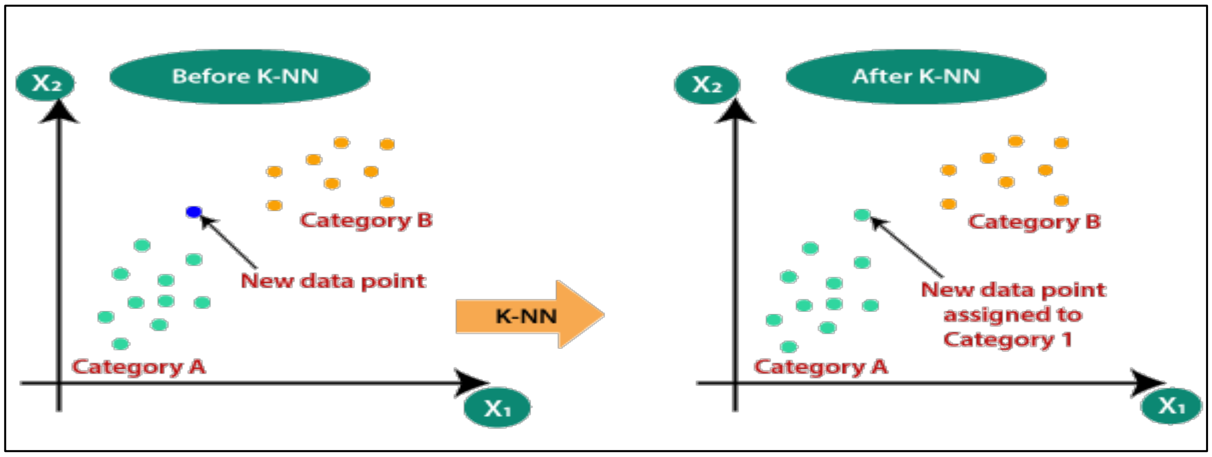


Figure (1): KNN Algorithm Illustration

1.2. Decision Tree Algorithm

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches [2].

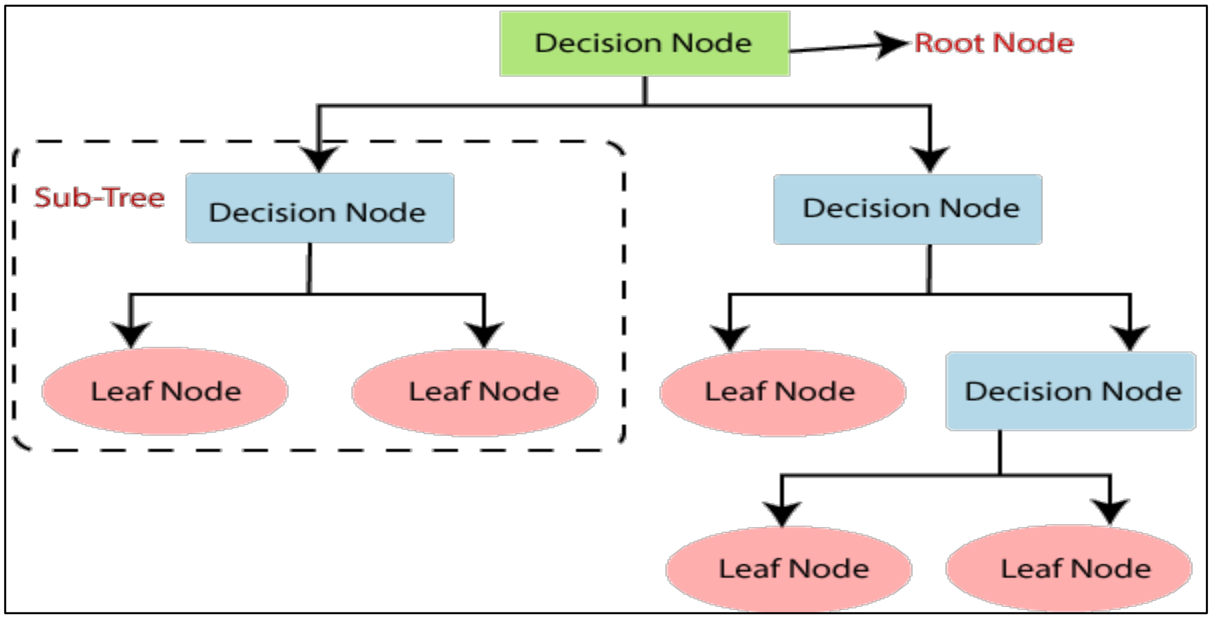


Figure (2): Decision Tree Algorithm Illustration

2. Objectives

The Iris flower dataset is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper [3]. The dataset contains a set of 150 records. Each record has one of three species of iris (Iris Setosa, Iris virginica, and Iris versicolor) which is called the **target variable**. Four attributes were measured for each record, the length and the width of the sepals and petals in centimeters. These attributes are called **feature variables** and used to determine the species of the iris. The objective of the experiments is to use two different supervised machine learning algorithms (KNN and Decision Tree) on the iris dataset to create two machine learning models [4]. We will use the models to predict the species of the iris flower based on the features (Petal Length, Petal Width, Sepal Length, Sepal width). Then we will measure the accuracy of each model to choose the best machine learning algorithm.

3. Justification of choice these 2 algorithms

To justify the used of the two algorithms, first we need to understand our data well. Iris flower dataset consist of only 150 samples. this is a minimal data collection. It has 4 numerical features (Sepal width, Sepal length, Petal width, and Petal length) and 1 target, the flower species which is either (Versicolor, Setosa, and Virginica).

KNN is one of the simplest forms of machine learning algorithms mostly used for classification [5]. Since iris flower is a labeled dataset and data is a noise free [6]. KNN will be a good choice especially it is used with small datasets [5]. KNN is also called a lazy leaner because there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm faster than other algorithms that require training e.g. SVM, Linear Regression etc. [7]. KNN is also non-parametric which means that it doesn't make assumptions regarding the dataset. So even If you don't know too much about the dataset initially this feature can be a lifesaver [8].

The second choice is the Decision Tree. Decision Tree algorithm is considered to be the easiest and popular classification algorithms to understand, interpret and visualize [9]. It is simple to understand because it follows the same process which a human follow while making any decision in real-life [10]. The logic behind the Decision Tree is often simply visualize as a result of it shows a tree-like structure [2] When using Decision Tree, it usually not necessary to normalize the data or scaling it as well [11]. It also a non-parametric algorithm like KNN which means that it doesn't make assumptions regarding the dataset.

4. Steps to build the machine learning models

4.1. KNN Model

There are four main steps to build machine learning models:

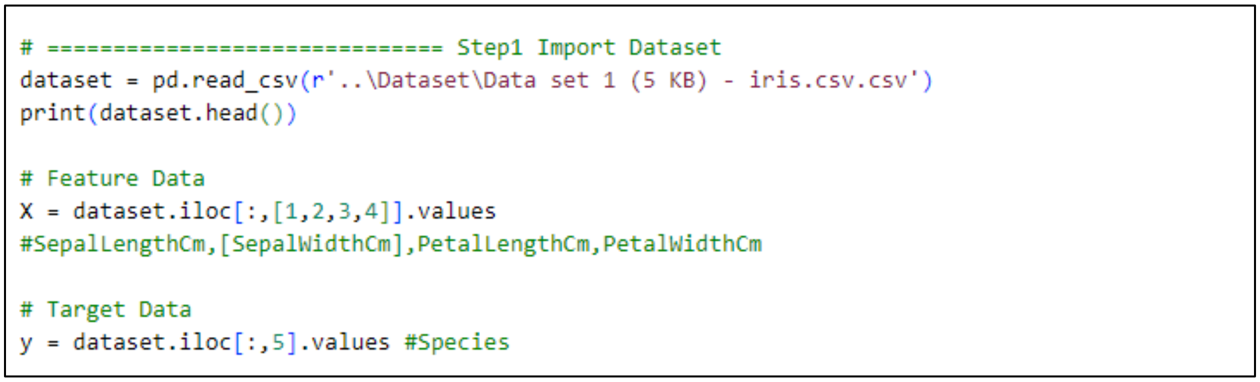
Step1: Import Dataset

First, we import **numpy** as np, numpy is python library that used for working with arrays [12].

```
import numpy as np
```

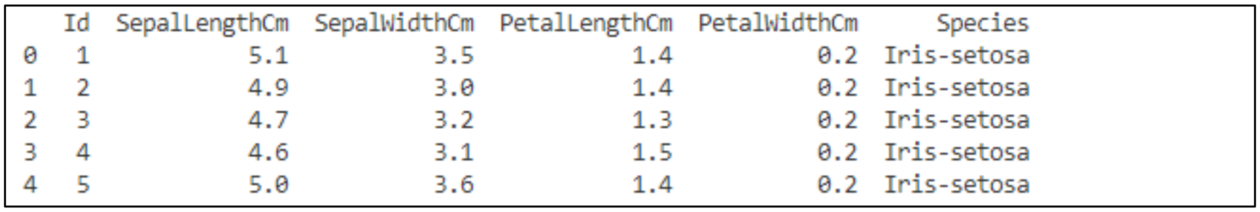
Also, we import **pandas** as pd, pandas is python library that used to analyze data [13].

```
import pandas as pd
```



```
# ============================== Step1 Import Dataset
dataset = pd.read_csv(r'..\Dataset\Data set 1 (5 KB) - iris.csv.csv')
print(dataset.head())

# Feature Data
X = dataset.iloc[:,[1,2,3,4]].values
#SepalLengthCm,[SepalWidthCm],PetalLengthCm,PetalWidthCm

# Target Data
y = dataset.iloc[:,5].values #Species
```

We read the iris dataset using **pd.read_csv(FilePath)** and store in a pandas dataframe [14] variable called **dataset**. We print the first 5 records to explore our data.



```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

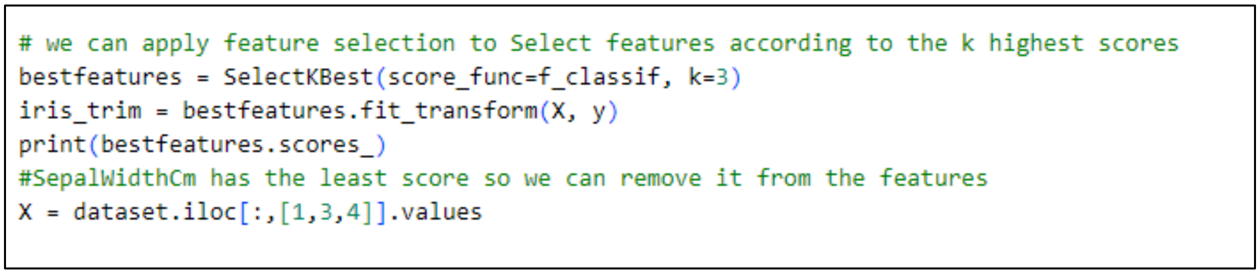
We divide the dataset into two main parts in **numpy arrays**: Feature Data called **X**, that contain the features of the dataset [SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm]. and Target Data called **y**, that is our target Iris species.

Step2: Data Splitting

Feature selection can be applied before splitting the dataset, Features selection techniques are either used to exclude irrelevant features, or to exclude redundant ones and sometimes they are used to do both of them [15].

We import **SelectKBest** and **f_classif** from **sklearn.feature_selection** to apply feature selection for the best 3 features effect the target [16].

```
from sklearn.feature_selection import SelectKBest, f_classif
```



```
# we can apply feature selection to Select features according to the k highest scores
bestfeatures = SelectKBest(score_func=f_classif, k=3)
iris_trim = bestfeatures.fit_transform(X, y)
print(bestfeatures.scores_)
#SepalWidthCm has the least score so we can remove it from the features
X = dataset.iloc[:,[1,3,4]].values
```

We could find that the second feature has the least score which is SepalWidth so we can remove it from the features

```
[ 119.26450218   47.3644614 1179.0343277   959.32440573]
```

Note that: we can see that 3-feature dataset performs better or at least equally good as a 4-feature dataset [17].

Splitting the dataset into train and test sets is one of the important parts of data pre-processing, as by doing so, we can improve the performance of our model and hence give better predictability. If we train and test the model with two different datasets, then it will decrease the performance of the model. Hence it is important to split a dataset into two parts, i.e., train and test set [18].

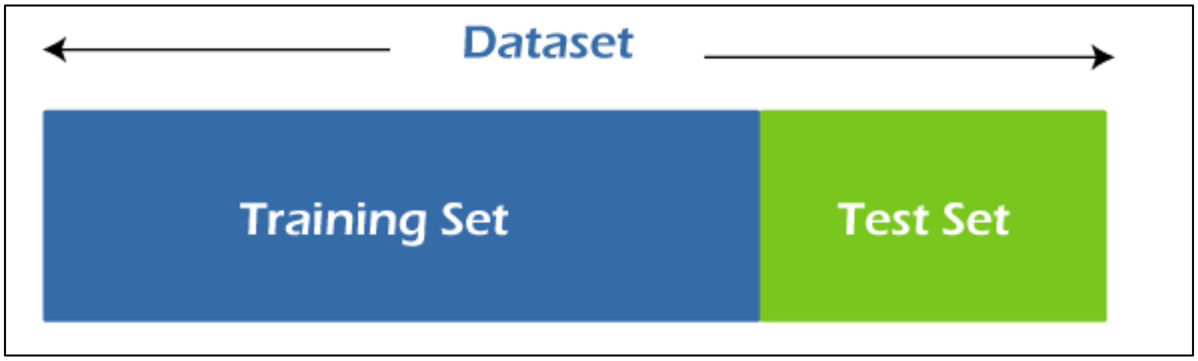


Figure (3): Dataset Train/Test

We import the **train_test_split function** from **sklearn.model_selection** that is used to Split arrays or matrices into random train and test subsets [19]

```
from sklearn.model_selection import train_test_split
```

We split both the feature data **X** and target data **y** into two **numpy arrays: X_train** that we will use to train(fit) our model and **X_test** that we will use to test(predict) the result of the model after training. **y_train** that will used also in train(fit) our model and compare the prediction result with **y_test**.

```
# Spliting dataset into features traning set(X_train , X_test) and
# target test set(y_train , y_test)
X_train, X_test , y_train , y_test = train_test_split(X ,y , test_size = 0.25 ,
random_state = 42)
```

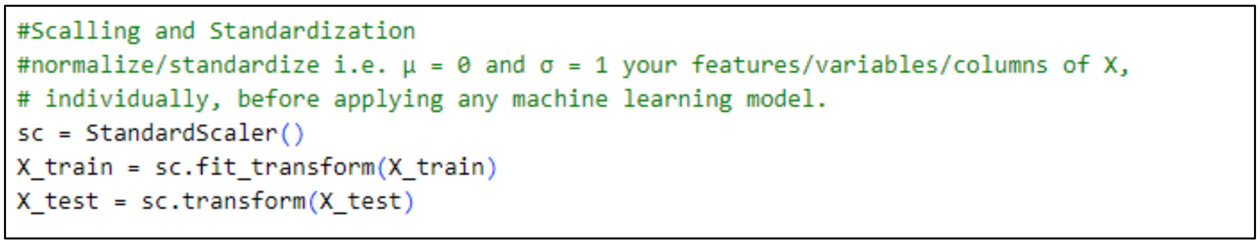
We assign the **test_size** parameter to become 25% of the dataset and set the seed of the random generator **random_state** to 42 [20].

Note that: if the test_size is set to None, it will be set to 0.25. [19]. Also, the most used value by developers for the random_state is 42 [18].

We import the **StandardScaler** from **sklearn.preprocessing** that is used to Standardize features by removing the mean and scaling to unit variance [21]

```
from sklearn.preprocessing import StandardScaler
```

We standardize **X_train** and **X_test**. In fact, it is a common requirement for many machine learning estimators: they might behave badly if the individual feature do not more or less look like standard normally distributed data [22].



```
#Scalling and Standardization
#normalize/standardize i.e. μ = 0 and σ = 1 your features/variables/columns of X,
# individually, before applying any machine learning model.
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step3: Train the Model

We import the **KNeighborsClassifier** from **sklearn.neighbors** , this Classifier implementing the k-nearest neighbors vote [23].

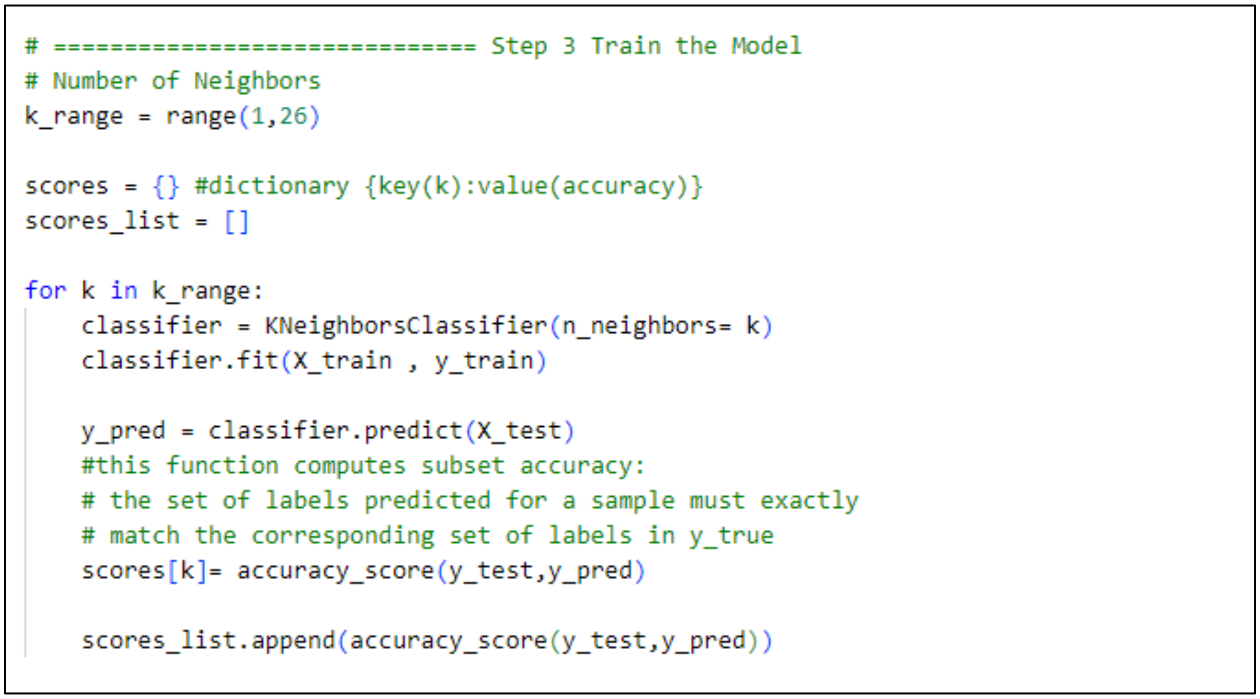
```
from sklearn.neighbors import KNeighborsClassifier
```

First, we need to choose the best k neighbors to guarantee the highest accuracy. So, we define **k_range** with a range from 1-26. For every value in this range.
We will create the **KNeighborsClassifier** and specify the k value to the **n_neighbours** parameter. Then we will train(fit) our model using the **X_train** and **y_train** variables. We define a variable called **y_pred** that store the prediction result when we input the **X_test** to the model.

We import **accuracy_score function** from **sklearn.metrics** to compute the accuracy classification score [24]

```
from sklearn.metrics import accuracy_score
```

This function compares the **y_test** (the actual labels of the iris dataset) with **y_pred** (the predicted labels after we train the model and input the **X_test** features to the model). For every k value we append the accuracy score to list called **scores_list.**

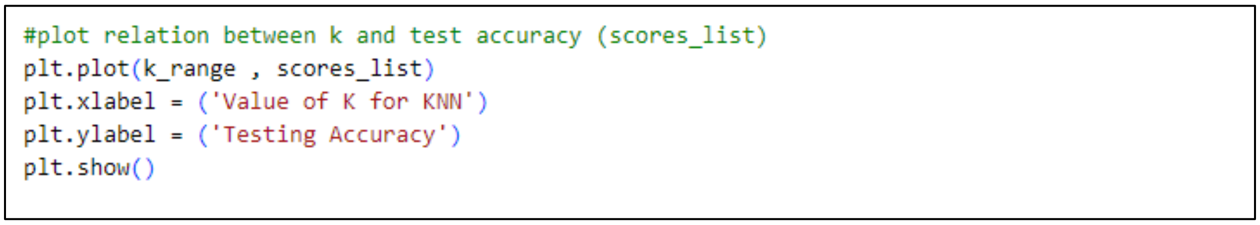


```
# ============================== Step 3 Train the Model
# Number of Neighbors
k_range = range(1,26)

scores = {} #dictionary {key(k):value(accuracy)}
scores_list = []

for k in k_range:
    classifier = KNeighborsClassifier(n_neighbors= k)
    classifier.fit(X_train , y_train)

    y_pred = classifier.predict(X_test)
    #this function computes subset accuracy:
    # the set of labels predicted for a sample must exactly
    # match the corresponding set of labels in y_true
    scores[k]= accuracy_score(y_test,y_pred)

    scores_list.append(accuracy_score(y_test,y_pred))
```

We import **matplotlib.pyplot** as **plt** , this interface allow to create a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. [25]

```
import matplotlib.pyplot as plt
```

We plot the relation between **k_range** and **scores_list** using matplotlib where the k values on the x-axis and accuracy score on the y-axis



```
#plot relation between k and test accuracy (scores_list)
plt.plot(k_range , scores_list)
plt.xlabel = ('Value of K for KNN')
plt.ylabel = ('Testing Accuracy')
plt.show()
```

We will note that there are several values of k neighbors that will output accuracy 1.00. so, for example we choose k=8 for the highest accuracy.

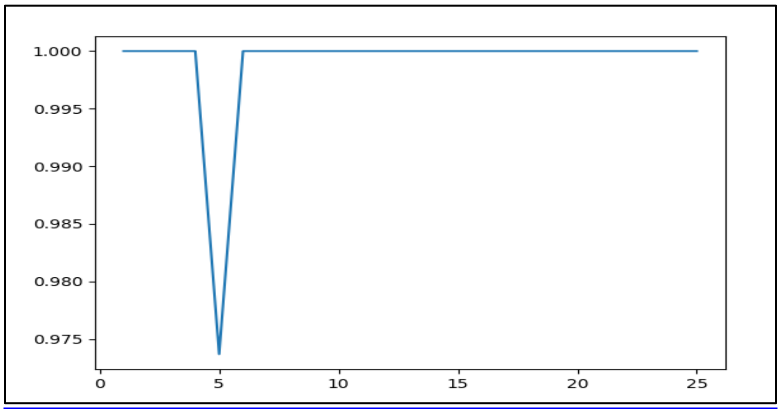
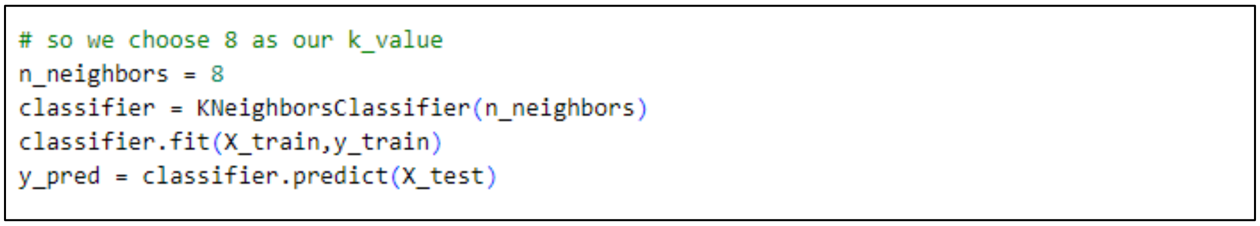


Figure (4): Relation between k on x-axis and accuracy score on y-axis

Same as done before, we will create the model with our **n_neighbors** = 8, train(fit) the model with **X_train** and **y_train**. Then we will predict the iris species using **X_test** and store the result in **y_pred**

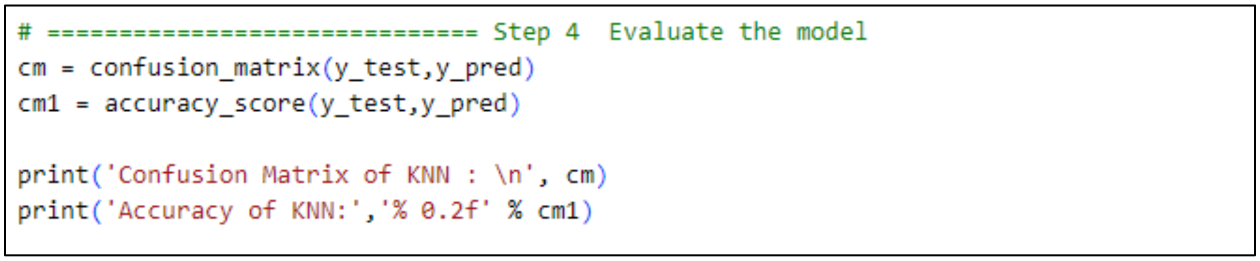


```
# so we choose 8 as our k_value
n_neighbors = 8
classifier = KNeighborsClassifier(n_neighbors)
classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test)
```

Step4: Evaluate the Model

We will evaluate the model using confusion matrix and accuracy score. We already use accuracy score in previous step to compute the difference between the actual target labels and the predicted labels. So, we will import **confusion_matrix** from **sklearn.metrics** to evaluate the accuracy of the classification [26].

```
from sklearn.metrics import confusion_matrix
```



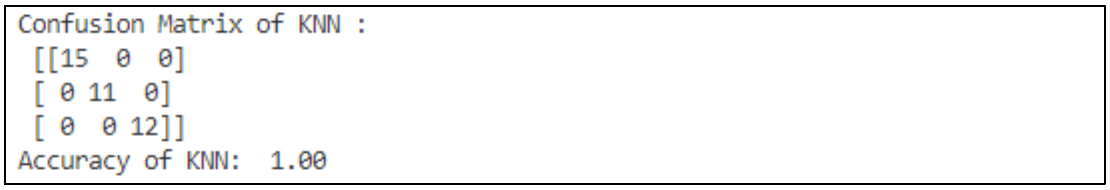
```
# ============================== Step 4  Evaluate the model
cm = confusion_matrix(y_test,y_pred)
cm1 = accuracy_score(y_test,y_pred)

print('Confusion Matrix of KNN : \n', cm)
print('Accuracy of KNN:','% 0.2f' % cm1)
```

In the confusion matrix row indicates the actual values of data and columns indicate the predicted data [27] so:

There are 15 iris flowers predicted as Iris-setosa and they were actually Iris-setosa.

There are 11 iris flowers predicted as Iris-versicolor and they were actually Iris-versicolor.

There are 12 iris flowers predicted as Iris-virginica and they were actually Iris-virginica.



```
Confusion Matrix of KNN :
 [[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
Accuracy of KNN:  1.00
```

The accuracy score is 1.00 which means that all the predicted species by the model are the same as the y_test labels

Step5: Plot the Model

We plot the model for more clarify. First, we represent the model using the last 2 features (highest features from feature selection) and convert the species values to numerical [28].

We import **LitstedColormap** from **matplotlib.colrs** , which is a Colormap object generated from a list of colors [29].

```
from matplotlib.colors import ListedColormap
```

We plot the decision boundary. for that, we will assign a color to each point in the mesh and put the result into a color plot

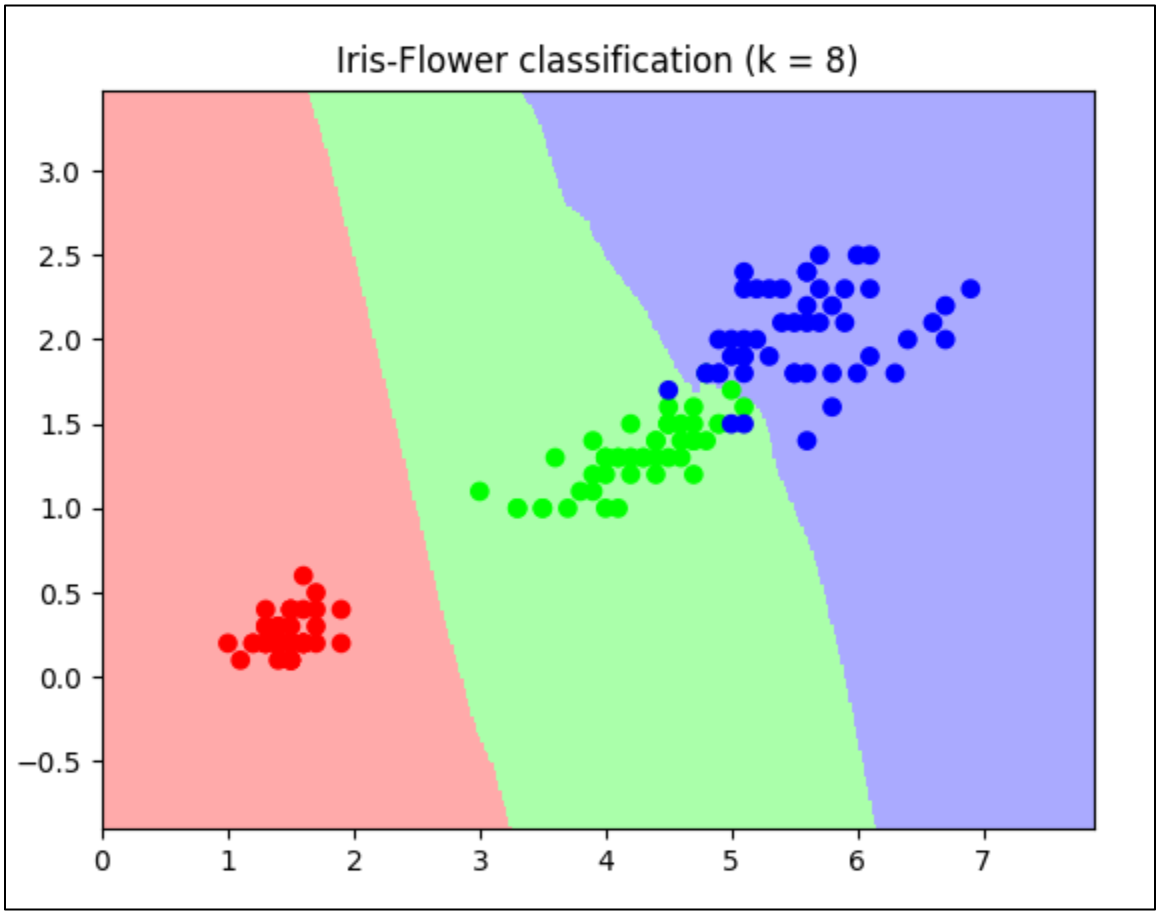
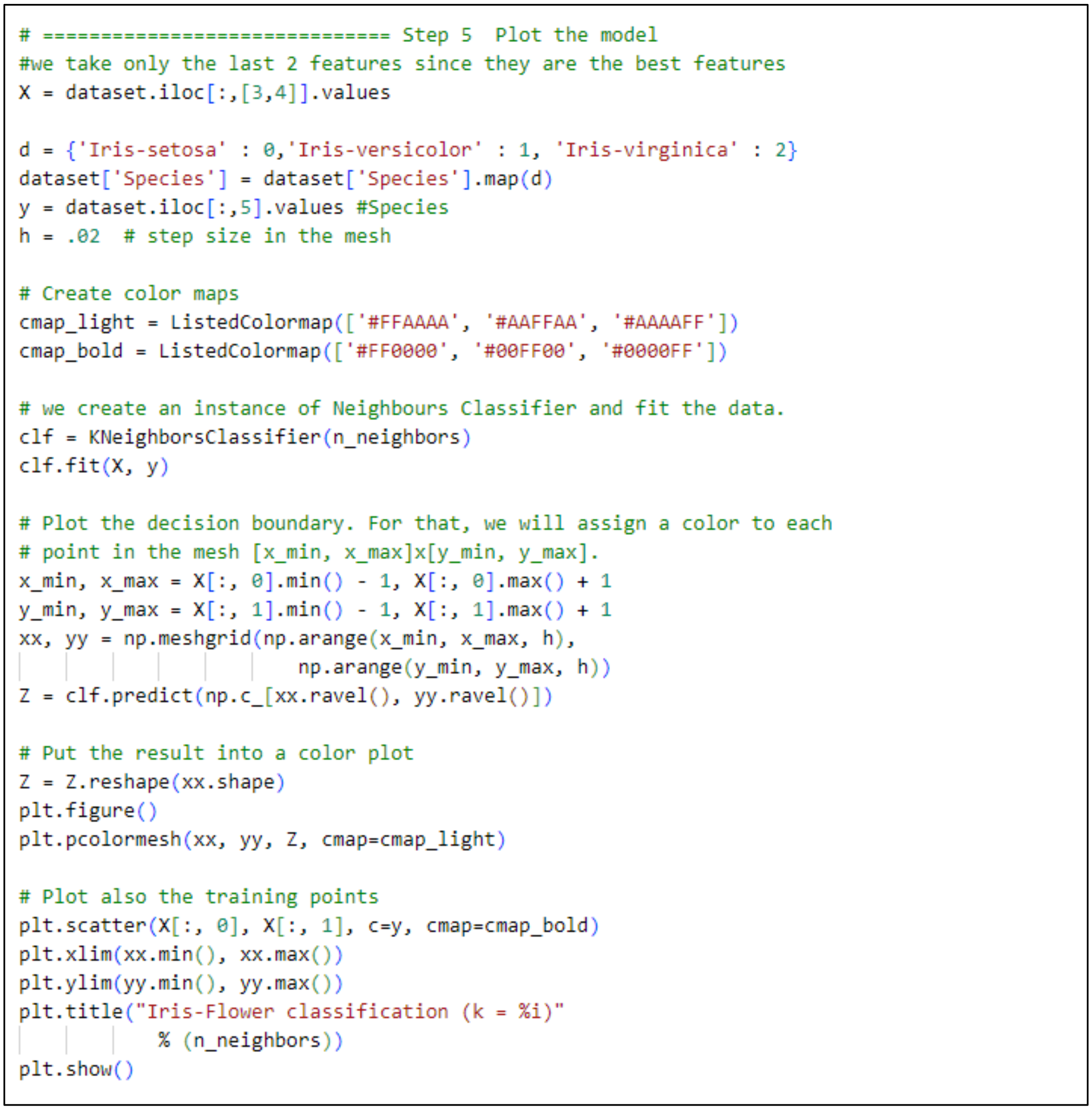


Figure (5): Plot KNN using Petal Length and Petal Width



```
# ============================== Step 5  Plot the model
#we take only the last 2 features since they are the best features
X = dataset.iloc[:,[3,4]].values

d = {'Iris-setosa' : 0,'Iris-versicolor' : 1, 'Iris-virginica' : 2}
dataset['Species'] = dataset['Species'].map(d)
y = dataset.iloc[:,5].values #Species
h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors)
clf.fit(X, y)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                        np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Iris-Flower classification (k = %i)"
            % (n_neighbors))
plt.show()
```

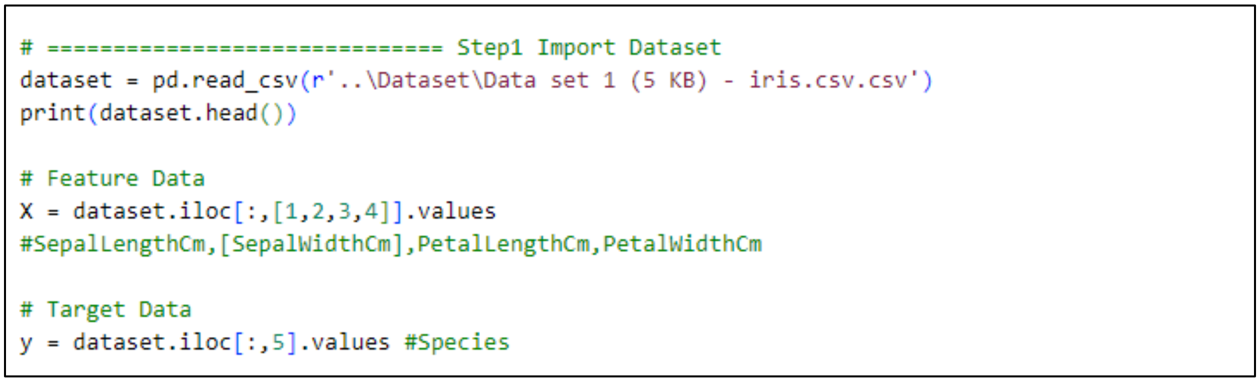
4.2. Decision Tree Model

Using the same steps to build machine learning models as the previous

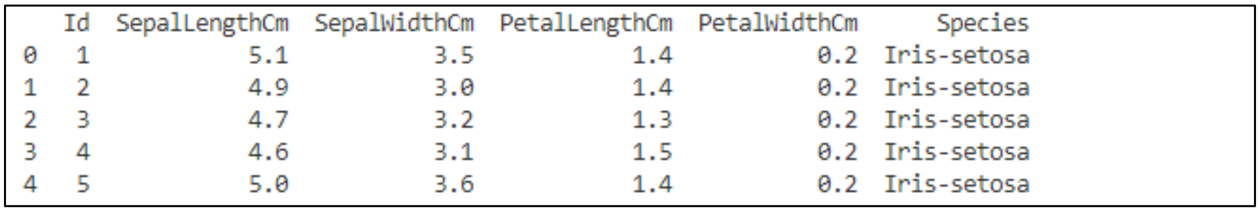
There are four main steps to build machine learning models:

Step1: Import Dataset

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



```
# ============================== Step1 Import Dataset
dataset = pd.read_csv(r'..\Dataset\Data set 1 (5 KB) - iris.csv.csv')
print(dataset.head())

# Feature Data
X = dataset.iloc[:,[1,2,3,4]].values
#SepalLengthCm,[SepalWidthCm],PetalLengthCm,PetalWidthCm

# Target Data
y = dataset.iloc[:,5].values #Species
```

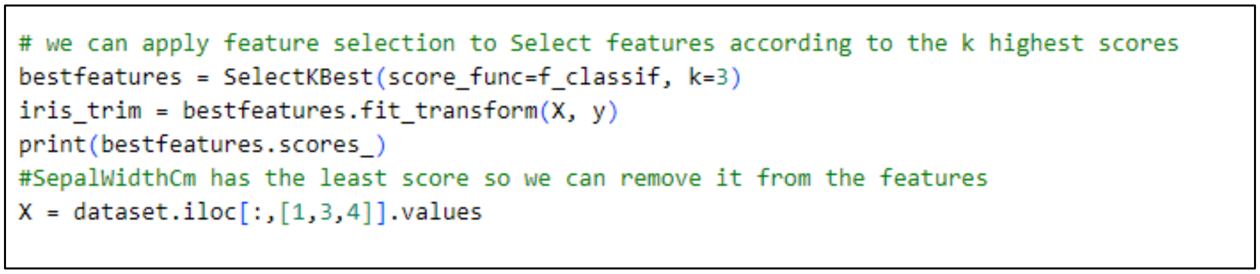


```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

Step2: Data Splitting

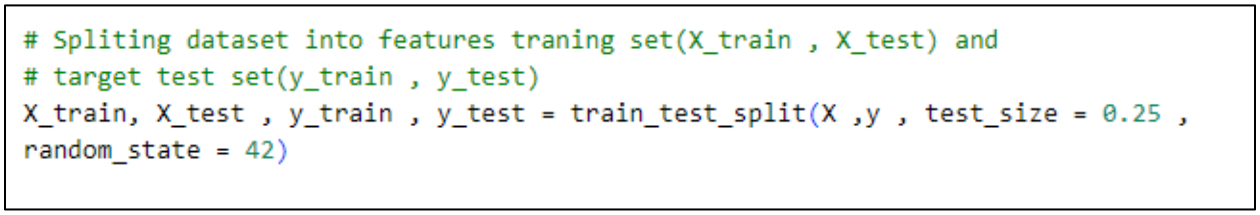
Feature selection

```
from sklearn.feature_selection import SelectKBest, f_classif
```



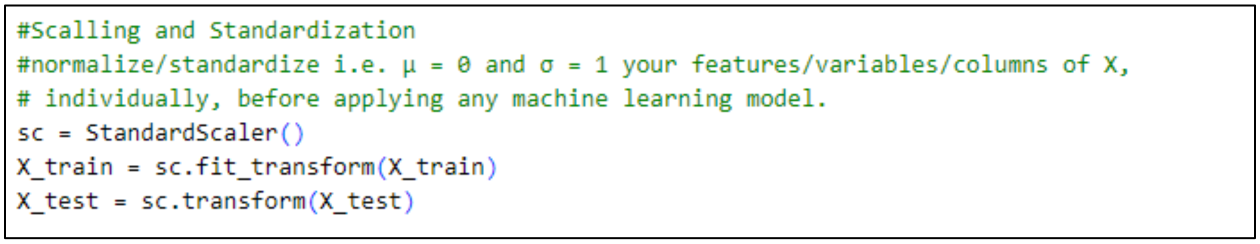
```
# we can apply feature selection to Select features according to the k highest scores
bestfeatures = SelectKBest(score_func=f_classif, k=3)
iris_trim = bestfeatures.fit_transform(X, y)
print(bestfeatures.scores_)
#SepalWidthCm has the least score so we can remove it from the features
X = dataset.iloc[:,[1,3,4]].values
```

```
from sklearn.model_selection import train_test_split
```



```
# Spliting dataset into features traning set(X_train , X_test) and
# target test set(y_train , y_test)
X_train, X_test , y_train , y_test = train_test_split(X ,y , test_size = 0.25 ,
random_state = 42)
```

```
from sklearn.preprocessing import StandardScaler
```



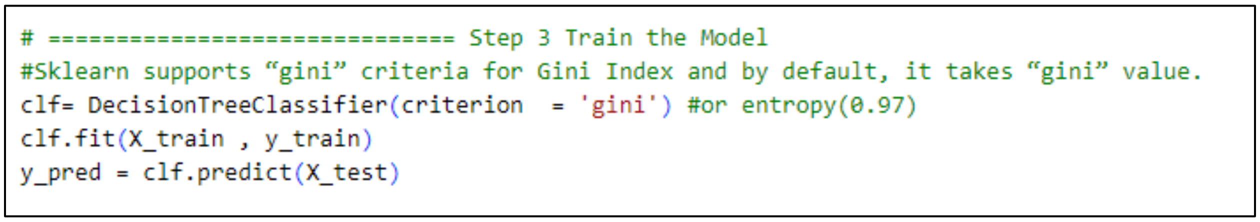
```
#Scalling and Standardization
#normalize/standardize i.e. μ = 0 and σ = 1 your features/variables/columns of X,
# individually, before applying any machine learning model.
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step3: Train the Model

We import the **DecisionTreeClassifier** from **sklearn.tree** , this Classifier implementing decision tree [30].

```
from sklearn.tree import DecisionTreeClassifier
```

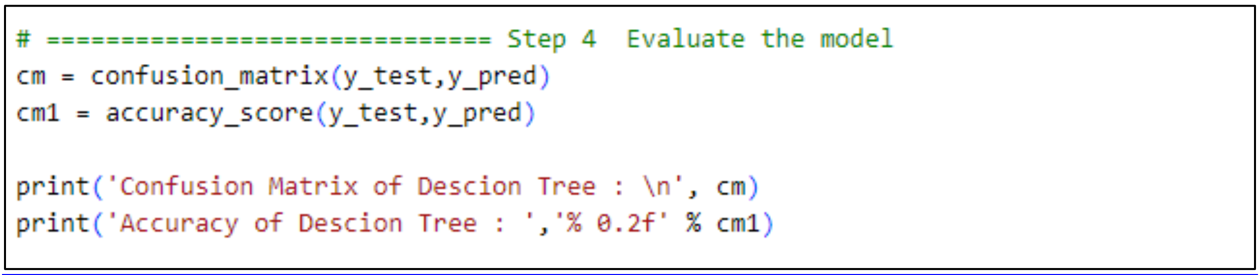
criterion is the parameter used to measure the quality of a split and it allows users to choose between 'Gini or 'Entropy' [31]. Gini Impurity measures the divergences between the probability distributions of the target attribute's values and splits a node such that it gives the least amount of impurity [32]. We choose 'gini' since its faster than 'Entropy' [31].



```
# ============================== Step 3 Train the Model
#Sklearn supports "gini" criteria for Gini Index and by default, it takes "gini" value.
clf= DecisionTreeClassifier(criterion  = 'gini') #or entropy(0.97)
clf.fit(X_train , y_train)
y_pred = clf.predict(X_test)
```

Step4: Evaluate the Model

```
from sklearn.metrics import confusion_matrix
```

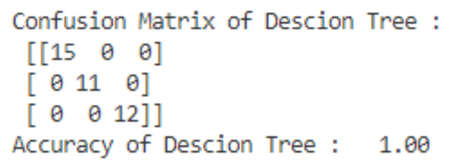


```
# ============================== Step 4  Evaluate the model
cm = confusion_matrix(y_test,y_pred)
cm1 = accuracy_score(y_test,y_pred)

print('Confusion Matrix of Descion Tree : \n', cm)
print('Accuracy of Descion Tree : ','% 0.2f' % cm1)
```

There are 15 iris flowers predicted as Iris-setosa and they were actually Iris-setosa.

There are 11 iris flowers predicted as Iris-versicolor and they were actually Iris-versicolor.

There are 12 iris flowers predicted as Iris-virginica and they were actually Iris-virginica.



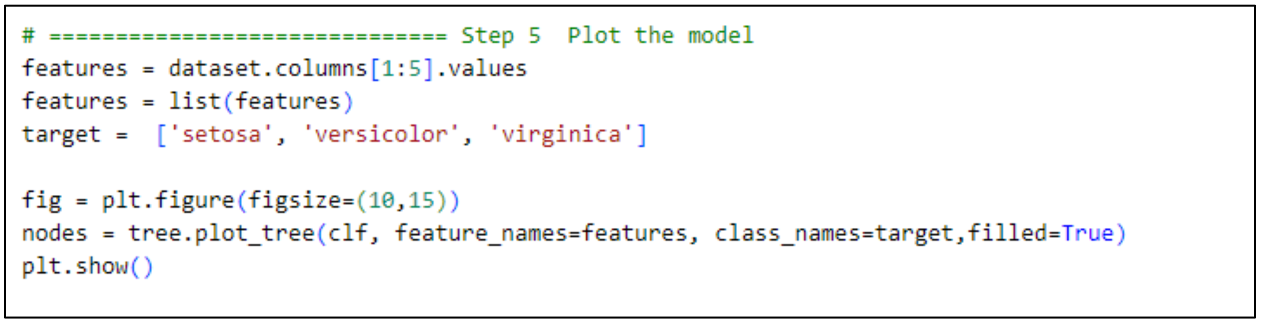
```
Confusion Matrix of Descion Tree :
 [[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
Accuracy of Descion Tree :   1.00
```

The accuracy score is 1.00 which means that all the predicted species by the model are the same as the **y_test** labels.

Overfitting

Overfitting happens when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples. Since the models perform well with both training and evaluation data then we are not suffering from overfitting.

Step5: Plot the Model



```
# ============================== Step 5  Plot the model
features = dataset.columns[1:5].values
features = list(features)
target =  ['setosa', 'versicolor', 'virginica']

fig = plt.figure(figsize=(10,15))
nodes = tree.plot_tree(clf, feature_names=features, class_names=target,filled=True)
plt.show()
```

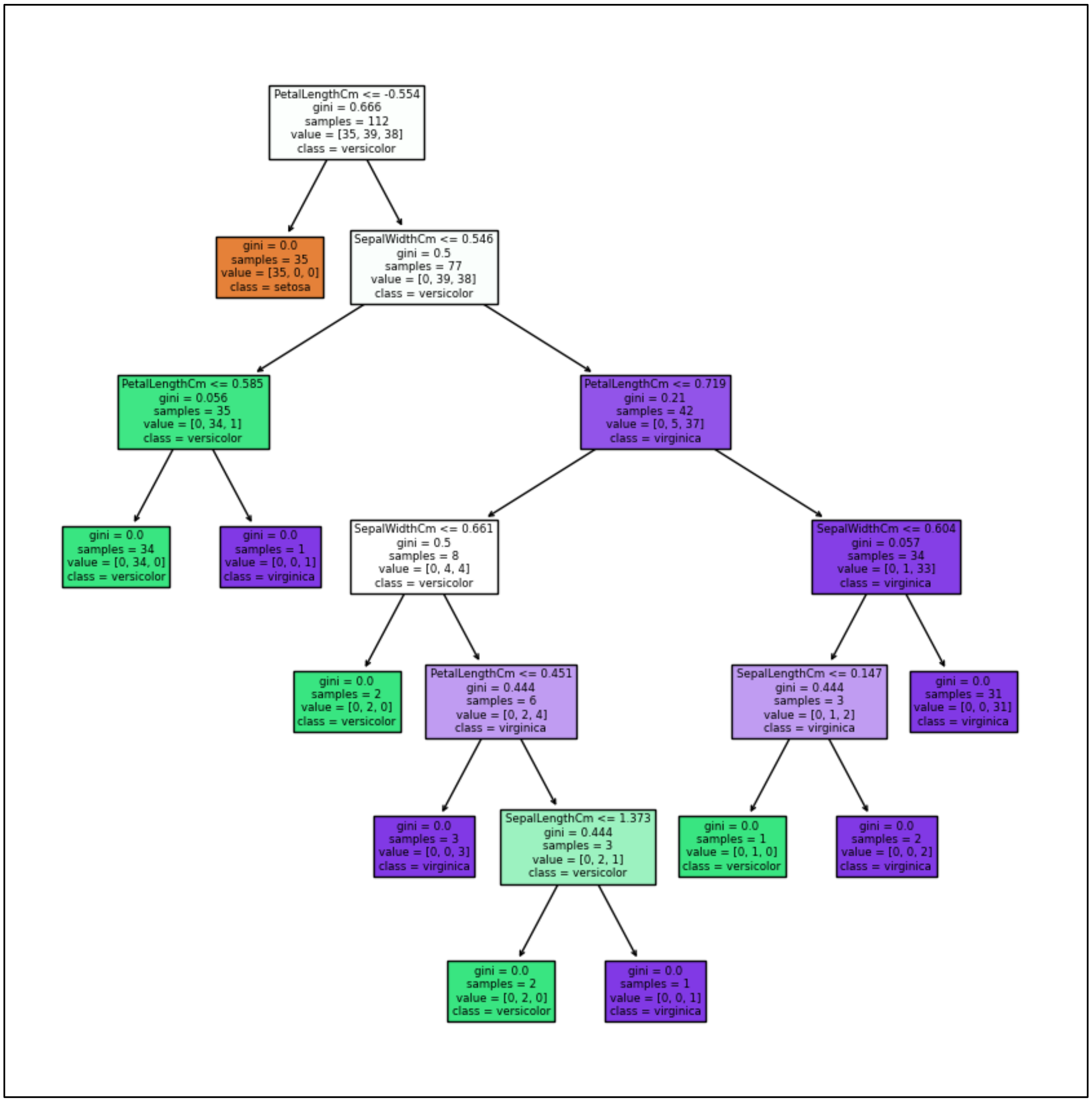
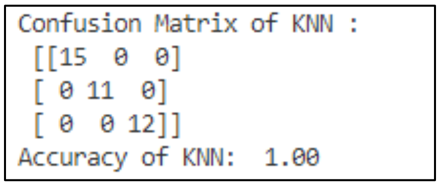


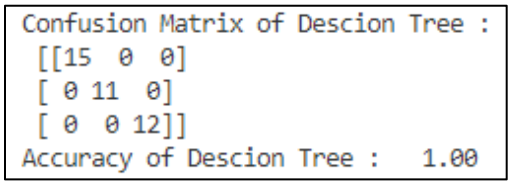
Figure (6): Decision Tree using gini impurity measure

5. Comparison

Both KNN and Decision Tree achieve the highest accuracy 1.00 and the same confusion matrix when both have the same test size 25% and random state 42



```
Confusion Matrix of KNN :
 [[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
Accuracy of KNN:  1.00
```



```
Confusion Matrix of Descion Tree :
 [[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
Accuracy of Descion Tree :   1.00
```

Both algorithms are non-parametric which mean they don't have an assumption on the distribution of the data. Both can be used for regression and classification problems. Decision trees can be faster that KNN in large data because KNN scans the whole dataset to predict as it doesn't generalize the data in advance [33]. However, iris flower dataset is only 150 record.

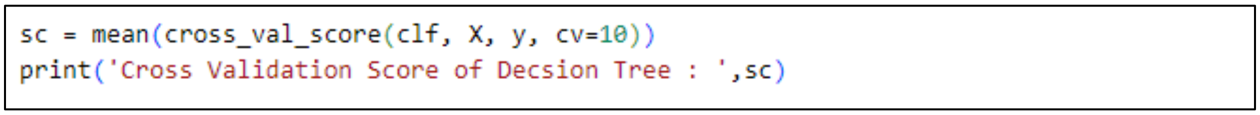
Since accuracy depends on the actual train/test datasets, which can be biased. We can use cross validation for better approximation [34].

We import **cross_val_score function** from **sklearn.model_selection**

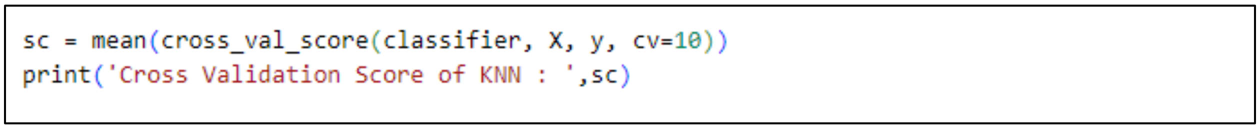
```
from sklearn.model_selection import cross_val_score
```

Apply the function in both KNN and Decision Tree models with cv = 10

It is always suggested that the value of k should be 10 as the lower value of k is takes towards validation and higher value of k leads to LOOCV method [35].

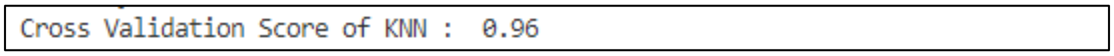


```
sc = mean(cross_val_score(clf, X, y, cv=10))
print('Cross Validation Score of Decsion Tree : ',sc)
```

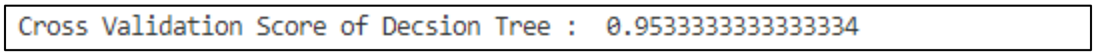


```
sc = mean(cross_val_score(classifier, X, y, cv=10))
print('Cross Validation Score of KNN : ',sc)
```

We find that cross validation score of KNN is higher than Decision Tree for the iris dataset



```
Cross Validation Score of KNN :  0.96
```



```
Cross Validation Score of Decsion Tree :  0.9533333333333334
```

6. Results and Discussion

KNN and Decision Tree are commonly used algorithms in supervised learning. Supervised is a machine learning approach defined by its use of labeled datasets like the iris flower. These datasets are designed to train(supervise) algorithms into classifying data or predicting outcomes accurately [36]. Each machine learning algorithm is designed to solve a specific type of problems. So, first of all, you should consider the type of project or dataset that you're dealing with. [37].

For the iris dataset we apply both KNN and Decision Tree. As we discussed the reasons of choosing the two algorithms for the iris dataset in section 3 (Justification of choice these 2 algorithms). We applied the steps of building machine learning models starting from importing the dataset and got to know its characteristics passing by applying feature selection to improve the performance of the machine learning models. We divided our dataset into train and test subsets and normalized the training dataset. We fitted the two models and perform the prediction to compare with the testing dataset. We could find that the two algorithms achieved the highest accuracy 1.00 and when both have the same test size 25% and random state 42 with the same confusion matrix values for the actual prediction result of the three species (Iris-setosa, Iris-versicolor, Iris-virginica). However, the cross validation of KNN was higher for this dataset thus its our champion algorithm in this case.

7. Lessons Learned from the Project

- Understanding KNN and Tree Algorithms.

- Dealing with Iris flower dataset and know its features and species.

- Identifying the Advantages of both KNN Algorithm and Decision Tree Algorithm.

- Determining when to use KNN and Tree Algorithms.

- Knowing and applying the steps of building machine learning models.

- Dealing with python libraries such as numpy, padnas and matplotlib.

- Plotting figures using matplotlib.

- Reading and Exploring dataset using python

- Defining Feature selection and its usage.

- Splitting data set to train and test subset and the reason behind that.

- Identifying parameter used in train/split like test_size and random_state.

- How to Standardize features in machine learning models.

- How to create machine learning models and the importance of its parameters.

- Fitting and predicting subsets datasets using machine learning models.

- Evaluating machine learning models based on accuracy, confusion matrix and cross validation.

- How to read the confusion matrix and understand its results.

- Identifying overfitting and reasons that lead to its occurrence.

- Comparing between machine learning to choose the best according to the dataset

8. References

[1] *K*-Nearest Neighbor (KNN) algorithm for Machine Learning - Javatpoint. www.javatpoint.com. (n.d.) from https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[2] Machine learning decision tree classification algorithm - javatpoint. www.javatpoint.com. (n.d.) from https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[3] MathNerd. (2018, March 22). Iris flower dataset. Kaggle from https://www.kaggle.com/datasets/arshid/iris-flower-dataset

[4] Brownlee, J. (2020, August 19). Difference between algorithm and model in machine learning. Machine Learning Mastery from https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/

[5] Kumar, A. (2020, May 27). KNN algorithm: What?when?why?how? Medium from https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

[6] What do you mean by noise in given dataset and how can you remove noise in dataset? i2tutorials. (2019, October 17) from https://www.i2tutorials.com/what-do-you-mean-by-noise-in-given-dataset-and-how-can-you-remove-noise-in-dataset/

[7] What are the advantages and disadvantages of KNN classifier? i2tutorials. (2019, November 15) from https://www.i2tutorials.com/advantages-and-disadvantages-of-knn-classifier/

[8] K nearest neighbor Pros & Cons - holypython.com. (n.d.) from https://holypython.com/knn/k-nearest-neighbor-pros-cons/

[9] Editorial. (2020, July 30). When to consider decision tree algorithm - pros and cons. RoboticsBiz from https://roboticsbiz.com/when-to-consider-decision-tree-algorithm-pros-and-cons/

[10] Sharma, M. (n.d.). Decision tree algorithm introduction. Cloud Training Program from https://k21academy.com/datascience/decision-tree-algorithm/

[11] BotBark. (2020, November 8). Top 6 advantages and disadvantages of Decision Tree Algorithm. Bot Bark from https://botbark.com/2019/12/19/top-6-advantages-and-disadvantages-of-decision-tree-algorithm/

[12] Numpy introduction. Introduction to NumPy. (n.d.) from https://www.w3schools.com/python/numpy/numpy_intro.asp

[13] Pandas tutorial. (n.d.). Retrieved July 6, 2022, from https://www.w3schools.com/python/pandas/default.asp

[14] Pandas.dataframe. pandas.DataFrame - pandas 1.4.3 documentation. (n.d.) from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

[15] GaganGagan 1, marilena.oitamarilena.oita 84977 silver badges1212 bronze badges, & AntoineAntoine 77666 silver badges2121 bronze badges. (1964, August 1). Feature selection & significant features in KNN. Stack Overflow from https://stackoverflow.com/questions/42016078/feature-selection-significant-features-in-knn

[16] Sklearn.feature_selection.Selectkbest. scikit. (n.d.) from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[17] Zhang, J. (2021, April 16). Data Engineering: Feature Selection with an Iris dataset example. LinkedIn. from https://www.linkedin.com/pulse/data-engineering-feature-selection-iris-dataset-example-justin-zhang

[18] Train and test datasets in Machine Learning - Javatpoint. www.javatpoint.com. (n.d.) from https://www.javatpoint.com/train-and-test-datasets-in-machine-learning

[19] Sklearn.model_selection.train_test_split. scikit. (n.d.) from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[20] "python - What is 'random-state' in sklearn.model_selection.train_test_split example? - Stack Overflow." https://stackoverflow.com/questions/49147774/what-is-random-state-in-sklearn-model-selection-train-test-split-example (accessed Jul. 06, 2022).

[21] Sklearn.preprocessing.StandardScaler. scikit. (n.d.) from https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[22] Why do data scientists use SKLEARN's StandardScaler and what does it do? Adrian Oprea | Website Performance and Core Web Vitals Optimization. (2018, June 2) from https://oprea.rocks/blog/why-use-sklearn-preprocessing-standardscaler

[23] Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved July 6, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[24] Sklearn.metrics.accuracy_score. scikit. (n.d.) from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[25] Pyplot tutorial#. Pyplot tutorial - Matplotlib 3.5.2 documentation. (n.d.) from https://matplotlib.org/stable/tutorials/introductory/pyplot.html

[26] Sklearn.metrics.confusion_matrix. scikit. (n.d.) from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[27] Compute Classification Report and confusion matrix in Python. GeeksforGeeks. (2022, March 18) from https://www.geeksforgeeks.org/compute-classification-report-and-confusion-matrix-in-python/

[28] pseudomarvinpseudomarvin 1, errorerror 2, & chao chongchao chong 4122 bronze badges. (1965, February 1). Graph K-NN decision boundaries in Matplotlib. Stack Overflow from https://stackoverflow.com/questions/45075638/graph-k-nn-decision-boundaries-in-matplotlib

[29] Matplotlib.colors.listedcolormap#. matplotlib.colors.ListedColormap - Matplotlib 3.5.2 documentation. (n.d.) from https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.ListedColormap.html

[30] Sklearn.tree.decisiontreeclassifier. scikit. (n.d.) from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[31] Decision trees: Gini vs entropy ⋆ Quantdare. Quantdare. (2020, December 13). Retrieved from https://quantdare.com/decision-trees-gini-vs-entropy/

[32] Mithrakumar, M. (2019, November 12). How to tune a decision tree? Medium. Retrieved from https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

[33] Afeworki, M. (2021, September 9). A comparison of machine learning algorithms: KNN vs decision trees. Medium from https://milena-pa.medium.com/a-comparison-of-machine-learning-algorithms-knn-vs-decision-trees-d6110e08bfea

[34] Why the model has high accuracy on test data, but lower with cross ... (n.d.) from https://www.researchgate.net/post/why_the_model_has_high_accuracy_on_test_data_but_lower_with_cross-validation

[35] 44, A. S. 44@A. S. (2020, January 7). Cross validation in Machine Learning. GeeksforGeeks from https://www.geeksforgeeks.org/cross-validation-machine-learning/

[36] IBM learning. (n.d.) from https://ibm-learning.udemy.com/

[37] Sydorenko, I. (2021, May 4). How to choose a machine learning algorithm. High quality data annotation for Machine Learning from https://labelyourdata.com/articles/how-to-choose-a-machine-learning-algorithm