Final Assignment Report: Decision Tree Classification in Python and K-Nearest Neighbor (KNN) classification Using Python

Yeasmin Akter
CSE-0408 Artificial Intelligence Lab
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
yasminepsha@gmail.com

I. Assignment Name:

Decision Tree Classification in Python

***Abstract*—Decision Trees usually implement exactly the human thinking ability while making a decision, so it is easy to understand.A Decision Tree is a supervised Machine learning algorithm. It is used in both classification and regression algorithms. The decision tree is like a tree with nodes. The branches depend on a number of factors. It splits data into branches like these till it achieves a threshold value. A decision tree consists of the root nodes, children nodes, and leaf nodes.**

***Index Terms*—Solving Decision Tree Classification in Python**

II. Introduction

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily 35 mimics the human level thinking. That is why decision trees are easy to understand and interpret.

III. II. LITERATURE REVIEW

A Decision Tree is a useful and popular classification technique that inductively learns a model from a given set of data. One reason for its popularity stems from the availability of existing algorithms that can be used to build decision trees, such as CART (Breiman et al., 1984), ID3 (Quinlan, 1986), and C4.5 (Quinlan, 1993). These algorithms all learn decision trees from a supplied set of training data, but do so in slightly different ways. As discussed in the introduction, a classifier is built by analyzing training data. That is to say, a classifier is built by analyzing a collection of instances where each instance is composed of a set of attribute values and a known class value. These decision tree algorithms build top down structures that partition instances into separate classes, and it is hoped that these structures generalize well to instances with unknown class values. This would mean that the decision trees have fulfilled their objectives and have indeed discovered some underlying property of the data (Quinlan, 1986). The idea behind the decision tree's structure is to split instances into separate groups that are as homogeneous as possible, and these splits are based upon the different values for particular attributes. Typically, the most meaningful attribute of a dataset is the attribute that when data is split along its possible values leads to the most homogeneous groups of data being formed. In most situations, it is this attribute that is selected to be the first branch of the tree.

IV. Advantage Of Decision Tree Classification

*Easy to Understand
*Easy to Generate Rules
*Inexpensive to construct
*Extremely Fast at classifying unknown records
*A decision tree does not require normalization of data.
*A decision tree does not require scaling of data as well.
*A Decision tree model is very intuitive and easy to explain to technical teams

V. Disadvantage of Decision Tree classification

*May suffer from overfitting
*can be Quite large
*Does not handle strearning data easily
*Decision boundary restricted to being parallel to attribute axes.

VI. . RESULT AND SOLVING PROBLEM

In this paper, Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

VII. MAIN CODE

Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion$_matrix$
$from sklearn.cross_validation import train_test_split$
$from sklearn.tree import$
$DecisionTreeClassifier$
$from sklearn.metrics import accuracy_score$
$from sklearn.metrics import$
$classification_report$

$Function importing Dataset def importdata()$:
$balance_data = pd.read_csv('https$:
$//archive.ics.uci.edu/ml/machine - learning -' +$
$'databases/balance - scale/balance - scale.data', sep =$
$',', header = None)$

Printing the dataswet shape
print ("Dataset Length: ",
len(balance$_data$))
$print("Dataset Shape : ",$
$balance_data.shape)$

$Printing the dataset observations print("Dataset$:
$", balance_data.head())return balance_data$

Function to split the dataset def splitdataset(balance$_data$) :

Separating the target variable
X = balance$_data.values[:, 1 : 5]$
$Y = balance_data.values[:, 0]$

Splitting the dataset into train and test
$X_train, X_test, y_train, y_test = train_test_split($
$X, Y, test_size = 0.3, random_state = 100)$

return X, Y, $X_train, X_test, y_train, y_test$

Function to perform training with giniIndex.
def train$_using_gini(X_train, X_test, y_train)$:

Creating the classifier object clf$_gini$ =
$DecisionTreeClassifier(criterion = "gini",$
$random_state = 100, max_depth = 3, min_samples_leaf = 5)$

Performing training clf$_gini.fit(X_train, y_train)return clf_gini$

Function to perform training with entropy. def tarin$_using_entropy(X_train, X_test, y_train)$:

Decision tree with entropy clf$_entropy$ =
$DecisionTreeClassifier(criterion$ =
$"entropy", random_state = 100, max_depth$ =
$3, min_samples_leaf = 5)$

Performing training clf$_entropy.fit(X_train, y_train)return clf_entropy$

Function to make predictions def prediction(X$_test, clf_object)$:

Predicton on test with giniIndex y$_pred$ =
$clf_object.predict(X_test)print("Predicted values$:
$")print(y_pred)return y_pred$

Function to calculate accuracy def cal$_accuracy(y_test, y_pred)$:

print("Confusion Matrix: ",
confusion$_matrix(y_test, y_pred))$

print ("Accuracy : ",
accuracy$_score(y_test, y_pred) * 100)$

print("Report : ",
classification$_report(y_test, y_pred))$

Driver code
def main():

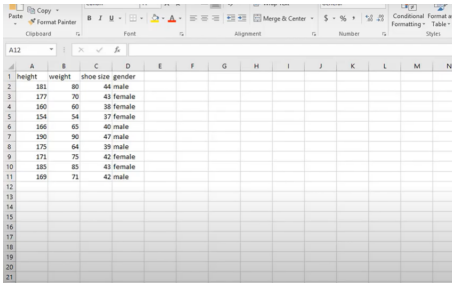
Building Phase
data = importdata()
X, Y, $X_train, X_test, y_train, y_test = splitdataset(data)$
$clf_gini = train_using_gini(X_train, X_test, y_train)$
$clf_entropy =$
$tarin_using_entropy(X_train, X_test, y_train)$

Operational Phase print("Results Using Gini Index:")

Prediction using gini
y$_pred_gini = prediction(X_test, clf_gini)cal_accuracy(y_test, y_pred_gini)$

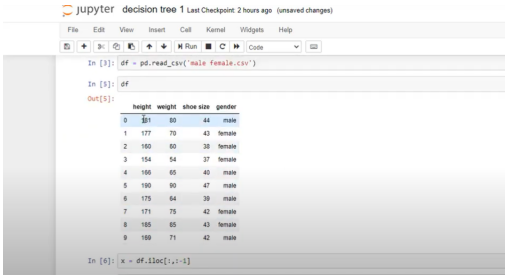
print("Results Using Entropy:")
Prediction using entropy
y$_pred_entropy = prediction(X_test, clf_entropy)$
$cal_accuracy(y_test, y_pred_entropy)$

Calling main function
if $_name_{=="}{}_main_{":main()}$



	height	weight	shoe size	gender
2	181	80	44	male
3	177	70	43	female
4	160	60	38	female
5	154	54	37	female
6	166	65	40	male
7	190	90	47	male
8	175	64	39	male
9	171	75	42	female
10	185	85	43	female
11	169	71	42	male

Fig. 1. Output1



	height	weight	shoe size	gender
0	181	80	44	male
1	177	70	43	female
2	160	60	38	female
3	154	54	37	female
4	166	65	40	male
5	190	90	47	male
6	175	64	39	male
7	171	75	42	female
8	185	85	43	female
9	169	71	42	male

Fig. 2. Output1

VIII. Assignment Name:

K-Nearest Neighbor (KNN) classification Using Python

***Abstract*—K-Nearest Neighbours (KNN) is an effortless but productive machine learning algorithm. It is effective for classification as well as regression. However, it is more widely used for classification prediction. KNN groups the data into coherent clusters or subsets and classifies the new inputted data based on its similarity with previously trained data. The input is assigned to the class with which it shares the most nearest neighbours. Though KNN is effective, it has many weaknesses. This paper highlights the KNN method and it's modified versions available in previously done researches. These variants remove the weaknesses of KNN and provide a more efficient method K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.**

Therefore, larger k value means smother curves of separation resulting in less complex models.

***Index Terms*—Solving KNN classification using python**

IX. Introduction

The K-Nearest-Neighbours (KNN) is a non-parametric classification algorithm, i.e. , it does not make any presumptions on the elementary dataset. It is known for it's simplicity and effectiveness. It is a supervised learning algorithm. A labeled training dataset is provided where the data points are categorized into various classes , so that class of the unlabeled data can be predicted. In Classification, different characteristics determine the class to which the unlabeled data belongs. KNN is mostly used as a classifier. It is used to classify data based on closest or neighbouring training examples in a given region. This method is used for its simplicity of execution and low computation time. For continuous data, it uses the euclidean distance to calculate its nearest neighbours . For a new input the K nearest neighbours are calculated and the majority among the neighbouring data ecides the classification for the new input. Even though this classifier is simple, the value of 'K' plays an important role in classifying the unlabeled data. There are many ways to decide the values for 'K', but we can simply run the classifier multiple times with different values to see which value gives the most effective result. The computation cost is slightly high because all the calculations are made when the training data is being classified, not when it is encountered in the dataset. It is a lazy learning algorithm as not much is done when the dataset is being trained except storing the training data and memorizing the dataset instead. It does not perform generalization on the training dataset. So the entire fundamental dataset being trained is required when in the testing stage. In regression, KNN predicts continuous values. This value is the average of the values of its K -nearest neighbours.

Using the k-nearest neighbor algorithm we fit the historical data (or train the model) and predict the future.

X. Proposed Methodology

A data set with lots of different points and labelled data is the ideal to use.
The best languages to use with KNN are R and python.
To find the most accurate results from your data set, you need to learn the correct practices for using this algorithm.

XI. KNN Algorithm

To understand better the working KNN algorithm applies the following steps when using it:

Step 1 – When implementing an algorithm, you will always need a data set. So, you start by loading the training and the test data.
Step 2 – Choose the nearest data points (the value of K). K can be any integer.
Step 3 – Do the following, for each test data –

1 – Use Euclidean distance, Hamming, or Manhattan to calculate the distance between test data and each row of training. The Euclidean method is the most used when calculating distance.
2 – Sort data set in ascending order based on the distance value.

3 – From the sorted array, choose the top K rows.

4 – Based on the most appearing class of these rows, it will assign a class to the test point.

Step 4 – End

XII. Advantage of KNN classification

*Quick calculation time
*Simple algorithm – to interpret
*Versatile – useful for regression and classification

*High accuracy – you do not need to compare with better-supervised learning models
*No assumptions about data – no need to make additional assumptions, tune several parameters, or build a model. This makes it crucial in nonlinear data case.

XIII. Disadvantage of KNN classification

*Accuracy depends on the quality of the data
*With large data, the prediction stage might be slow
*Sensitive to the scale of the data and irrelevant features
*Require high memory – need to store all of the training data
*Given that it stores all of the training, it can be computationally expensive

XIV. Summary

K is a positive integer
With a new sample, you have to specify K
K is selected from database closest to the new sample KNN doesn't learn any model
KNN makes predictions using the similarity between an input sample and each training instance.
This blog has given you the fundamentals of one of the most basic machine learning algorithms. KNN is a great place to start when first learning to build models based on different data sets. Data set with a lot of different points and accurate information is your best place, to begin with KNN.

XV. Conclusion

The K Nearest Neighbors algorithm doesn't require any additional training when new data becomes available. Rather it determines the K closest points according to some distance metric (the samples must reside in memory). Then, it looks at the target label for each of the neighbors and places the new found data point into the same category as the majority. Given that KNN computes distance, it's imperative that we scale our data. In addition, since KNN disregards the underlying features, it's our responsibility to filter out any features that are deemed irrelevant.

Acknowledgment

I would like to express my gratefulness and gratitude to all those people who gave me the inspiration to complete this proposal. A special thanks to my instructor **Khan Md. Hasib Sir**, Lecturer, department of Computer Science and Engineering (CSE), whose help, stimulating suggestions, corrections and encouragement gave me the opportunity to coordinate information in writing this research proposal.

References

[1] thebibliography

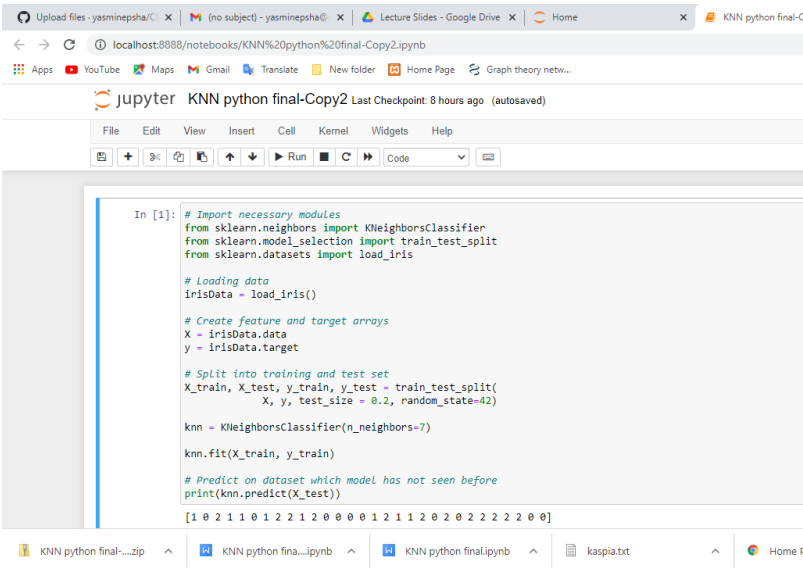


Fig. 3. Output

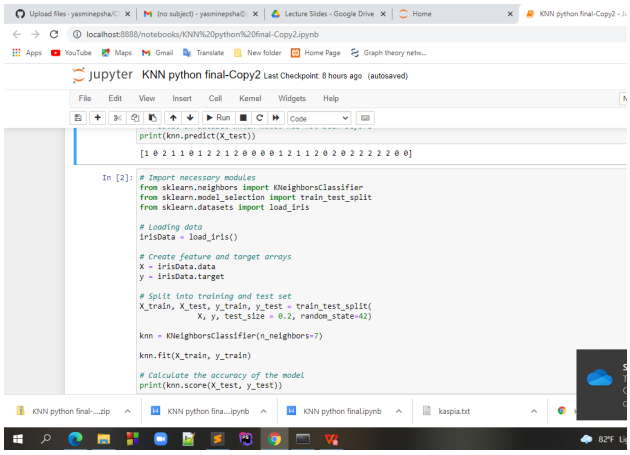


Fig. 4. Output

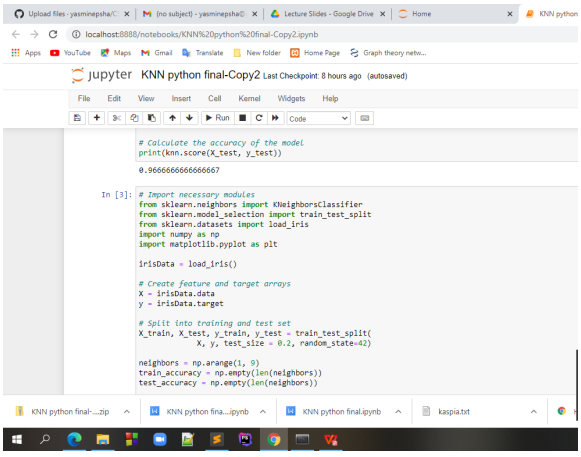


Fig. 5. Output

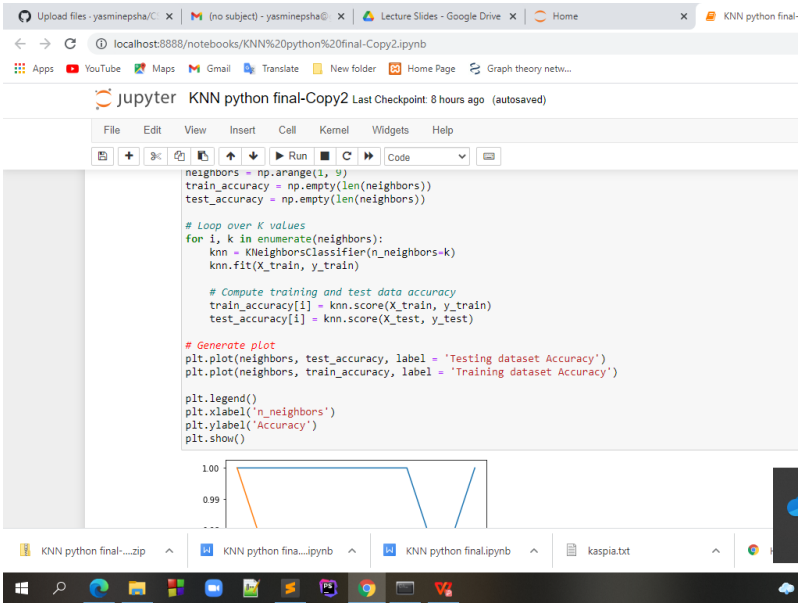


Fig. 6. Output

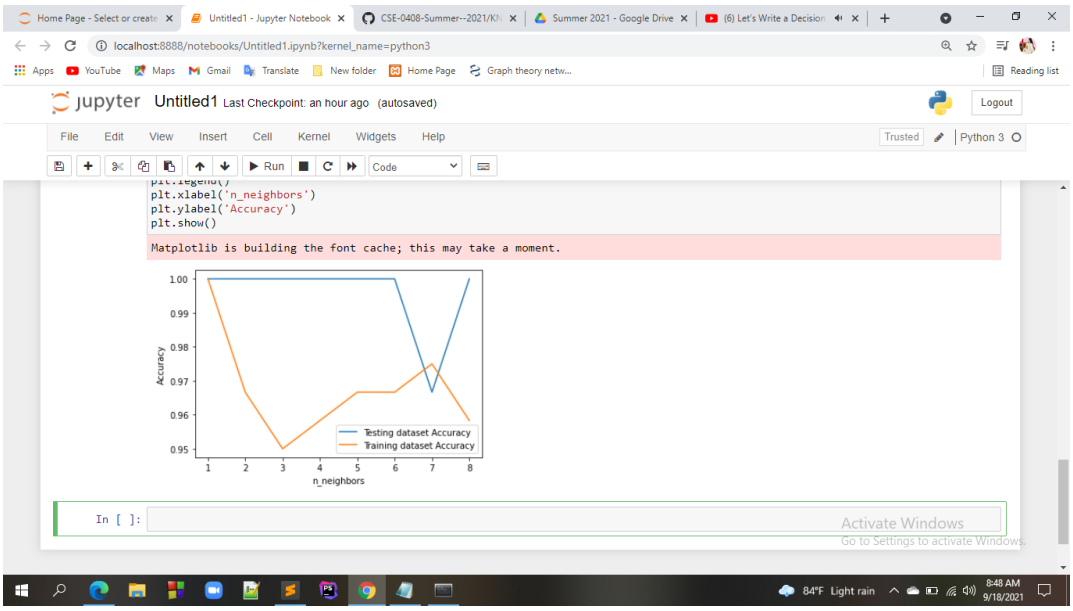


Fig. 7. Output