CS60050: Machine Learning Mini Project 1

Group - 7

[PPDT] Customer Purchase Prediction using Decision Tree based Learning Model

Tanay Raghavendra (18EC10063), Ayan Chakraborty (18EC10075), Debjoy Saha (18EC30010)

Formulation of Problem Statement

1) Develop a Decision Tree Based machine learning solution to predict whether a customer makes a purchase decision based on several personal attributes as well as the expenditure profile of the customer.

2) Estimate the performance of our Model on Training, Validation and Test Data using k-fold cross validation and observe the effects of Reduced Order Pruning and other hyperparameters on the generated results.

3) Compare the results of our Decision Tree built from scratch and compare with already existing Implementations in the Scikit-learn package and also Generate a visualization of the Decision Tree Model for the purpose of interpretability.

Theory behind Decision Trees

Decision Trees is a classification algorithm which improves upon the limitations of the Find-S and the Candidate Elimination Algorithm. From the training data, Decision Trees can approximate Boolean Functions using different combinations of AND and OR statements. Unlike the Find-S and Candidate Elimination Algorithms, which can only form AND expressions, the Decision Tree can give a more General Boolean expression.

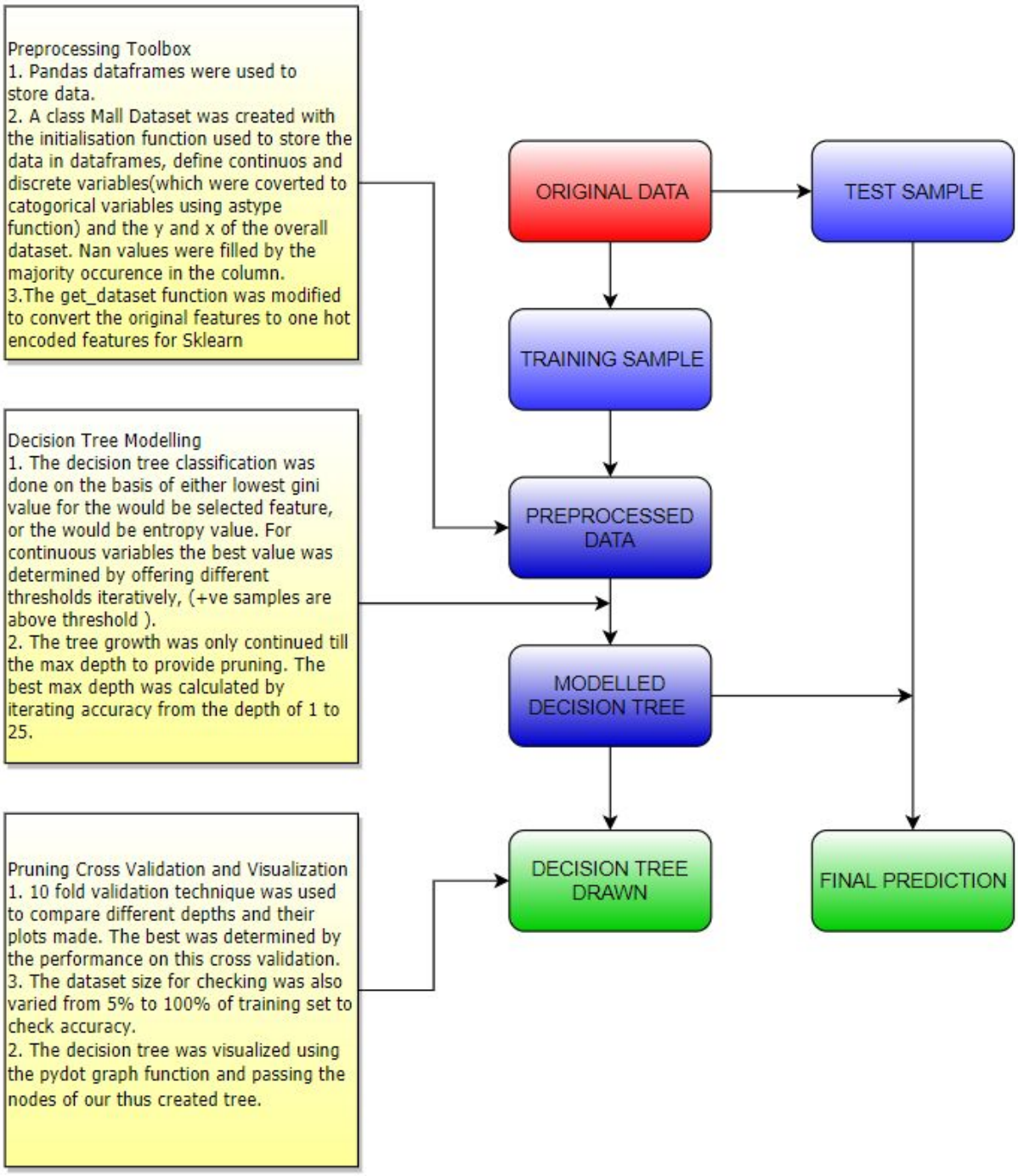
Each training example has a set of attributes and a corresponding result. The results belong to different classes. Our aim is to predict the class of the result given an unknown example. The Decision Tree tries to find a boolean relationship between the values of the attributes and the result. It first picks an attribute and then splits the data into two or more halves depending on the value of the attribute. It then repeats the process on each half of the data and continues, until it reaches a state where all the results belong to the same class. In this way, it generates a “tree” of nodes, where each node applies a boolean comparison on a single attribute.

The most important task in a decision tree is to choose which attributes to compare first. If it is not selected properly, then the number of comparisons to be made to generate a prediction will increase dramatically leading to more computational load and a larger memory to store the decision tree. Hence, a measure of Information Gain is used to determine the best classification possible at each stage. This is a greedy approach compared to whole solution space exploration. In practice, it works quite well. In this assignment, we focus on two main Information Gain Measures - Entropy gain and Gini Index. The Gini Index has a small advantage, that it is computationally less expensive.

Hence, when an unknown example arrives, it starts from the root of the generated tree and follows a path down to a specific leaf node by evaluating the boolean expressions on each intermediate node. The predicted class is the class associated with that leaf node.

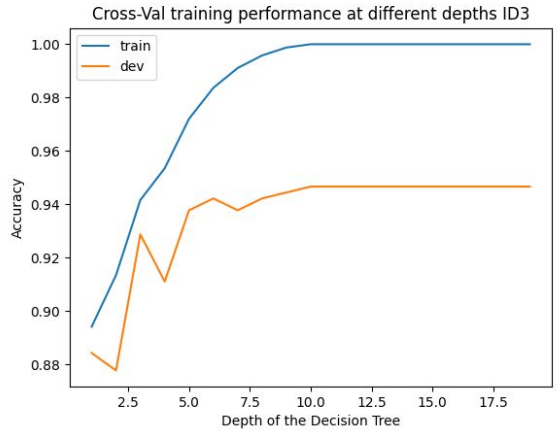
Decision Trees are widely used in many fields. One of the many advantages of the Decision Trees is its explainability. Its algorithm is similar to our decision making process and can be easily interpreted as a series of if-else statements. This is required in many fields such as Medicine where results have to be interpretable in order to be considered safe and risk-free.

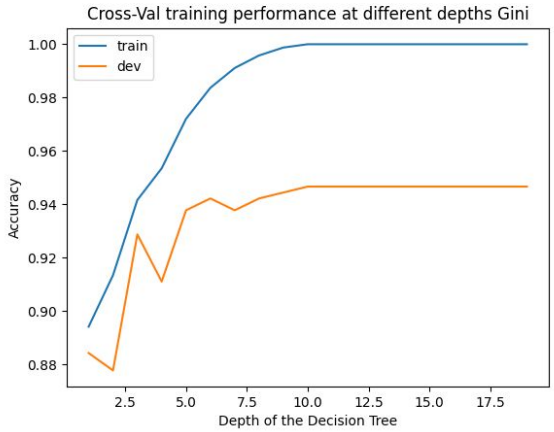
Explanation of Algorithms



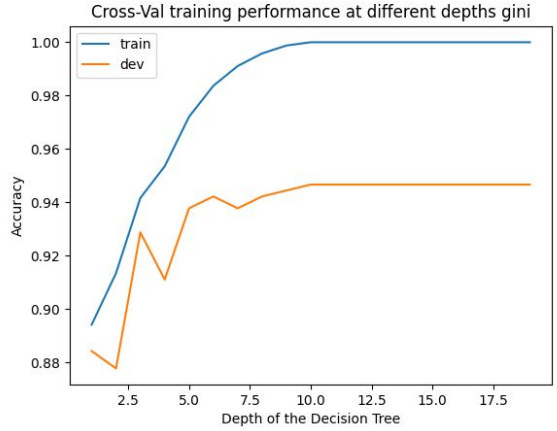
Plots of Results and Hyper-Parameter Tuning

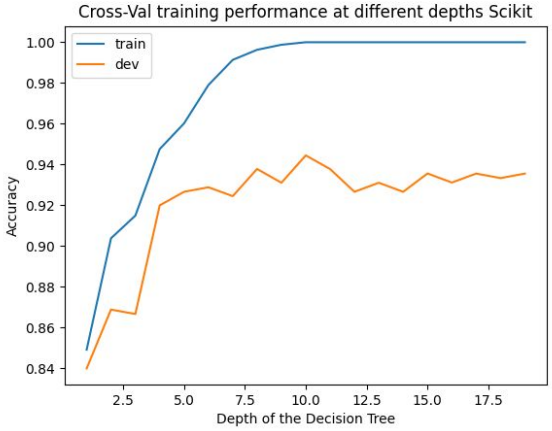
Comparison of Gini Index and Entropy as Information Gain Measures





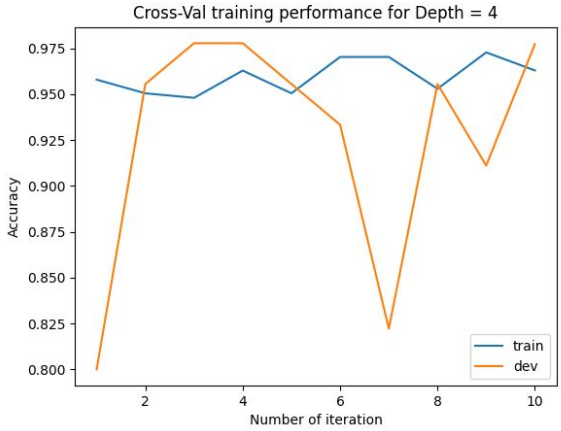
Comparison of Model built from Scratch (Using Gini) with Scikit Library Model



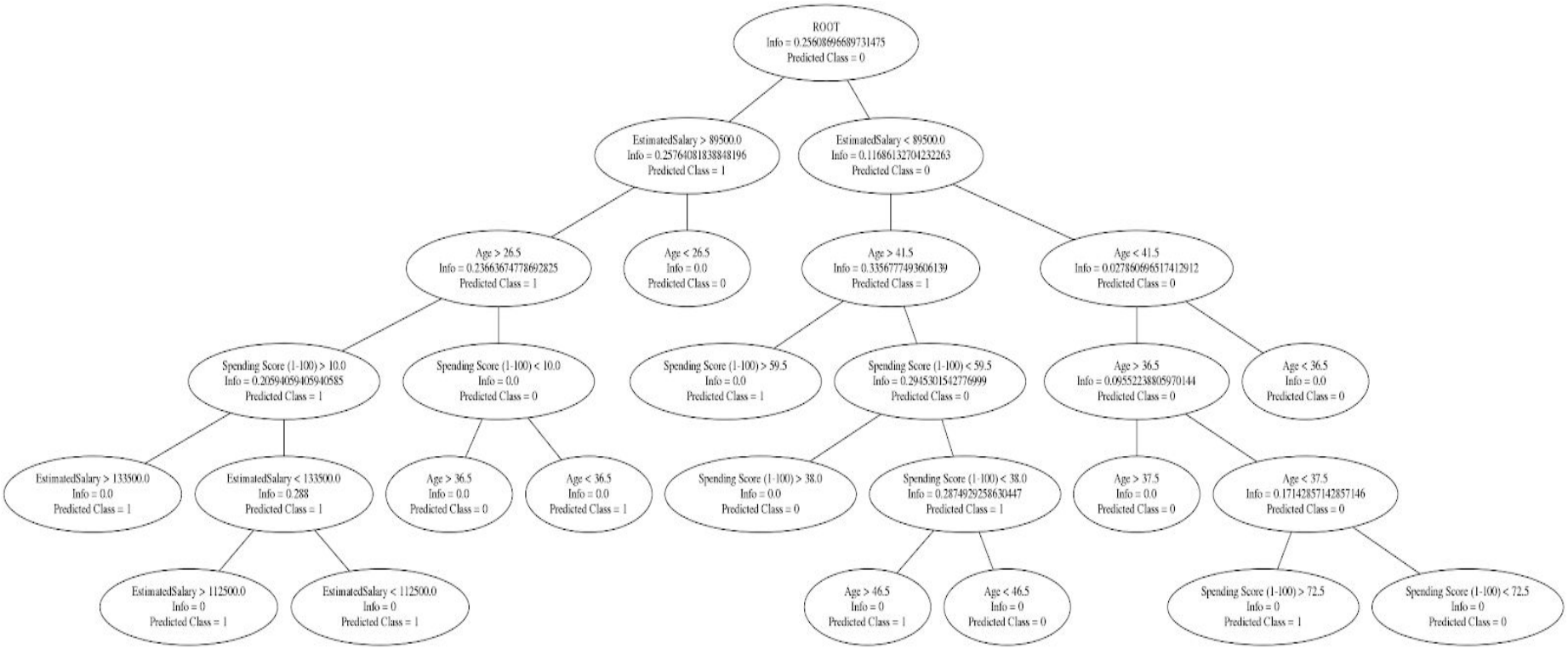


Comparison of Pruned (Depth = 4) and Non Pruned Models (Depth = 7) with respect to Training Data Size

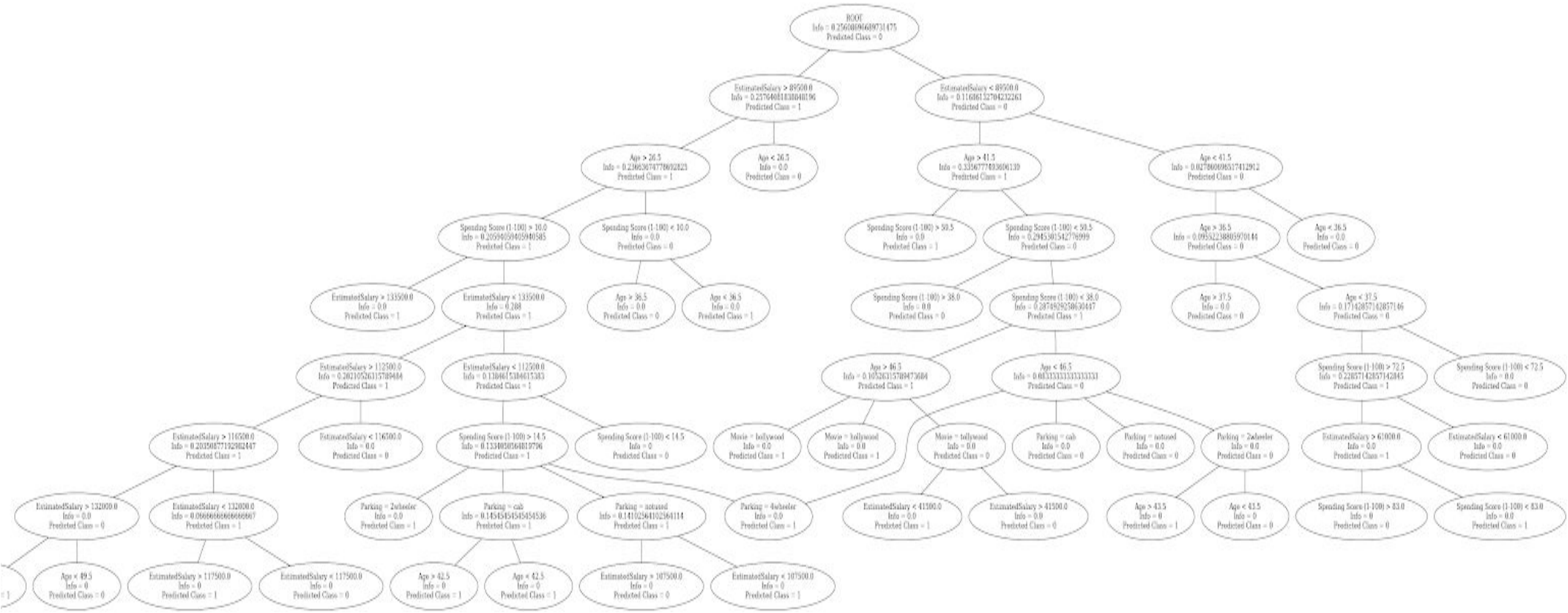
K- Fold (10 fold) Verification of Pruned (N = 4) and Non Pruned (N = 7) Models



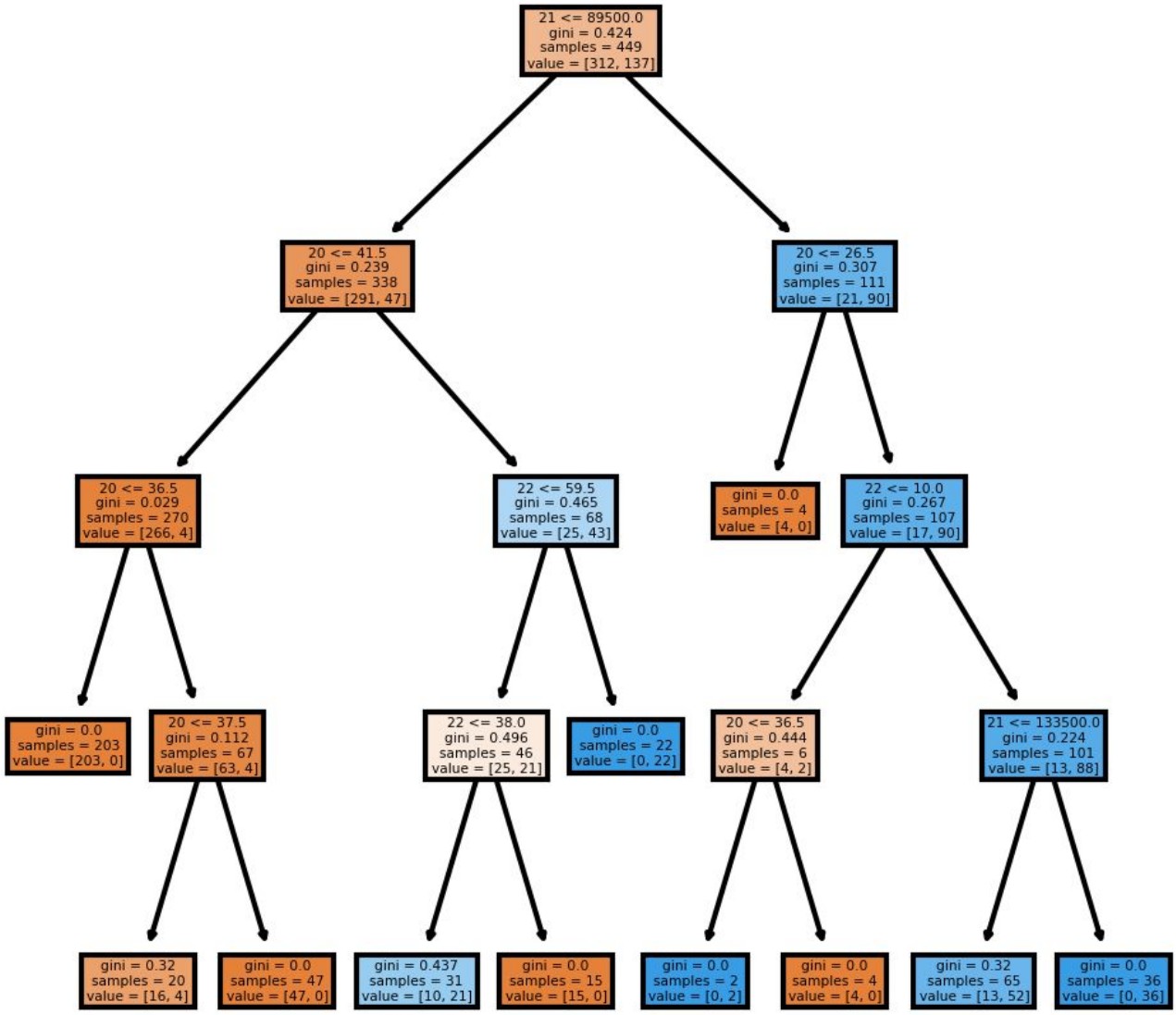
Visualization of the Pruned Best Decision Tree Generated



Visualization of the Non Pruned Best Decision Tree Generated



Visualization of the Best Performing Decision Tree generated by Scikit Library



Conclusion

1. Finding the maximum depth - We initially found the most optimum depth of the tree on the validation data by seeing the performance of the tree on the training set. It was found that 7 gave the same results as larger depth values and hence became the optimum choice. Finally after compiling the whole training data together, we ran this tree on depths lower than 7. 4 gave similar results to seven and was hence chosen to be the final depth.

2.Dataset Size - We compared the two depths of 4 and 7 with varying data set sizes . First they were just compared for the training set of different sizes. A peculiar behaviour was observed here as both the depth gave an accuracy of one at smaller sizes of data. This was just because of the high correlation the tree had with those data points. While comparing the depths on the validation set, we found similar behavior.

3. Why Sklearn underperforms - Sklearn cannot handle categorical variables, which means they need to be converted to one hot encoding form. This creates sparse vectors, as for one categorical variable with 4 categories, one hot would create 4 variables with 0 1 values. This leads to a bad decision tree. On a comparative, we are not handling such variables in a one hot way. We are making branches for each category of the variable. Hence, our performance is better.