A Mathematical Essay on Decision Trees

Swathi V
Dept. of Engineering Design
Indian Institute of Technology Madras
Chennai, India
ed18b034@smail.iitm.ac.in

***Abstract*—Decision trees are strong predictive models used for regression and classification purposes. In this article, we explore the mathematics behind building a decision tree, and their application in a multi-class classification problem. We also discuss various metrics to quantify their performance, and how it can be improved.**

***Index Terms*—Decision tree, Recursive binary split, Gini index, Cross-entropy, Tree Pruning, Car evaluation**

I. Introduction

Decision trees are a family of supervised machine learning algorithms, which can be used for both classification and regression problems. In this report, we will be looking at decision trees in the scope of their application to classification problems.

Decision tree algorithms can be used for classification on datasets consisting of both categorical and continuous variables. These variables are used to segment the input space into a number of regions, using a set of rules. These rules then form an inverted tree-like structure, which can be used to make predictions on new data points.

In this report, we try to use decision trees to build a predictive model on the car evaluation dataset, which contains 6 categorical input features. The target variable, or the variable to be predicted, is the class that the car belongs to: *unacc*, *acc*, *good*, *vgood*.

We start by analysing the mathematics behind building a decision tree. We then move on to building a model to predict the target variable on the car evaluation dataset, and examine the performance using various metrics.

II. Decision Trees - A Mathematical Overview

A classification tree is used to predict a qualitative response (such as Yes/No, 0/1, Apple/Mango/Orange), by splitting the input space into various regions based on a set of rules. While using the model to make predictions on a new observation, we take the *majority vote*, or the most commonly occurring class of training observations in that region. While interpreting the results of a classification tree, we are interested not only in the majority class, but also in what proportion of training data points in that region belong to the majority class. This helps us understand the uncertainty involved in our prediction.

A. Understanding a Decision Tree

Let a classification problem be posed, with input vectors $x_i = [x_1\ x_2\ \cdots\ x_m]$ and target variables y_i. A decision tree can be used to divide the $m-$dimensional input space into various regions. For example, let all the observations with $x_2 > 0$ belong to class 1, and out of those with $x_2 \leq 0$, those with $x_1 > 10$ belong to class 1, and the rest belong to class 2. Then the decision tree would be as shown in Fig. 1.

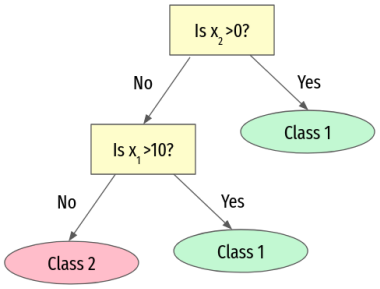

Fig. 1. An example decision tree

In the following sections, we try to answer three major questions about a decision tree:

- What features to use for splitting? What are the associated conditions?
- When to stop splitting a node further?
- How to prevent overfitting?

B. Growing a Tree

The method used for to grow a decision tree is the *recursive binary split*, which is a *top-down*, *greedy* approach. It begins at the top of the tree, where all observations belong to a single region, and successively splits the input space. Each split is represented by two new branches further down on the tree. It is called a greedy algorithm because at each step of the tree-building process, the best split is made at that step only, and not taking into account what will lead to a better tree in future steps.

In order to decide the best split, we need a measure that tells us the proportion of training observations that fall in a particular class after that split. Two measures are commonly used for this purpose, the *Gini index* and *cross-entropy*.

1) Gini Index:: The Gini index is defined as

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \tag{1}$$

where K is the number of classes, and p_{mk} is the proportion of training observations in the m^{th} region that belong to the

k^{th} class. The Gini index is a measure of the total variance across the K classes. The Gini index is called a measure of *purity*, as a small value indicates that most observations are from a single class.

2) Cross-Entropy:: The cross-entropy measure is given by:

$$D = -\sum_{k=1}^{K} p_{mk} \log(p_{mk}) \tag{2}$$

D will take on a value close to 0 if all the class proportions are near zero or one. Similar to the Gini index, a small value of cross-entropy shows a *pure* node.

We repeat this process at each of the subsequent nodes, looking for the best split to minimize either the Gini index or cross-entropy with each of the resulting regions.

C. Stopping Criteria

This process of splitting the nodes continues until a stopping criterion is reached. Two most commonly used stopping criteria are:

- Stopping when the number of observations in every node is not more than a set number.
- Stopping when the depth of the tree is not less than a set number.

D. Tree Pruning

The process of building a decision tree is likely to overfit to the training data, leading to a low bias but a high variance between datasets. A smaller tree (with fewer splits), on the other hand, might lead to lower variance but a high bias. A common method used to overcome this issue is *tree pruning*, i.e., building a large tree T_0 and then pruning it back to obtain a smaller *subtree*. In order to determine the best way to prune a tree, cross-validation or the validation set approach is used.

However, estimating the validation error for every possible subtree would lead to high computation costs. Hence, *cost complexity pruning* is used to select a small set of subtrees which will then be considered for cross-validation. In this approach, only subtrees indexed by a non-negative tuning parameter α will be considered. For each value of α, there exists a subtree T such that:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - y_{R_m})^2 + \alpha|T| \tag{3}$$

Here, $|T|$ indicates the number of terminal nodes of the tree T, R_m is the region corresponding to the m^{th} terminal node, and y_{R_m} is the predicted response associated with R_m. The tuning parameter α is a tradeoff between the complexity and the better fit to training data. When $\alpha = 0$, the subtree T will be equal to T_0. As α increases, the value to be paid for each node will be higher, and this will lead to a smaller subtree.

Out of these subtrees, the one that minimizes the average error during k-fold cross-validation will be chosen as the best model.

E. Using the Decision Tree to Make Predictions

Once the tree is built, for any new test observation, we start from the top of the tree and narrow down one region where the observation belongs. Once in this region, we take the *majority vote*, or the class label corresponding to the maximum number of data points in that region, to be the prediction.

III. Data

The data use for this task is the *Car Evaluation Model*, derived from a hierarchical decision model. The various input features and their definitions are given in Table I. The *target variable*, or the variable to be predicted is the class that the car belongs to: unacc, acc, good, vgood.

TABLE I
VARIOUS INPUT FEATURES THAT ARE PRESENT IN THE DATASETS

Variable	Type and Description	Key (if applicable)
buying	Buying Price: Categorical	vhigh, high, med, low
maint	Price of Maintenance: Categorical	vhigh, high, med, low
doors	Number of Doors: Categorical	2, 3, 4, 5more
persons	Capacity: Categorical	2, 4, more
lug_boot	Size of the luggage boot: Categorical	small, med, big
safety	Estimated safety of the car: Categorical	low, med, high

A. Preliminary Data Analytics

The dataset consists of **1727** data points, with no missing values. The distribution of data points for each input feature is given in Fig. 2.

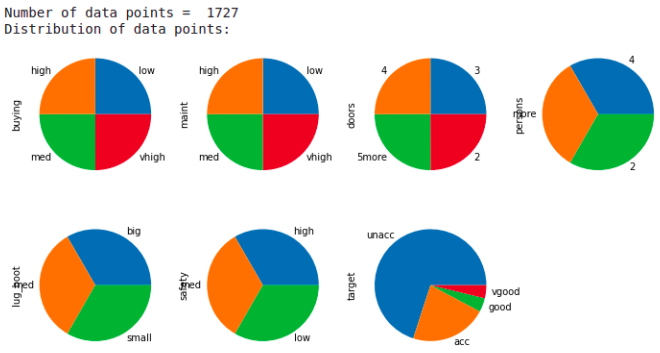

Fig. 2. Distribution of the original data according to each feature

IV. The Problem

For the given dataset, we try to predict the target variable using the input features given. The steps for the same are:

- Data cleaning
- Data analysis and visualization
- Feature Engineering
- Building a decision tree for classification
- Pruning the decision tree and selecting the best model
- Using the best model to make predictions on the test data

A. Data Cleaning

As mentioned above, there are no missing fields in the given dataset. The dataset was divided into a train and test dataset (70% and 30% respectively). However, since the distribution of the target variable is very skewed, methods for augmentation of the training dataset were explored. Common methods include *random over-sampling* and *synthetic minority over-sampling technique (SMOTE)*. Since random over-sampling only creates duplicates of existing data points, we use SMOTE in the following implementation. The distribution of input features after SMOTE is shown in Fig. 3. After over-sampling, we see that the number of data points has increased significantly. The target variable classes are also evenly distributed, but this has led to uneven distribution of other features.

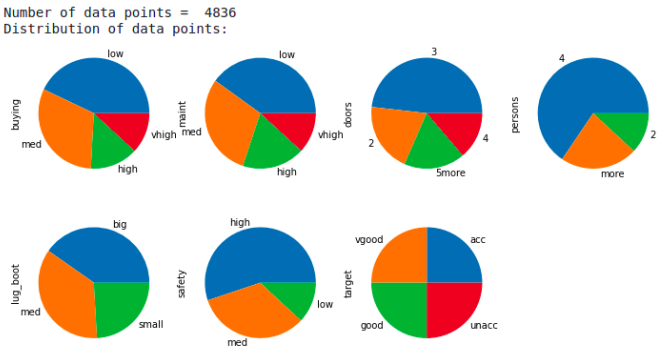

Fig. 3. Distribution of the over-sampled dataset according to each feature

B. Data Visualisation and Analysis

For each input feature, the visualization of the target variable w.r.t. the input feature is shown in Fig. 4 to Fig. 9. This is shown for both the original and augmented datasets.

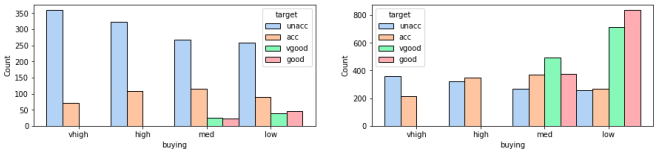

Fig. 4. Dependence of target variable on buying price

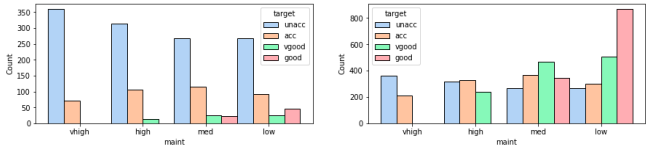

Fig. 5. Dependence of target variable on maintenance price

The information for the augmented and original datasets are plotted together, so as to verify the correctness of the augmentation technique. We observe that if for a particular value of input feature, there are no cars belonging to a label in the original dataset, the same can be said about the augmented dataset as well. Thus, we do not induce any systematic bias in the system because of the augmentation. As seen from the above figures, it is not possible to fit a simple decision boundary for the classification required, which motivates the use of decision trees.

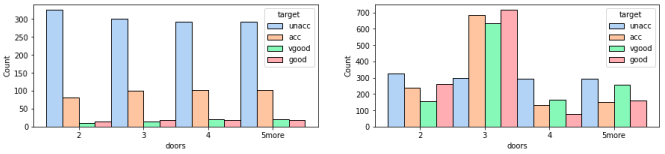

Fig. 6. Dependence of target variable on number of doors

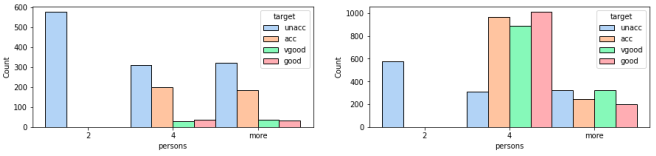

Fig. 7. Dependence of target variable on capacity

C. Feature Engineering

As all the variables are categorical, we will encode them to numerical class labels. This is done for all features, including the target variable. The assigned class labels are given in Fig. 10.

D. Building a Decision Tree

While building a decision tree, the major parameter to tune is the maximum depth of the tree, and the split criterion. Fig. 11 shows the common classification metrics obtained for each tree depth, for both split criteria.

We observe that both training and test performances saturate after a maximum depth of around 10. Though there is a gap between the training and test performances, it does not lead to overfitting as the test performance doesn't reduce.

It is also clear that both Gini and cross-entropy models perform very similarly, as expected from their mathematical definitions.

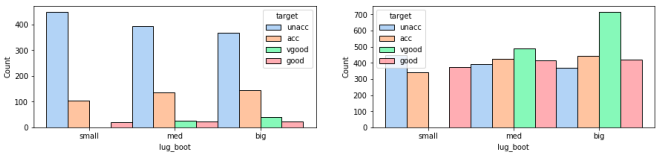

Fig. 8. Dependence of target variable on luggage space

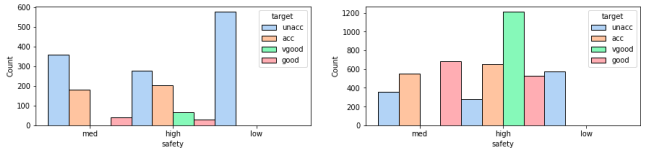

Fig. 9. Dependence of target variable on safety rating

```
buying :
        0 : high;       1 : low;        2 : med;        3 : vhigh;
maint :
        0 : high;       1 : low;        2 : med;        3 : vhigh;
doors :
        0 : 2;  1 : 3;  2 : 4;  3 : 5more;
persons :
        0 : 2;  1 : 4;  2 : more;
lug_boot :
        0 : big;        1 : med;        2 : small;
safety :
        0 : high;       1 : low;        2 : med;
target :
        0 : acc;        1 : good;       2 : unacc;      3 : vgood;
```

Fig. 10. Assigned class labels after feature engineering

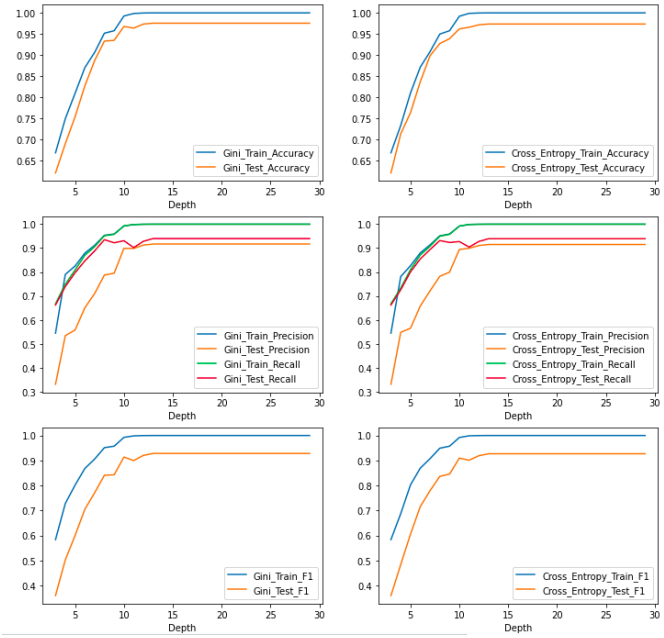

Fig. 11. Accuracy, Precision, Recall and F1 score for various tree depths

Thus, for further steps (pruning), we use the decision tree with maximum depth = 15, and splitting criteria as the Gini index. The performance metrics obtained are summarized in Fig. 12.

```
Training:
Accuracy =  0.9994089834515366
Precision =  0.9994089834515366
Recall =  0.9994103773584906
F1 Score =  0.9994089826257658
Test:
Accuracy =  0.976878612716763
Precision =  0.9524705092886911
Recall =  0.9366198752228163
F1 Score =  0.944325462028887
```

Fig. 12. Performance metrics of the selected model

Fig. 13 and Fig. 14 show a visual representation of the decision tree towards the root and leaf nodes. We observe that most of the leaf nodes are pure, i.e., have a single class. This can lead to overfitting in most cases, but in this case, most of the leaf nodes contain around 5 data points. This shows that there is no overfitting, and instead shows that the input space can be split effectively.

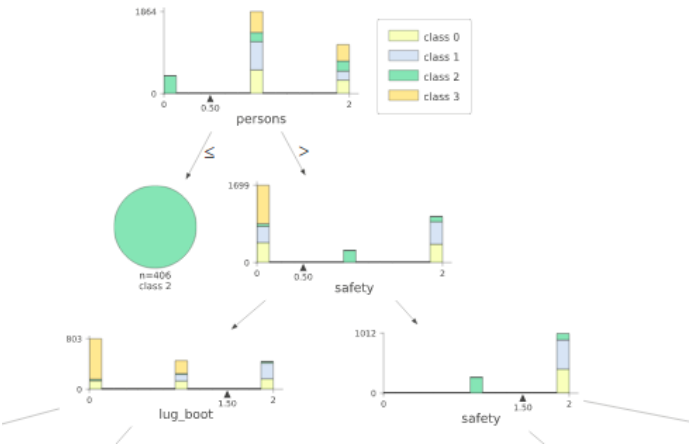

Fig. 13. A visual representation of the tree towards the root node

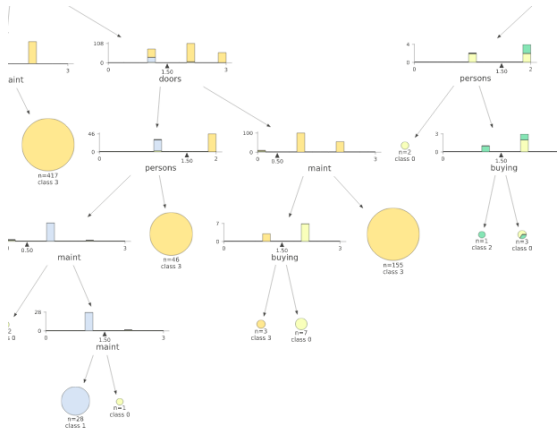

Fig. 14. A visual representation of the tree towards the leaf nodes

E. Post-Pruning the Decision Tree and Selecting the Best Model

As mentioned above, the model with maximum depth 15 was chosen to be the optimum. For various values of α, the total leaf nodes impurity is calculated, which is plotted in Fig. 15. As expected, as the value of α increases, the impurity in the leaf nodes increases. Next, we train a decision tree using the effective alphas. Fig. 16 shows the variation of depth and number of leaf nodes as a function of α. Here as well, the trend is as expected, since α is a hyperparameter that reduces overfitting. We now test the training and test accuracies for various values of α. The results are shown in Fig. 17. Here, the expected trend for the test data is that the test accuracy first increases and then decreases. However, the results obtained show that the test accuracy varies along with the training accuracy.

Hence, it is evident that all the data points are very similar, making the test and training dataset very similar. Thus, we finally select the model with no post-pruning, the performance

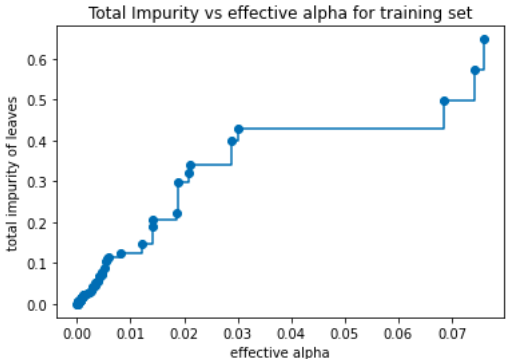

Fig. 15. Impurity in leaf nodes for various values of α

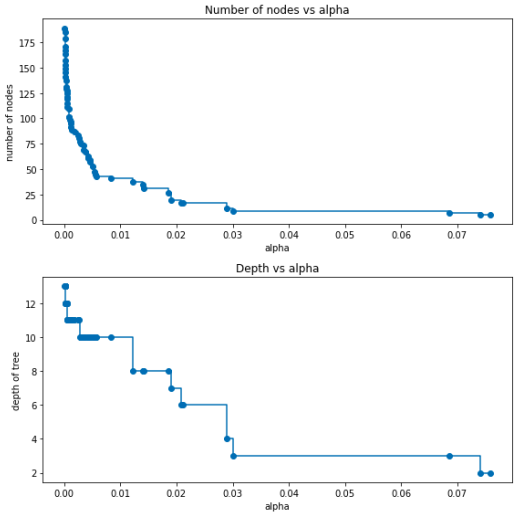

Fig. 16. Depth of the tree and number of leaf nodes for various values of α

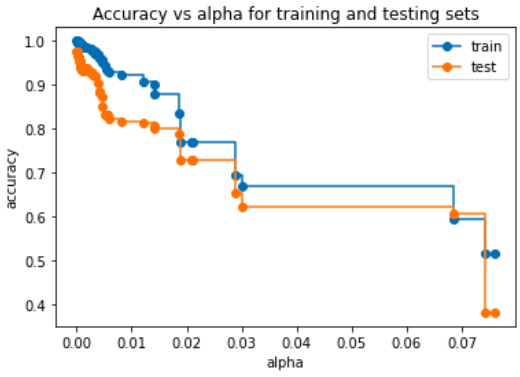

Fig. 17. Training and test accuracies for various values of α

metrics of which are given in Fig. 11. The confusion matrix of the same is shown in Fig. 18, and the classification report is shown in Fig. 19. We see that there is no systematic bias towards any class in particular, as all classes have fairly similar metric values.

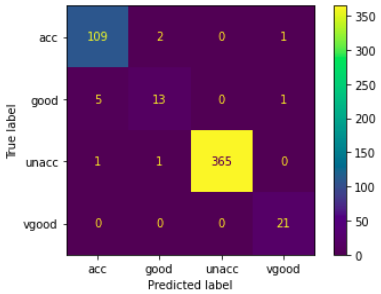

Fig. 18. Confusion matrix for the final model

	precision	recall	f1-score	support
acc	0.95	0.97	0.96	112
good	0.81	0.68	0.74	19
unacc	1.00	0.99	1.00	367
vgood	0.91	1.00	0.95	21
accuracy			0.98	519
macro avg	0.92	0.91	0.91	519
weighted avg	0.98	0.98	0.98	519

Fig. 19. Classification report for the final model

V. Conclusions

In this report, we analysed the mathematical formulation behind a decision tree model, and addressed issues such as stopping criteria and prevention of overfitting. The decision tree algorithm has built a good predictive model, which generalises well to the train and test data even without post-pruning. The performance has been analysed in terms of metrics such as accuracy, precision, recall, and f1-score, for each target class.

A. Avenues for Future Work

Bagging and boosting algorithms can be used to improve the performance of the decision tree algorithm, and ensure that it generalizes better to new data points.

References

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, "Tree-Based Methods" in *An Introduction to Statistical Learning,* New York, Springer, 2013.

[2] towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052