Decision Tree Classifier on Wines Dataset

Sai Manogyana Tokachichu
Department of Computer Science
George Mason University
Fairfax, USA
stokachi@gmu.edu

***Abstract*—This paper looks at the implementation of the supervised machine learning technique, Decision Tree Classification, for classifying wines into three different kinds based on thirteen different features, trying to achieve at least 80% accuracy.**

***Index Terms*—supervised learning, decision trees, entropy, gini index, information gain, accuracy**

I. Introduction

Decision Tree Classification, which is one of the techniques of supervised learning techniques, was implemented and used for classification on the wines data set. This paper outlines the Decision Tree Classification method. Then, it presents the details of the implementation along with the observations made for different parameters such as test size, minimum number of samples per split, maximum depth of the tree and the data distribution of the training data in relation to the accuracy achieved.

II. Background

A. Decision Tree Classifier

Classification using decision trees is a supervised machine learning technique. Supervised machine learning technique is the one in which labelled data is fed to the model to make it learn the rule of classification.

Decision Tree Classifiers have two types of nodes: Decision nodes and leaf nodes

Decision Nodes consist of an attribute based on which the samples are split. The attribute which goes into a decision node is chosen based on "information gain" it produces i.e., how much impurity in the data it can reduce. The attribute with the highest information gain is chosen as the decision node at any particular node.

Information gain is based on entropy, which measures the impurity present in the data.

$$Entropy = H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1}$$

$$InformationGain = IG(S, A) = H(S) - \sum_{v \in V} \frac{|S_v|}{|S|} H(S_v) \tag{2}$$

Alternatively, to reduce the computation costs, gini index can be used instead of entropy and it'd still give the same result.

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3}$$

Leaf Nodes are the labels of the dataset. Once a label is decided for a node, the node cannot be split further.

B. Wines Dataset

Wines dataset has 13 features and can be classified into 3 different labels. All 13 features were continuous-valued numbers. There are a total of 30 samples in the given dataset. However, the classification to be performed was multiclass classification. Multi–Class Classification has more than two mutually exclusive classes, where the goal is to predict to which class a given input belongs to.

III. Proposed Approach

This implementation of decision tree classification provides the option to use entropy or gini index. It used gini index to reduce computational expenses.

In this implementation of Decision Tree Classifier, two parameters were used. They are minimum number of samples per split and maximum depth of the tree. There are other parameters that can be used, but this implementation limited the parameters of the model to these two.

Minimum number of samples per split defines the threshold of samples that need to be present in a node for it to be split further. If the number of samples per node is less than the minimum number of samples per split, then the node will not be split any further. Generally, this value is set to 2 in scikit-learn module.

Maximum depth of the tree indicates to how much depth the tree can be built. If this is set to a higher value, then there could be overfitting [2] i.e., the model may learn only the training dataset too well and its accuracy would be too high on the training set but it'd perform poorly on other datasets. If the value is set to a lower number, there could be underfitting and the model may not learn sufficient rules at all. For smaller datasets, which is the case here, generally the value is set lower, otherwise the model will learn to adapt completely to the given dataset only.

These parameters have to be experimented with various values to check which ones are giving best accuracy but mentioned above are the general expectations.

In addition, since the size of the data set is 30, test size and random state values had to be chosen such that they

represent the underlying data distribution accurately, which will ultimately lead to improved accuracy.

For this implementation, different values of test size were tried i.e., 0.20, 0.25, 0.30. Random state values were also tried varying from [0,42], which are the recommended values. Random state values produces different distributions of training data and the one that represents the underlying nuances in data produces best accuracy.

IV. Experimental Results

For the experiment, the following parameters were set to the indicated values through all the 43 runs of the experiment:

test size = 0.3

max depth = 5

min samples per split = 2

Then, the model was trained with different values for random state ranging from [0,42], which are the usually recommended values [1]. Different values of random state generate different distributions of training data. It is expected that the distribution which represents the original distribution most accurately is one of the ways to improve model accuracy.

The following table, TABLE I, shows the number of instances of labels (1s, 2s, 3s) each random state generated of training data and its relation to accuracy.

V. Conclusions

It can be observed that the model predicted with an accuracy of 100.0% when random state is 6. But we do not take into consideration this model as it may lead to overfitting of data on a newer, unseen dataset.

In addition, it can also be observed that the very first time the highest accuracy that is less than 100.0%, which is 88.8% was achieved, is when the random state is 7 and the data distribution in this instance of training data has more or less equal representation of different labels (here, 1s, 2s, 3s).

Furthermore, it should also be taken into consideration that though some random state values generated such training data with equal number of instances for all three labels, the model's accuracy was low. This could be due to different factors such as the model learning too well the training data (overfitting) or there needs to be feature engineering performed. But it can also be observed that the models with skewed training data performed poorly in terms of accuracy.

TABLE I
Data Distribution - Accuracy

Random State	Data Distribution	Accuracy
0	1s: 9, 2s: 6, 3s: 6	55.5%
1	1s: 9, 2s: 6, 3s: 6	77.7%
2	1s: 5, 2s: 8, 3s: 8	77.7%
3	1s: 8, 2s: 5, 3s: 8	66.6%
4	1s: 10, 2s: 6, 3s: 5	44.4%
5	1s: 10, 2s: 6, 3s: 5	66.6%
6	1s: 6, 2s: 8, 3s: 7	100.0%
7	1s: 6, 2s: 7, 3s: 8	88.8%
8	1s: 6, 2s: 8, 3s: 7	66.6%
9	1s: 5, 2s: 7, 3s: 9	55.5%
10	1s: 6, 2s: 8, 3s: 7	100.0%
11	1s: 6, 2s: 7, 3s: 8	77.7%
12	1s: 7, 2s: 7, 3s: 7	66.6%
13	1s: 7, 2s: 8, 3s: 6	44.4%
14	1s: 6, 2s: 8, 3s: 7	77.7%
15	1s: 6, 2s: 8, 3s: 7	33.3%
16	1s: 6, 2s: 7, 3s: 8	88.8%
17	1s: 8, 2s: 7, 3s: 6	55.5%
18	1s: 5, 2s: 8, 3s: 8	66.6%
19	1s: 7, 2s: 7, 3s: 7	77.7%
20	1s: 8, 2s: 6, 3s: 7	66.6%
21	1s: 7, 2s: 9, 3s: 5	66.6%
22	1s: 7, 2s: 8, 3s: 6	66.6%
23	1s: 7, 2s: 7, 3s: 7	66.6%
24	1s: 7, 2s: 7, 3s: 7	77.7%
25	1s: 7, 2s: 6, 3s: 8	77.7%
26	1s: 7, 2s: 9, 3s: 5	88.8%
27	1s: 8, 2s: 7, 3s: 6	55.5%
28	1s: 7, 2s: 6, 3s: 8	66.6%
29	1s: 9, 2s: 6, 3s: 6	66.6%
30	1s: 7, 2s: 8, 3s: 6	77.7%
31	1s: 5, 2s: 8, 3s: 8	66.6%
32	1s: 8, 2s: 6, 3s: 7	55.5%
33	1s: 7, 2s: 7, 3s: 7	88.8%
34	1s: 7, 2s: 6, 3s: 8	77.7%
35	1s: 7, 2s: 6, 3s: 8	88.8%
36	1s: 9, 2s: 6, 3s: 6	66.6%
37	1s: 3, 2s: 9, 3s: 9	77.7%
38	1s: 6, 2s: 9, 3s: 6	66.6%
39	1s: 7, 2s: 7, 3s: 7	66.6%
40	1s: 8, 2s: 5, 3s: 8	66.6%
41	1s: 8, 2s: 7, 3s: 6	77.7%
42	1s: 8, 2s: 7, 3s: 6	77.7%

References

[1] Kishan Modasiya, "What is random_state?" on medium.com, Jun 25, 2022, https://medium.com/mlearning-ai/what-the-heck-is-random-state-24a7a8389f3d

[2] R.Stuart, N. Peter, Artificial Intelligence A Modern Approach, 4th ed., 2023, p. 651-654