Final Report On Decision Tree

CSE-0408 Summer 2021

Sheikh Afrin
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
sheikhafrin2016@gmail.com

***Abstract*—Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.**

***Index Terms*—Python**

I. Introduction

A decision tree is a graphical representation of all possible solutions to a decision. These days, tree-based algorithms are the most commonly used algorithms in the case of supervised learning scenarios.They are easier to interpret and visualize with great adaptability. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

Decision tree is a flowchart-like tree structure where an internal node represents feature, the branch represents a decision rule, and each leaf node represents the outcome.

II. Literature Review

Lertworaprachaya et al.,2014 [1] proposed a new model for compose decision trees using interval-valued fuzzy membership values.Most existing fuzzy decision trees do not consider the concerned associated with their membership values; however, precise values of fuzzy membership values are not always possible.Bahnsen et al. 2015 [2] proposed an example-reliant costsensitive decision tree algorithm, by incorporating the different example-reliant costs into a new cost-based impurity measure and new cost-based pruning criteria.Subsequently, using three different databases,credit scoringand direct marketing, the authors evaluated their proposed method.

III. Proposed Methodology

Decision trees are a simple classification tool capable of separating records of data into specific categories by proposing a series of questions.Decision trees are commonly used due to many factors, including their relatively small learning curve for interpretability. Kingsford and Salzberg note that decision trees are commonly more easily interpreted than other machine learning algorithms. The decision tree structure follows an intuitive tree shape that can be interpreted by following a series of questions at each level. The responses to each question in the tree can include either discrete values, a range, or a probability distribution.

IV. Types Of Nodes

A decision tree consists of three types of nodes:

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

V. ADVANTAGES

1.Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.

2.Help determine worst, best and expected values for different scenarios.

3.Use a white box model. If a given result is provided by a model.

4.Can be combined with other decision techniques.

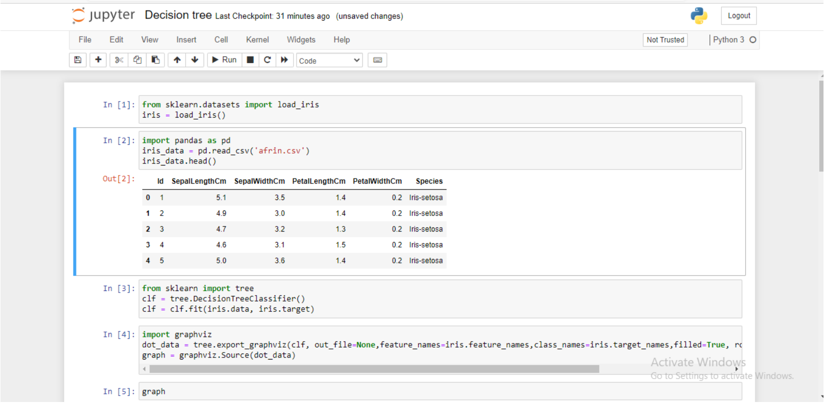
VI. DISADVANTAGES

1.They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.

2.They are often relatively inaccurate.

3.Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.

VII. CODE


jupyter
Decision tree Last Checkpoint: 31 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Activate Windows
Go to Settings to activate Windows.

```
In [1]: from sklearn.datasets import load_iris
        iris = load_iris()

In [2]: import pandas as pd
        iris_data = pd.read_csv('afrin.csv')
        iris_data.head()
```

Out[2]:

	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

```
In [3]: from sklearn import tree
        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(iris.data, iris.target)

In [4]: import graphviz
        dot_data = tree.export_graphviz(clf, out_file=None,feature_names=iris.feature_names,class_names=iris.target_names,filled=True, rc
        graph = graphviz.Source(dot_data)

In [5]: graph
```

Fig. 1.

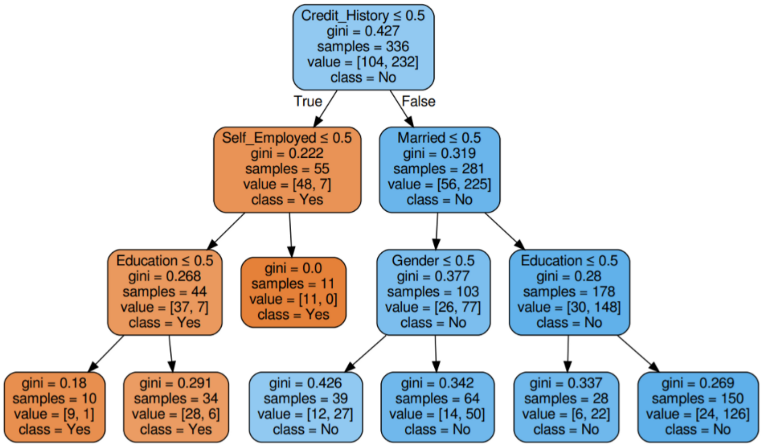


Fig. 2.

VIII. Conclusion

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] Kadiyala, A., & Kumar, A. (2018). Applications of python to evaluate the performance of decision tree-based boosting algorithms. Environmental Progress & Sustainable Energy, 37(2), 618-623.

[2] Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.

[3] Patel, H. H., & Prajapati, P. (2018). Study and analysis of decision tree based classification algorithms. International Journal of Computer Sciences and Engineering, 6(10), 74-78.

Final Report On K-Nearest Neighbors

CSE-0408 Summer 2021

Sheikh Afrin
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
sheikhafrin2016@gmail.com

***Abstract*—This algorithm is used to solve the classification model problems. 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.**

***Index Terms*—Python**

I. Introduction

The K-Nearest-Neighbors (KNN) is a nonparametric 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 the 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 .

II. Literature Review

Along the years, a great effort was done in the scientific community in order to solve or mitigate the imbalanced dataset problem. Specifically for KNN, there are several balancing methods based on this algorithm. This section will provide a bibliographic review about the KNN and its derivate algorithms for dataset balancing. Also, the random oversampling and undersampling methods, the class overlapping problem, and evaluation measures will be reviewed.

III. Proposed Methodology

K-Nearest Neighbors (kNN) is such a method and, despite its simplicity, continues to perform fairly well for large training sets. It essentially relies only on the most basic assumption underlying all prediction: that observations with similar characteristics will tend to have similar outcomes. Nearest Neighbor methods assign a predicted value to a new observation based on the plurality or mean of its k "Nearest Neighbors" in the training set.K-Nearest Neighbors (KNN) is a standard machine-learning method that has been extended to large-scale data mining efforts.

IV. ADVANTAGES

1. No Training Period: KNN is called Lazy Learner.It does not learn anything in the training period. It does not derive any discriminative function from the training data.

2. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

3. KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function.

V. DISADVANTAGES

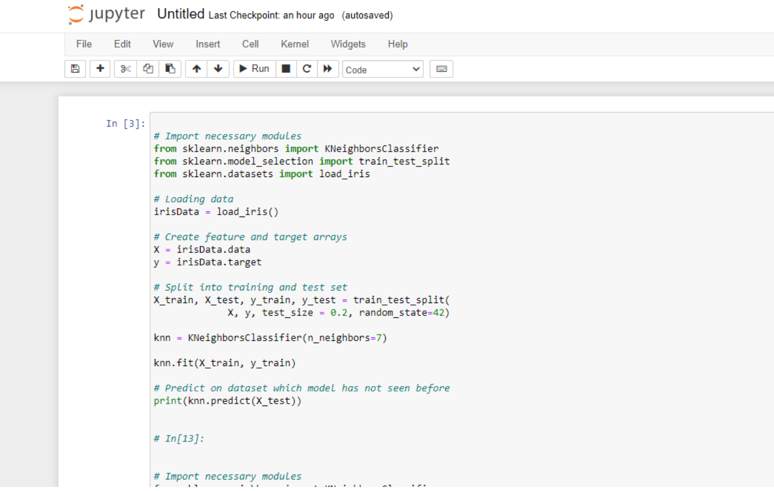
1. Does not work well with large dataset.

2. Does not work well with high dimensions.

3. Need feature scaling: We need to do feature scaling before applying KNN algorithm to any dataset.

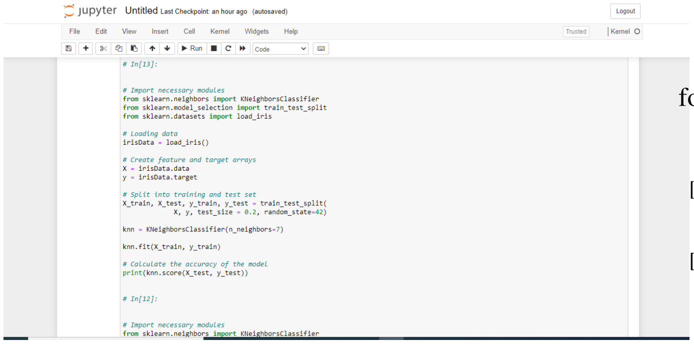
4. Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset.

VI. CODE



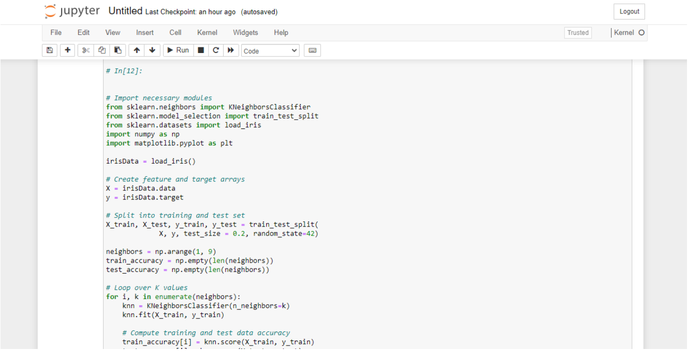
```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Loading data
irisData = load_iris()

# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)

# Predict on dataset which model has not seen before
print(knn.predict(X_test))


# In[13]:


# Import necessary modules
```

Fig. 1.



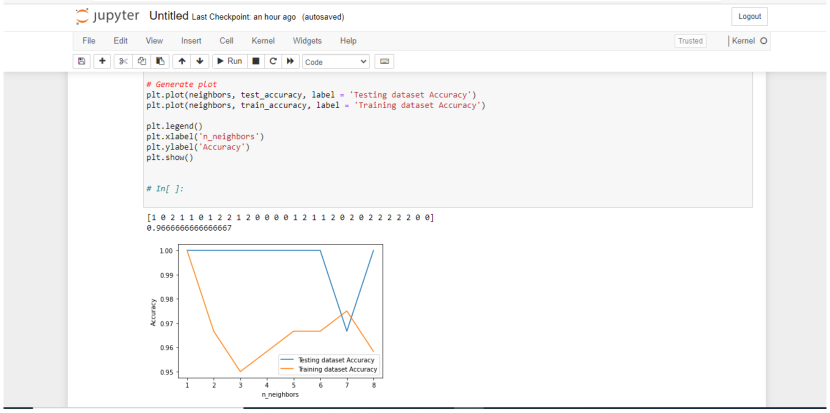
```
# In[13]:


# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Loading data
irisData = load_iris()

# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)

# Calculate the accuracy of the model
print(knn.score(X_test, y_test))


# In[12]:


# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
```

Fig. 2.



```
# In[12]:


# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

irisData = load_iris()

# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Compute training and test data accuracy
    train_accuracy[i] = knn.score(X_train, y_train)
```

Fig. 3.



```
# Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()


# In[ ]:
```

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
0.9666666666666667
```

Fig. 4.

VII. Conclusion

Machine learning algorithms have improved with the increase in research and data mining tools. K- nearest neighbour algorithm is a simple but high accuracy algorithm that has proven effective in several cases.The nearest neighbour algorithm works by classifying the new unlabeled data by examining the classes of it's nearest neighbours.In KNN algorithm, a constant number of nearest neighbours determine the classification of an unlabeled data which is assigned by K, where K is a positive integer.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] Beckmann, M., Ebecken, N. F., & de Lima, B. S. P. (2015). A KNN undersampling approach for data balancing. Journal of Intelligent Learning Systems and Applications, 7(04), 104.

[2] Makkar, T., Kumar, Y., Dubey, A. K., Rocha, Á., & Goyal, A. (2017, December). Analogizing time complexity of KNN and CNN in recognizing handwritten digits. In 2017 Fourth International Conference on Image Information Processing (ICIIP) (pp. 1-6). IEEE.

[3] Hazra, T. K., Singh, D. P., & Daga, N. (2017, August). Optical character recognition using KNN on custom image dataset. In 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON) (pp. 110-114). IEEE.