A mathematical essay on Decision Trees

Assignment 4 : EE4708 Data Analytics Laboratory

Madhur Jindal
Inter Disciplinary Dual Degree Program in Data Science
Indian Institute of Technology (IIT) Madras
Chennai, India
me18b059@smail.iitm.ac.in

***Abstract*—This document is a mathematical essay on Decision Tree methods wherein it is applied on a dataset containing different attributes of a car and the key task is to classify it as unacceptable, acceptable, good or very good , from their buying price, cost of maintenance, number of doors, capacity in terms of persons to carry, size of the luggage boot, estimated safety of the car, and determine whether some characteristics of the car are more likely to make it a good choice.**

I. Introduction

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks. They are very powerful algorithms, capable of fitting complex datasets. Decision Trees split the instances into two or more homogeneous sets based on most significant splitter / differentiator in input variables. Decision Trees are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today. It uses a tree like structure and their possible combinations to solve a particular problem. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

II. Decision Trees

A. Classification and Regression Trees (CART)

Nowadays, Decision Tree algorithm is known by its modern name CART which stands for Classification and Regression Trees. Classification and Regression Trees or CART is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification and regression modeling problems.

The CART algorithm provides a foundation for other important algorithms like bagged decision trees, random forest and boosted decision trees. In this kernel, I will solve a classification problem. So, I will refer the algorithm also as Decision Tree Classification problem.

Identify applicable funding agency here. If none, delete this.

B. Terminology

In a Decision Tree algorithm, there is a tree like structure in which each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from the root node to leaf node represent classification rules.

We can see that there is some terminology involved in Decision Tree algorithm. The terms involved in Decision Tree algorithm are as follows:

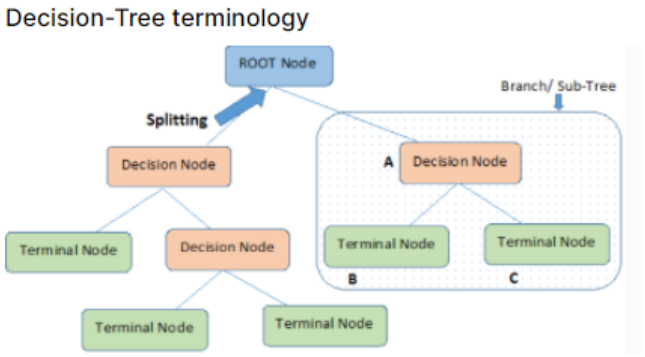

Fig. 1. Basic Decision Tree Terminology

1) Root Node: It represents the entire population or sample. This further gets divided into two or more homogeneous sets.

2) Splitting: It is a process of dividing a node into two or more sub-nodes.

3) Decision Node: When a sub-node splits into further sub-nodes, then it is called a decision node.

4) Leaf/Terminal Node: Nodes that do not split are called Leaf or Terminal nodes.

5) Pruning: When we remove sub-nodes of a decision node, this process is called pruning. It is the opposite process of splitting.

6) Branch/Sub-Tree: A sub-section of an entire tree is called a branch or sub-tree.

7) Parent and Child Node: A node, which is divided into sub-nodes is called the parent node of sub-nodes where sub-nodes are the children of a parent node.

C. Intuition

The Decision-Tree algorithm is one of the most frequently and widely used supervised machine learning algorithms that can be used for both classification and regression tasks. The intuition behind the Decision-Tree algorithm is very simple to understand.

The Decision Tree algorithm intuition is as follows:-

For each attribute in the dataset, the Decision-Tree algorithm forms a node. The most important attribute is placed at the root node.

For evaluating the task in hand, we start at the root node and we work our way down the tree by following the corresponding node that meets our condition or decision.

This process continues until a leaf node is reached. It contains the prediction or the outcome of the Decision Tree.

D. Attribute selection measures

The primary challenge in the Decision Tree implementation is to identify the attributes which we consider as the root node and each level. This process is known as the attributes selection. There are different attributes selection measure to identify the attribute which can be considered as the root node at each level.

There are 2 popular attribute selection measures. They are as follows:-

1. Information gain
2. Gini index

While using Information gain as a criterion, we assume attributes to be categorical and for Gini index attributes are assumed to be continuous. These attribute selection measures are described below.

1) Information gain: By using information gain as a criterion, we try to estimate the information contained by each attribute. To understand the concept of Information Gain, we need to know another concept called Entropy.

2) Entropy: Entropy measures the impurity in the given dataset. In Physics and Mathematics, entropy is referred to as the randomness or uncertainty of a random variable X. In information theory, it refers to the impurity in a group of examples. Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.

Entropy is represented by the following formula:

$$\text{Entropy} = \sum_{i=1}^{C} -p_i * \log_2 (pi)$$

Here, c is the number of classes and pi is the probability associated with the ith class.

The ID3 (Iterative Dichotomiser) Decision Tree algorithm uses entropy to calculate information gain. So, by calculating decrease in entropy measure of each attribute we can calculate their information gain. The attribute with the highest information gain is chosen as the splitting attribute at the node.

3) Gini Index: Another attribute selection measure that CART (Categorical and Regression Trees) uses is the Gini index. It uses the Gini method to create split points.

Gini index can be represented with the following formula:

$$\text{Gini} = 1 - \sum_{i=1}^{C} (p_i)^2$$

Here, again c is the number of classes and pi is the probability associated with the ith class.

Gini index says, if we randomly select two items from a population, they must be of the same class and probability for this is 1 if the population is pure.

It works with the categorical target variable "Success" or "Failure". It performs only binary splits. The higher the value of Gini, higher the homogeneity. CART (Classification and Regression Tree) uses the Gini method to create binary splits.

Steps to Calculate Gini for a split

1. Calculate Gini for sub-nodes, using formula sum of the square of probability for success and failure $(p^2 + q^2)$.

2. Calculate Gini for split using weighted Gini score of each node of that split.

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point. The attribute with minimum Gini index is chosen as the splitting attribute.

E. Random Forest

Bootstrap aggregating, also called bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach. Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

F. XGBoost

Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest.It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular

data.XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. XGBoost works as Newton Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order taylor's approximation is used in the loss function to make the connection to Newton Raphson method.

G. Overfitting in Decision Tree algorithm

Overfitting is a practical problem while building a Decision-Tree model. The problem of overfitting is considered when the algorithm continues to go deeper and deeper to reduce the training-set error but results with an increased test-set error. So, accuracy of prediction for our model goes down. It generally happens when we build many branches due to outliers and irregularities in data.

Two approaches which can be used to avoid overfitting are as follows:-

1) Pre-Pruning: In pre-pruning, we stop the tree construction a bit early. We prefer not to split a node if its goodness measure is below a threshold value. But it is difficult to choose an appropriate stopping point.

2) Post-Pruning: In post-pruning, we go deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use the cross-validation data to check the effect of our pruning. Using cross-validation data, we test whether expanding a node will result in improve or not. If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded. So, the node should be converted to a leaf node.

H. Accuracy Metrics

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

we can assign the event row as "positive" and the no-event row as "negative". We can then assign the event column of predictions as "true" and the no-event as "false".

This gives us:

"true positive" for correctly predicted event values. "false positive" for incorrectly predicted event values. "true negative" for correctly predicted no-event values. "false negative" for incorrectly predicted no-event values.

1) Precision: Precision can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true and false positives (TP + FP).

So, Precision identifies the proportion of correctly predicted positive outcome. It is more concerned with the positive class than the negative class.

Mathematically, precision can be defined as the ratio of TP to (TP + FP).

2) Recall: Recall can be defined as the percentage of correctly predicted positive outcomes out of all the actual positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true positives and false negatives (TP + FN). Recall is also called Sensitivity.

Recall identifies the proportion of correctly predicted actual positives.

Mathematically, recall can be given as the ratio of TP to (TP + FN).

3) F1-Score: f1-score is the weighted harmonic mean of precision and recall. The best possible f1-score would be 1.0 and the worst would be 0.0. f1-score is the harmonic mean of precision and recall. So, f1-score is always lower than accuracy measures as they embed precision and recall into their computation. The weighted average of f1-score should be used to compare classifier models, not global accuracy.

4) Support: Support is the actual number of occurrences of the class in our dataset.

5) Null-Hypothesis and P-value: The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (¡ 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable.

Conversely, a larger (insignificant) p-value suggests that changes in the predictor are not associated with changes in the response.

III. Problem - Understanding and Modelling

We are presented with a problem wherein we are provided with certain characteristic features of different cars including buying price, cost of maintenance, number of doors, capacity in terms of persons to carry, size of the luggage boot, estimated safety of the car, and determine whether some characteristics of the car are more likely to make it a good choice.

A. Data Pre-processing

We start with reading the data to a pandas dataframe. We observe a total of 1728 data points, with 7 columns in total, including the different features related to the car. On visualizing the distributions of the target variable, we see a multi-class imbalanced dataset problem, wherein around 70% of the total cars are classified unacceptable, 22.2% just acceptable, 4% good and 3.8% very good (Figure 1). We see that all the 6 features are categorical and most certainly ordinal. We then move on to checking for the null values in

the data and find that we do not have any NaN values. We then go on to checking for any features with high cardinality, that is the number of different unique values each feature can take as a higher number can cause problems, but then find out that most of the features have 3/4 unique classes, and also most of the classes are balanced, hence we conclude that this is good clean data.

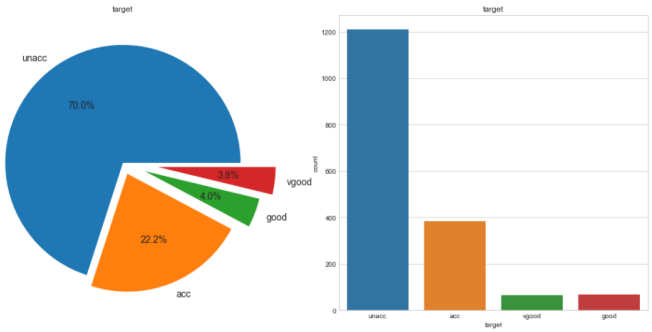

Fig. 2. Target Distribution

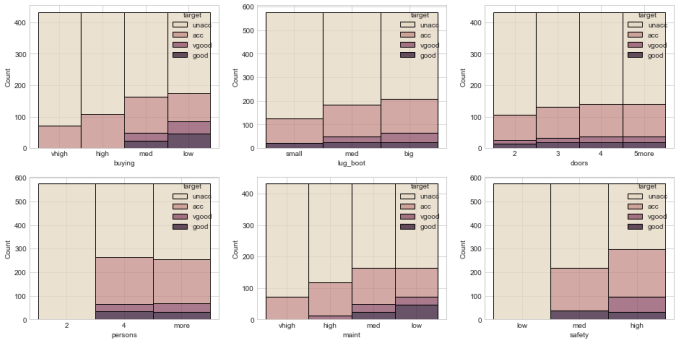

Fig. 3. Univariate Analysis

Fig. 4. Multivariate Analysis 1

B. Exploratory Data Analysis

We start with a univariate analysis, wherein we create histplots for each of the six features with hue as the target

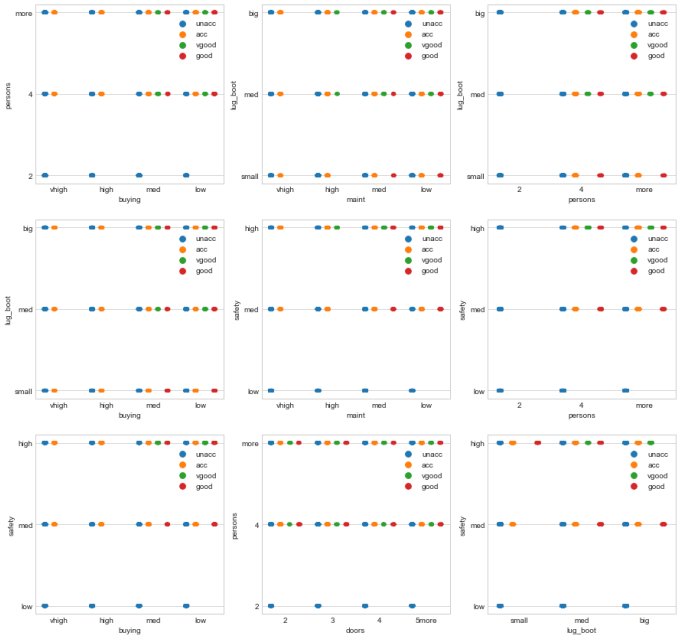

Fig. 5. Multivariate Analysis 2

column, using the seaborn library. We see that some of the classes in some features have certain target classes missing which is essentially very useful for our decision tree model as out model tries to get pure leaves. For e.g. considering buying, we see that high and very high buying costs only leads to unacceptable and acceptable cars; small luggage boot capacity leads to not being very good; 2 person cars are only seen as unacceptable, very high maintenance cars leads to unnaceptable and acceptable, while high maintenance costs lead to no very good classified cars. We see that low safety cars are unacceptable too.

Moving on to bivariate analysis, wherein we create stripplots for each of the 15 feature pairs with the hue as the target column, using the seaborn library, using jitter and dodge values as true. From the analysis we see a similar trend as the univariate analysis wherein some target classes are missing which is expected. I have provided the whole analysis for further observation.

C. Postprocessing

As the dataset has categorical features and hence they needs to be encoded in the appropriate form. There are two main method of encodig:

1. One hot encoding
2. Label encoding

As we have categorial features that are ordinal in nature i.e that can be ranked (ordered) hence label encoding will solve our purpose.Had there been nominal features we could have preferred one hot encoding. We use category encoders library for this.

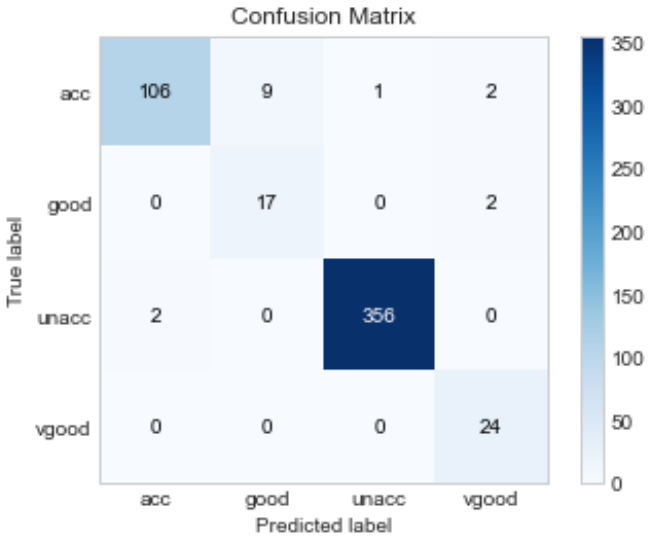

Fig. 6. Confusion Matrix for Decision Tree

Fig. 7. Visualization of the decision Tree (look into the codefile for bigger pic

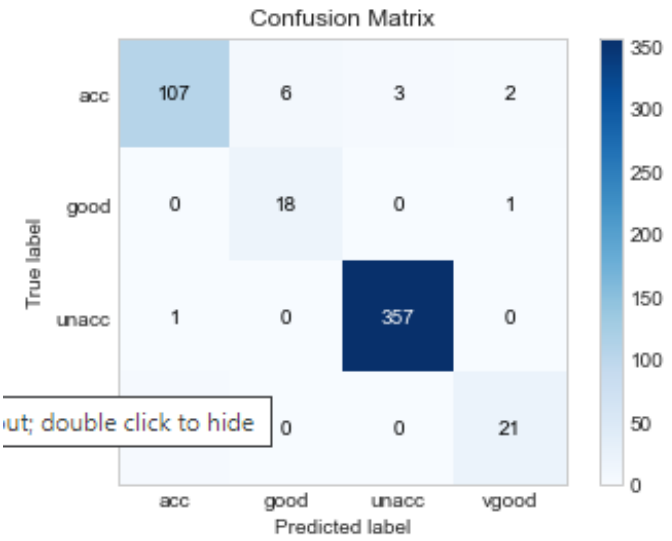

Fig. 8. Confusion Matrix for Random Forest

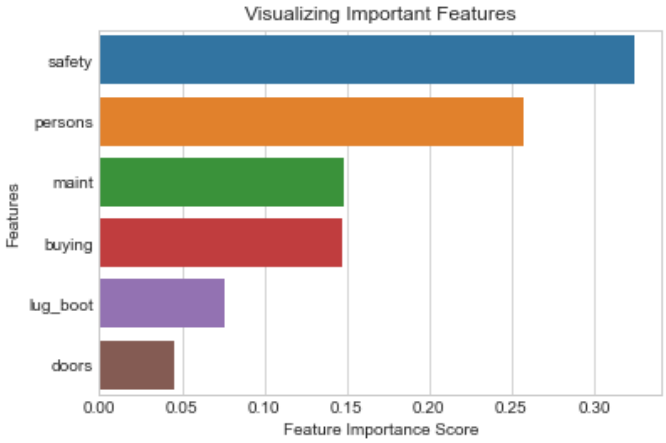

Fig. 9. Feature Importance chart (higher is better)

Following this we split the data using a 70/30 split, with the final dataset being 1209 examples in training set and 519 examples in crossvalidation set.

D. Decision Tree Modelling

In this paper, we model three versions of decision based classifiers namely Decision Trees, Random Forests and XG-Boost.

We start with modelling Decision tree classifier first. We use grid search for finding the best parameters four our decision tree model and checking for model classification accuracy on the cross validation dataset. Starting with class weight hyperparameter, we keep it as balanced as this automatically calculates the class weights according to their distribution in the train dataset. number of jobs is kept as -1 so that it chooses all the CPU cores available for fitting the model. The scoring metric used by us is accuracy. We use grid search for finding the best parameters by defining a range to check in. The parameters are Max depth, The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min samples split samples. Min samples split: The minimum number of samples required to split an internal node: If int, then consider min samples split as the minimum number. If float, then min samples split is a fraction and ceil(min samples split * n samples) are the minimum number of samples for each split. Min samples leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min samples leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. Finally we get the best

Fig. 10. Confusion Matrix for XGBoost

parameters as max depth 9, min samples leaf 2, min samples split 5. Using this we are able to achieve a train set accuracy of 0.98 and test accuracy of 0.969.

Then we move on to modelling using the Random Forest classifier which is a bagged version of Decision trees. In this case we use randomized search as this does not search the whole space but tries to get to the optimal using random sampled values of hyperparameters as this model is expensive to train. The hyperparameters being searched are: n estimators = number of trees in the forest, max features = max number of features considered for splitting a node, max depth = max number of levels in each decision tree, min samples split = min number of data points placed in a node before the node is split, min samples leaf = min number of data points allowed in a leaf node, bootstrap = method for sampling data points (with or without replacement).Fitting 3 folds for each of 100 candidates, totalling 300 fits we find the best parameters as 'n estimators': 800, 'min samples split': 3, 'min samples leaf': 1, 'max features': 'auto', 'max depth': None, 'bootstrap': False. Using this we are able to achieve a train set accuracy of 1 and test accuracy of 0.969. We then proceed to check for the feature importance as this is provided by the implementation of random forests in the sklearn library. We see that safety has the highest importance with doors having the least. We then train a model dropping the least important feature, but we observe reduced accuracy to 0.969 in train set and 0.933 in test set.

At last we take a look at the boosted algorithm very famously called as the XGBoost. XGBoost provides large range of hyperparameters. We can leverage the maximum power of XGBoost by tuning its hyperparameters. The most powerful ML algorithm like XGBoost is famous for picking up patterns and regularities in the data by automatically tuning thousands of learnable parameters. In this paper we use hyperopt library for finding the best hyperparameters some of which are max depth: The maximum depth of a tree, same as GBM. It is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample. Gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Reg alpha: L1 regularization term on weights (analogous to Lasso regression). It can be used in case of very high dimensionality so that the algorithm runs faster when implemented. Increasing this value will make model more conservative. Reg Lambda: L2 regularization term on weights (analogous to Ridge regression). Colsample bytree: is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. Min Child Weight: It defines the minimum sum of weights of all observations required in a child.This is similar to min child leaf in GBM but not exactly. This refers to min "sum of weights" of observations while GBM has min "number of observations". The best hyperparameters are 'colsample bytree': 0.959166117786024, 'gamma': 7.417890672096632, 'max depth': 13.0, 'merror': 6, 'min child weight': 8.0, 'reg alpha': 53.0, 'reg lambda': 0.73987405692127. Using these we get an accuracy value of 1 on the training set and 0.98 on the test set.

IV. Conclusion

Having done a thorough analysis of the tree based models, in the raw form, with bagging and with boosting, we see that all of the models perform well, but boosting based XGBoost is able to outperform the other models with an accuracy value that is 1.1 percent more on the test dataset. We see that these models have a significantly higher accuracy on the train set and thus are most likely to overfit, but using the three different hyperparmeter search methods - gridsearch, randomsearch and hyperopt we are able to find the hyperparameters with highest accuracy on the cross validation sets. We also see that the bagging based method Random Forest is not able to help us in getting better score on the test dataset. We see that tuning these hyperparameters let us get a granular level control on the trees being build to the ensemble being created.

As evident from our univariate and multivariate exploratory data analysis we see that some of the features have certain classes which do not contain all the target classes and also the distribution of the classes is mostly uniform which helps the tree based models to get pure leaves and thus get such high accuracy values. Some of such variables like buying with high and vhigh, small lug boot, very high and high maintenance, and low safety.Further avenues of growth can be checking for more features that could explain the target variable better.

References

[1] Decision Tree, Data Analytics Laboratory EE4708, July - November 2021
[2] Decision Trees by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[3] https://en.wikipedia.org/wiki/Gradient_boosting
[4] https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[6] https://www.kaggle.com/prashant111/a-guide-on-xgboost-hyperparameters-tuning
[7] https://www.kaggle.com/shubham47/random-forest-classifier-tutorial
[8] https://www.kaggle.com/xuchanyou/decision-tree-classifier-tutorial
[9] https://www.kaggle.com/mohitcr7/decision-tree-classifier-beginner-level
[10] https://www.kaggle.com/vipulgandhi/a-guide-to-decision-trees-for-beginners
[11] https://en.wikipedia.org/wiki/Decision_tree_pruning
[12] https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680