A mathematical essay on decision tree

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind decision tree classifier and then to use it in a real-life application. In this assignment, as a real-life application, decision tree classifier is used to formally identify the factors which could have been used to predict how acceptable a car is from the popular car evaluation dataset. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that it is possible to make reasonable predictions regarding the acceptability of a car using factors including but not limited to safety rating, price and luggage boot space. This is a reworked version of the original assignment with improvements to the Modelling section in the form of enhanced benchmarking of the model including a comparison with an XGBoost model. The plots have also been improved to enhance the clarity of presentation.**

***Index Terms*—decision tree, python, visualization, predictive modelling, multinomial classification**

I. Introduction

Given a set of features and a target variable, predictive modelling is typically used for generating a model which can make predictions for cases where we do not know the value of the target variable, i.e., only the features are available. Apart from this use, a model can also be used for developing an intuition of how various factors influence the target variable. In this assignment, we try to make use of a model for the purpose of identifying the key factors which influence the decision in a classification problem.

In particular, we make use of decision tree classifier for the purpose of identifying relationships in a classification problem. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.. Though they are by nature very simple, they can give good, and more importantly, interpretable results in many problem settings. Decision tree is a very flexible classifier in the sense that it can accommodate both continuous and categorical variables. In our particular problem, we will restrict ourselves to a classification problem using categorical variables.

In our problem setting, the goal is to use decision tree classifier to predict the acceptability level of a car given a variety of factors like price, number of doors, passenger capacity, luggage boot space and safety rating. We make use of the publicly available car evaluation dataset for building the model. After building the model, we evaluate the model using a variety of evaluation metrics. By examining how well the model performs, we can identify how good the identified relationships are. Also, we check how critical each feature is to the prediction and create a ranking based on the importance of features.

Section II gives an overview of the various techniques used for data cleaning, data visualization and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind decision tree. Section IV describes the various models that were tried and the results that were obtained by applying decision tree in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and data visualization. We also make some qualitative observations.

A. Preliminary analysis

The given dataset has 1728 rows and 7 columns. It must be noted that the dataset itself lacked any column headings: they had to be added in manually. A brief overview of the dataset is presented in Figure 1. We observe that we have **only categorical variables.** It also seems that there are no null values.

We also look at the distribution for our target class, which is the acceptability rating termed 'target'. On plotting, we obtain Figure 2. We have four categories: unacc (Unacceptable), acc (Acceptable), good (Good) and vgood (Very good). The count of cars in the unacceptable category is disproportionately high compared to other categories. We will have to account for this during model building.

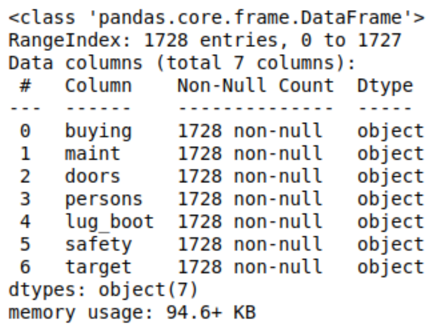

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   target    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Fig. 1. Summary of the dataset

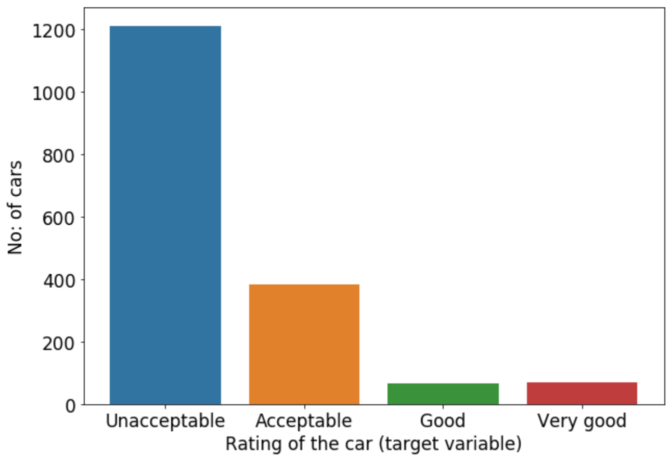


Fig. 2. Target class imbalance

B. *Feature by feature analysis*

Since all 6 features are categorical, for each feature we create a count plot, separating out the target classes. This can give qualitative insights regarding how a feature may affect the target variable.

Buying price

We obtain the count plot as shown in Figure 3. We make the following observations:

- Cars with high and very high buying price have a large proportion of unacceptable vehicles. Furthermore, there are no good or very good vehicles at this price range.
- Medium and low priced cars have representation from all four target categories.

Intuitively, this makes sense since people generally tend to find high prices unacceptable. For a deal to be considered good/ very good, it is almost always the case that the price should be on the lower end.

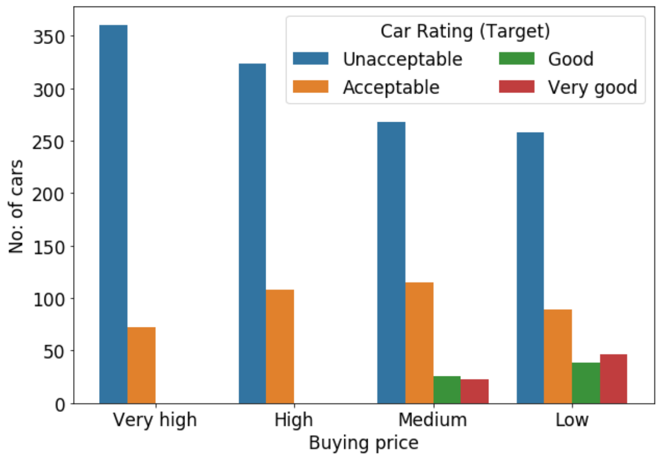


Fig. 3. Distribution of buying price

Maintenance price

We obtain the count plot as shown in Figure 4. We observe that the distribution is largely identical to the distribution for buying price. Again, we expect people to find high maintenance cost to be generally unacceptable.

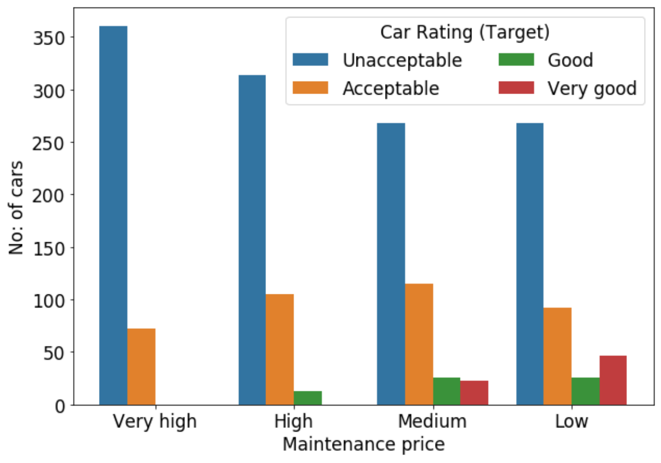


Fig. 4. Distribution of maintenance price

Number of doors

We obtain the count plot as shown in Figure 5. We observe that the distribution is largely identical for each category, i.e., irrespective of what the number of doors is, the distribution is same. In other words, the number of doors doesn't really provide any information regarding what would be the acceptability rating of a car. This implies that number of doors is not really a deciding factor as far as acceptance rating is concerned.

Passenger capacity

We obtain the count plot as shown in Figure 6. We make the following observations:

- Cars that can accommodate only two people are all unacceptable.
- Cars which can accommodate four and more than four have similar distributions.

This seems to suggest that while having at least 4 seats is essential, beyond that the number of seats doesn't really have an impact on the acceptability rating.

Size of luggage boot

We obtain the count plot as shown in Figure 7. We observe that though all categories have representation from all target categories, in general, the rating increases as the luggage boot size increases, which is what we would expect intuitively anyway.

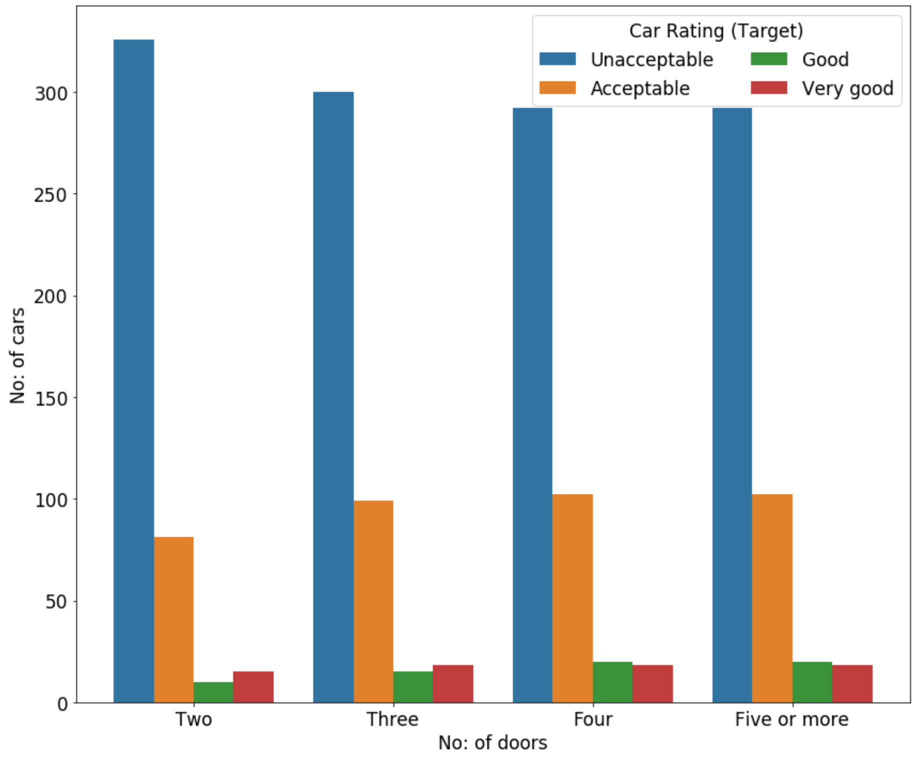


Fig. 5. Distribution of number of doors

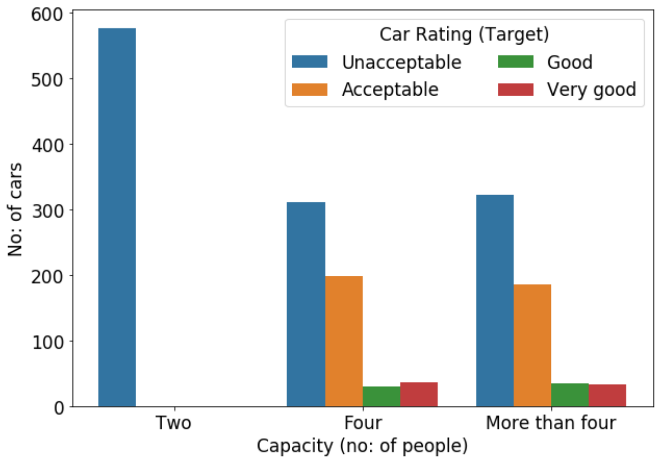


Fig. 6. Distribution of passenger capacity

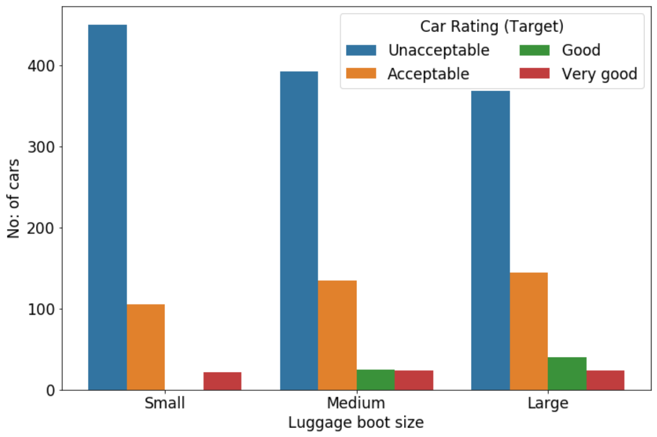


Fig. 7. Distribution of luggage boot size

Safety rating

We obtain the count plot as shown in Figure 8. We make the following observations:

- All low safety rated cars are unacceptable, which is what we would expect.
- As safety level increases, proportion of unacceptable cars decreases.

This seems to suggest that safety is a strong factor in predicting the acceptability of a car.

III. Model: Decision Tree Classifier

In this section, we will give a brief overview of the mathematical formalism behind the decision tree classifier.

In general, decision trees come under the broad umbrella of algorithms called CART algorithms: Classification and Regression Tree algorithms. Decision tree is a specific algorithm within this broad class. Decision trees themselves are of two kinds:

1) Classification trees: When the predicted outcome is the class (discrete) to which the data belongs.

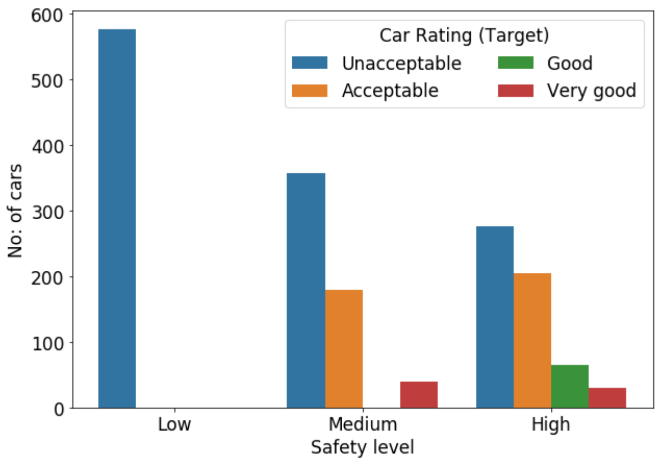


Fig. 8. Distribution of safety levels

2) Regression trees: When the predicted outcome can be considered a real number.

In this assignment, we will be making use of classification trees since the problem is a multi-label classification problem.

A. *Algorithm Overview*

Before describing the algorithm used by a decision tree classifier, it is necessary to make oneself familiar with the terminology and structure of a decision tree. A prototypical decision tree is shown in Figure 9. A brief overview of the terminology:

- Root node: Root node is from where the decision tree starts. It represents the entire dataset.
- Decision node: Decision node represents a node where a decision has to be made regarding a split to the dataset under current consideration.
- Leaf node: Leaf node represents the final output node. The tree cannot be segregated further after reaching a leaf node.

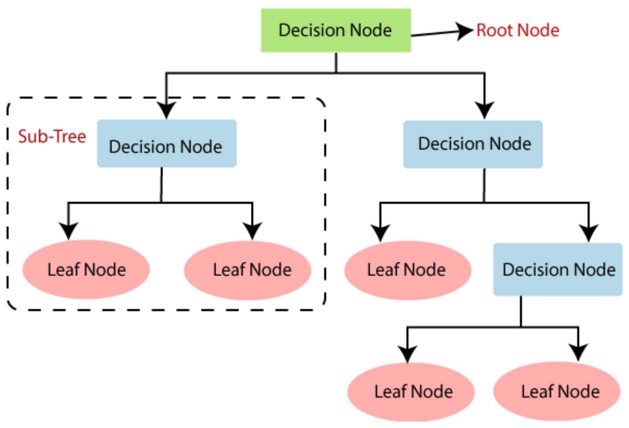


Fig. 9. Structure of a decision tree

A decision tree classifier makes used of a **rule-based classification algorithm** as opposed to a probabilistic model, which is adopted by algorithms like naive bayes. This means that rather than learning a number of parameters and then using this for predicting the probability of a sample belonging to a particular class, a set of learnt rules are used for directly predicting the class of a sample. In a sense, this rule-based approach mimics human decision making process.

The general structure of the algorithm used by a decision tree classifier is outlined below:

1) Begin the tree with the root node which contains the complete dataset.
2) Find the best attribute in the dataset using Attribute Selection Measure (ASM). More details regrading ASM will be discussed in the following section.
3) Split the dataset my making a decision rule based on an optimal value of the best attribute.
4) Recursively make new decision trees using the subsets of the dataset created in step 3. Continue this process until you reach a leaf node.

Once the tree has been built, classifying a new sample is as simple as traversing the tree till you reach a leaf node, which will provide the predicted class.

The major benefits offered by a decision tree classifier are:

- **Uses a white box/open-box model:** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model, the explanation for the results is typically difficult to understand, for example with an artificial neural network.
- **Non-parametric approach**: Makes no assumptions of the training data or prediction residuals; e.g., no distributional, independence, or constant variance assumptions.
- **Requires little data preparation**: Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables

Some limitations of this approach:

- **Trees are unstable:** A small change in the training data can result in a large change in the tree and consequently the final predictions.
- **Non-optimality**: Practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.

B. *Mathematical formalism*

What a decision tree essentially learns is a decision for each decision node. In particular, suppose a decision tree makes use of an impurity measure S (various measures for quality will be discussed). Suppose a node splits the dataset into two subsets left and right. Then the node computes the quantity:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where m_{left} is the number of samples in the left subset, m_{right} is the number of samples in the right subset, S_{left} is the impurity measure for the left subset and S_{right} is the impurity measure for the right subset. The attempt is to find the split which minimizes $J(k, t_k)$.

We can regard $J(k, t_k)$ as the cost function. The algorithm first splits the training set into 2 subsets using a single feature k and a threshold t_k . A pair (t_k, k) is searched that minimizes the cost function the best. This procedure is repeated, looking for the best predictor and the best cut-point in order to split the data further so as to minimise the cost function. This process continues till the set stopping criterion is reached or when every training sample is perfectly classified.

It is to be noted that without the intervention of appropriate stopping criterion or other means, the tree could grow very deep, leading to over-fitting. Decision Trees are also unstable, meaning trees learnt from different samples of a dataset are likely to differ in their structure compared to approaches such as logistic regression or naive bayes.

An overview of two common attribute selection measures/ impurity measures is given below:

1) Gini Index/Impurity: This is essentially a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p_{i,k}}(1 - \hat{p_{i,k}})$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. The Gini index takes on a small value if all of the $\hat{p}_{i,k}$'s are close to zero or one. So, small value implies that a node predominantly contains one class. And a zero value implies that the node is pure. Thus Gini index could also be viewed as a measure of impurity. The equation for Gini index can also be rewritten as:

$$G = 1 - \sum_{k=1}^{K} \hat{p_{i,k}^2}$$

2) Entropy: The formulation has a close resemblance with the entropy formulation in thermodynamics.

$$D = -\sum_{k=1}^{K} \hat{p_{i,k}} \log \hat{p_{i,k}}$$

where again K is the total number of classes and $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. As was the case with Gini index, a smaller value of entropy signifies higher purity and a larger value of entropy signifies impurity.

C. Regularization techniques

As discussed above, allowing a tree to grow without stopping condition will inevitable lead to over-fitting. To mitigate this, we generally apply regularization techniques so as to increase the ability of the mode to generalize. We effectively tune the low bias - high variance model such that we decrease the variance at the cost of slightly increasing the bias. Two common regularization techniques are discussed below:

1) **Cost-complexity pruning:** Rather than considering every possible sub-tree, we consider a sequence of trees indexed by non-negative tuning parameter α. For each α, these correspond to a sub-tree $T \in T_0$ such that

$$J(k, t_k) + \alpha|T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of a tree. Notice that if $\alpha = 0$, the sub-tree T is the original tree T_0 . We can select a value for α using a validation set or cross-validation.

2) **Bagging:** In bagging, we construct B trees using B bootstrapped training sets and average the resulting predictions. Bootstrapping is a technique to obtain datapoints using random sampling with replacement. Although the constructed trees may be deep and hence suffer from high variance individually, averaging reduces variance and results in a better model. This, in fact, is a precursor to Random forests which in addition to this, deploys random features during splitting . Bagging improves prediction by lowering the variance, but at the expense of loss of interpretability.

IV. Modelling

In this section, we discuss the application of the decision tree classifier to our problem.

We create and evaluate three models:

1) In Model 1, we let the decision tree grow without any constraints.
2) In Model 2, we perform pruning on the tree to avoid overfitting.
3) In Model 3, we perform bagging to avoid overfitting.

For each model, we take care to account for the inherent imbalance in the training dataset with respect to the target class. This is done by making use of appropriate wweights during the training of the decision tree.

We evaluate the models based on multiple metrics. A short description of the used metrics is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.
- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

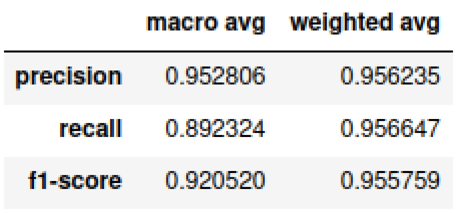
It must be noted that, strictly speaking, precision, recall and f1-score are defined only for binary classification. However,for multi-class classification, we can define these metrics for each class separately. Furthermore, we can take averages of these measures across all classes to compute a global metric. This can be done through the following two metrics:

- **Macro averaging:** In this method, a simple average is done without taking into account any class imbalance.
- **Weighted averaging:** In this method, averaging is done in a weighted manner after accounting for class imbalance using support, which is simply the number of instances belonging to a particular class according to true labels.

We will be using **weighted averaged f1-score** as the metric for comparison between models. The dataset is split into training, validation and test sets. The test set is kept aside and will be used only after choosing the best model. Comparison between models is done using validation set.

A. Model 1

In this model, we allow the tree to grow without any constraints whatsoever. We observe that such a tree achieves a **training accuracy of 100%!**. This is a clear indication that the model is **overfitting.** The performance of the model on validation set is shown in Figure 10. The table reports macro averages and weighted averages. The weighted averaged f1-score is **0.956.**



	macro avg	weighted avg
precision	0.952806	0.956235
recall	0.892324	0.956647
f1-score	0.920520	0.955759

Fig. 10. Metrics for model 1

B. Model 2

In this model, we perform pruning by restricting the depth. We take the maximum allowed depth as a hyperparameter and perform a grid search on possible values. The evolution of weighted f1-score on **validation set** with tree depth is shown in Figure 11. The highest score is obtained for a **depth of 9.** The performance of the model with a depth of 9 on validation set is shown in Figure 12. The weighted averaged f1-score is **0.968. Clearly the model outperforms Model 1.**

C. Model 3

In this model, we perform bagging of decision trees. Since we are taking an ensemble, we can allow each tree to grow in an unrestricted fashion. We train each tree on a random subset of the training data, with the subset size being half of the total dataset size. We take the number of trees as a hyperparameter and perform a grid search on possible values. The evolution of weighted f1-score on **validation set** with number of trees is shown in Figure 13. The highest score is obtained for a **tree count of 50.** The performance of the model with number of trees set to 50 on validation set is shown in Figure 14. The weighted averaged f1-score is **0.962. Clearly the model outperforms Model 1, but is not as good as Model 2.**

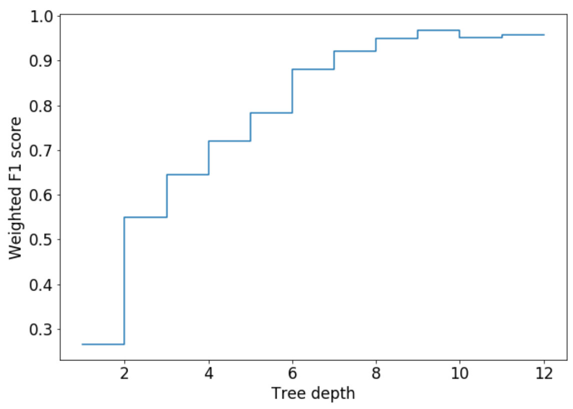
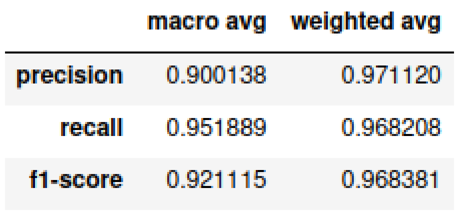


Fig. 11. Evolution of f1-score with tree depth



	macro avg	weighted avg
precision	0.900138	0.971120
recall	0.951889	0.968208
f1-score	0.921115	0.968381

Fig. 12. Metrics for model 2

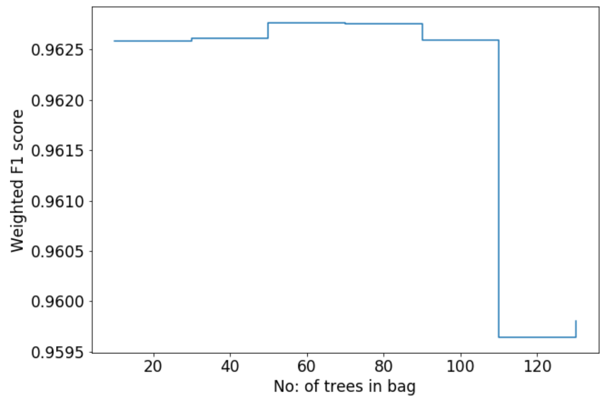
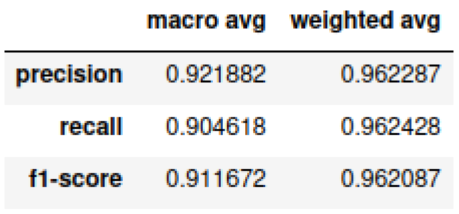


Fig. 13. Evolution of f1-score with number of trees

D. Best Model

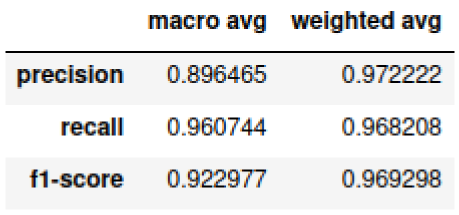
From the above analysis, we conclude that Model 2 performs the best. Hence, we train a decision tree of depth 9 on the whole dataset consisting of training set and the validation set. We then evaluate this on the test set. The metrics for this final model is given in Figure 15 and the confusion matrix is given in Figure 16.



	macro avg	weighted avg
precision	0.921882	0.962287
recall	0.904618	0.962428
f1-score	0.911672	0.962087

Fig. 14. Metrics for model 3

To get a sense of the importance of features, we also calculate the feature importance scores of each feature using an in-built method provided by scikit-learn. Essentially, the importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance. A plot showing the feature importance scores is given in Figure 17. **It can be seen that safety rating is the most important feature, whereas number of doors is the least important feature.** It is noteworthy that this agrees with our analysis during the exploratory phase.



	macro avg	weighted avg
precision	0.896465	0.972222
recall	0.960744	0.968208
f1-score	0.922977	0.969298

Fig. 15. Metrics for the best model

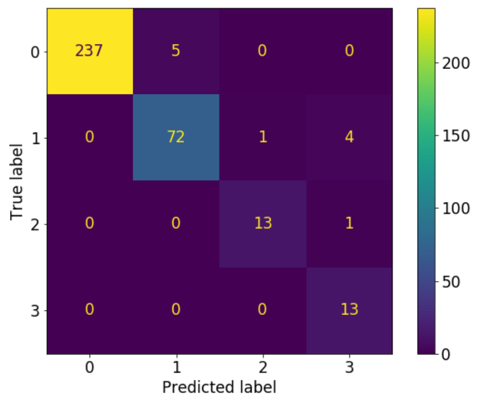


Fig. 16. Confusion matrix for the best model

Comparison with XGBoost

In almost any modern-day classification problem, the best results are often obtained by making use of boosting models. The most commonly used as well as most effective of the boosting techniques is XGBoost. By comparing the performance of our decision tree model with that of an XGBoost model, we can get a sort of benchmark on the performance.

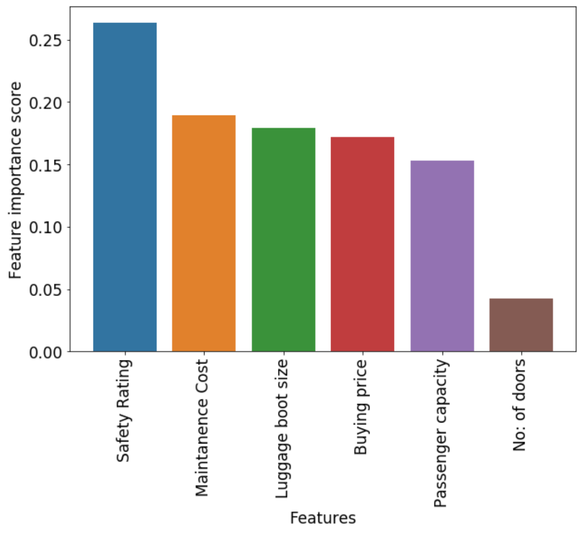
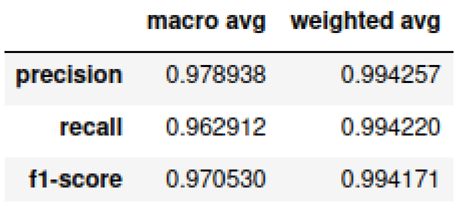


Fig. 17. Feature importance scores for the best model

XGBoost model was trained using a grid search on its hyperparameters. Cross-validation with three folds was performed for evaluating the performance. The values for the evaluation metrics and confusion matrix for the best XGBoost model are given in Figures 18 and 19 respectively. It can be seen that **the model clearly outperforms the best decision tree model.** This is expected since boosting is, by design, an ensemble of decision trees and hence superior to a single decision tree.



	macro avg	weighted avg
precision	0.978938	0.994257
recall	0.962912	0.994220
f1-score	0.970530	0.994171

Fig. 18. Metrics for XGBoost model

XGBoost also allows us to get the feature importance scores for the features. This is shown in Figure 20. Surprisingly, buying price seems to be the most important feature for this model. This feature is very closely followed by maintenance cost and safety rating. Rest of the features have an ordering similar to what we observed for the decision tree model. From the graph, it can also be seen that buying price, maintenance cost and safety rating all have very close feature importance scores. This highlights another important aspect of boosting models: **the classification decision is based on a number of features and not one feature alone. This helps enhance the robustness of the model.**

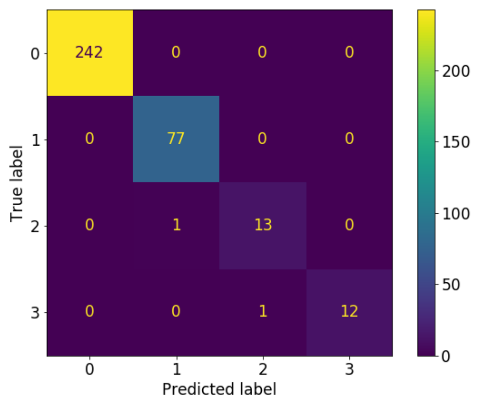


Fig. 19. Confusion matrix for XGBoost model

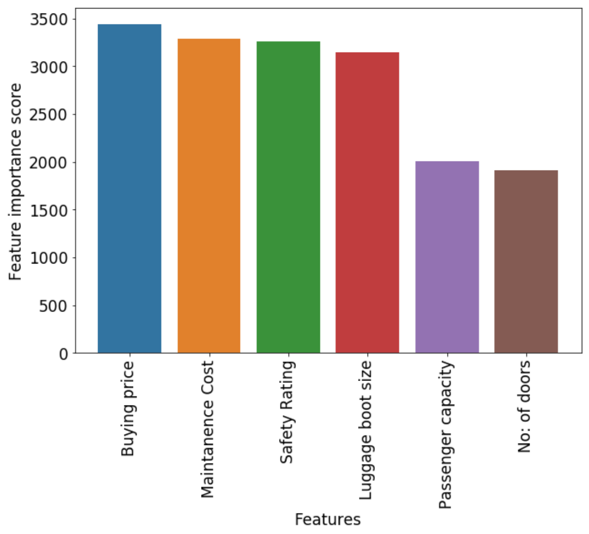


Fig. 20. Feature importance scores for XGBoost model

V. Conclusions

From our extensive analysis of the given dataset using decision tree classifier, we arrive at the following conclusions:

- Decision tree classifier, despite having a very simple structure, performs reasonably well in the classification task and mimics human decision making process.
- Decision trees are prone to overfitting. Hence, it is necessary to take remedial measures like pruning and bagging to overcome this issue.
- Accuracy may not be a good measure of performance for imbalanced classification tasks. It is necessary to use metrics like weighted f1-score for such cases.
- Best performance is obtained after using pruning and setting the maximum depth to 9.
- Safety rating is found to be the most important feature whereas number of doors is found to be the least important feature, as was also observed during the initial qualitative analysis.

VI. Avenues for Further Research

Decision classifier, despite working reasonably well, has a very simple structure which might not be sufficient in many cases. Using more sophisticated tree-based models like Random Forests might produce better results, though at the expense of interpretability. Doing some kind of feature preprocessing like PCA, to extract important features might also help.

References

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Decision tree classifier." https://en.wikipedia.org/wiki/Decision_tree.