Final Report on Decision Tree

CSE-0408 Summer 2021

Afifa Nowreen Akhter
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
afifa.nowreen@gmail.com

***Abstract*—Decision Tree algorithm belongs to the family of supervised learning algorithms. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).**

***Index Terms*—Decision, Tree, Dataset**

I. Introduction

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. The decisions or the test are performed on the basis of features of the given dataset.

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. Because of that, the authors represented fuzzy membership values as distance to model concerned and employ the look-ahead based fuzzy decision tree induction method to construct decision trees. The authors also measured the significance of different neighbourhood values and define a new parameter unkind to specific data sets using fuzzy sets. Some examples are provided to establish the effectiveness of their approach.

III. Proposed Methodology

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

IV. Advantages of the Decision Tree

1. It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
2. It can be very useful for solving decision-related problems. It helps to think about all the possible outcomes for a problem.
3. There is less requirement of data cleaning compared to other algorithms.

V. Disadvantages of the Decision Tree

1. The decision tree contains lots of layers, which makes it complex.
2. It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
3. For more class labels, the computational complexity of the decision tree may increase.

VI. 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] Ben-Haim, Y., & Tom-Tov, E. (2010). A Streaming Parallel Decision Tree Algorithm. Journal of Machine Learning Research, 11(2).
[2] Brijain, M., Patel, R., Kushik, M. R., & Rana, K. (2014). A survey on decision tree algorithm for classification.
[3] Freund, Y., & Mason, L. (1999, June). The alternating decision tree learning algorithm. In icml (Vol. 99, pp. 124-133).
[4]] Jin, C., De-Lin, L., & Fen-Xiang, M. (2009, July). An improved ID3 decision tree algorithm. In 2009 4th International Conference on Computer Science & Education (pp. 127-130). IEEE.
[5] Su, J., & Zhang, H. (2006, July). A fast decision tree learning algorithm. In Aaai (Vol. 6, pp. 500-505).
[6] Chandra, B., & Varghese, P. P. (2008). Fuzzy SLIQ decision tree algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 38(5), 1294-1301.

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

```
dataset = pd.read_csv("dtdata.csv")
```

```
dataset
```

	Income	gender	Mstatus	Ages
0	High	Male	Single	<21
1	High	Male	Married	<21
2	High	Male	Single	21-35
3	Medium	Male	Single	Weak
4	Low	Female	Single	>35
5	Low	Female	Married	>35
6	Low	Female	Married	21-35
7	Medium	Male	Single	>21
8	Low	Female	Married	<21
9	Medium	Female	Single	>35
10	Medium	Female	Married	<21
11	Medium	Male	Married	21-35
12	High	Female	Single	21-35
13	Medium	Male	Married	>35

Fig. 1. Code of Decision Tree

12	High	Female	Single	21-35
13	Medium	Male	Married	>35

```
X = dataset.iloc[:,:-1]
X
```

	Income	gender	Mstatus
0	High	Male	Single
1	High	Male	Married
2	High	Male	Single
3	Medium	Male	Single
4	Low	Female	Single
5	Low	Female	Married
6	Low	Female	Married
7	Medium	Male	Single
8	Low	Female	Married
9	Medium	Female	Single
10	Medium	Female	Married
11	Medium	Male	Married
12	High	Female	Single
13	Medium	Male	Married

Fig. 2. Code of Decision Tree

```
13  Medium      Male    Married
```

```
y= dataset.iloc[:,3]
```

```
y
```

```
0        <21
1        <21
2     21-35
3      Weak
4        >35
5        >35
6     21-35
7        >21
8        <21
9        >35
10       <21
11    21-35
12    21-35
13       >35
Name: Ages, dtype: object
```

Fig. 3. Code of Decision Tree

Final Report on KNN

***Abstract*—KNN stands for k-Nearest Neighbours. It is a supervised learning algorithm. KNN is very simple to implement and is most widely used as a first step in any machine learning setup. It can be used to solve both classification and regression problems. It is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM).**

***Index Terms*—KNN, Neighbours, Dataset**

I. Introduction

KNN 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. KNN 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 KNN algorithm. KNN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

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

1. Calculate the distance between test data and each row of training data.
2. Sort the calculated distances in ascending order based on distance values.
3. Get top k rows from the sorted array.
4. Get the most frequent class of these rows.
5. Return the predicted class.

IV. Advantages of KNN

1. It is simple to implement.
2. It is robust to the noisy training data.
3. It can be more effective if the training data is large.

V. Disadvantages of KNN

1. Always needs to determine the value of K which may be complex some time.
2. The computation cost is high because of calculating the distance between the data points for all the training samples.

VI. 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.

Acknowledgment

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

References

[1] D. Wettschereck and D. Thomas G., "Locally adaptive nearest neighbour algorithms," Adv. Neural Inf. Process. Syst., pg. 184–186, 1994.

[2] Shengyi Jiang,Guansong Pang,Meiling Wu,Limin Kuang, "An improved K-nearest-neighbour algorithm for text categorization",Expert Systems with Applications,Elsevier(2012)

[3] Han EH.., Karypis G., Kumar V. (2001) "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification". In: Cheung D., Williams G.J., Li Q. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2001. Lecture Notes in Computer Science, vol 2035. Springer, Berlin, Heidelberg.

[4]] Song Yang, Jian Huang, Ding Zhou, Hongyuan Zha, and C. Lee Giles, "Iknn: Informative k-nearest neighbour pattern classification," in Knowledge Discovery in Databases, (2007), pg. 248– 264.

[5] Giuseppe Nuti, "An Efficient Algorithm for Bayesian Nearest Neighbours",Cornell University, (2017)

[6] Wei Zheng, HaiDong Wang, Lin Ma, RuoYi Wang, "An Improved k-Nearest Neighbour Classification Algorithm Using Shared Nearest Neighbour Similarity" , Metallurgical & Mining Industry . (2015), Issue 10, pg. 133-137. 5p.

[7] Shiliang Sun ; Rongqing Huang ," An adaptive k-nearest neighbour algorithm", in 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discover.

[8] P. WiraBuana, S. Jannet D.R.M., and I. Ketut Gede Darma Putra, "Combination of K-Nearest Neighbour and K-Means based on Term Re-weighting for Classify Indonesian News," Int. J. Comput. Appl., vol. 50, no. 11, pp. 37–42, Jul. 2012.

```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


# 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))
```

Fig. 1. Python Code of KNN

```
# 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))
```

Fig. 2. Python Code of KNN

```
# 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. Python Code of KNN

```
    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)

# 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
```

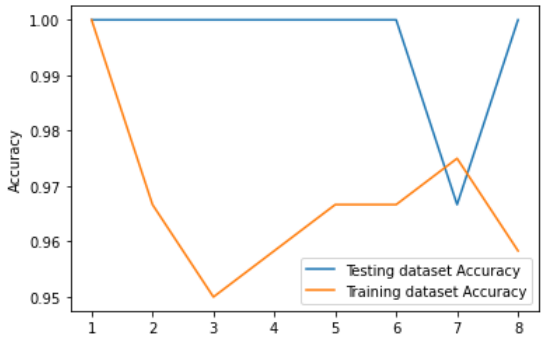

Fig. 4. Python Code of KNN and Output