Wine Classification Project

Vamsi Chinta

# I. DATASET DESCRIPTION

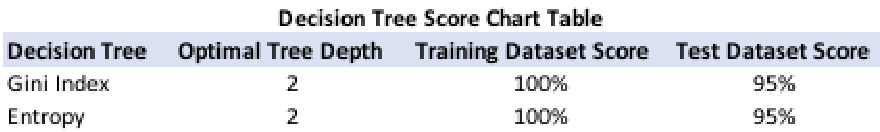
The Wine Dataset consists of 118 instances and 13 descriptive features, all of which have no missing values. Also, it is an ill-posed problem.

The target feature is categorical and consists of two levels: Chardonnay and Merlot. Both Target classes consist of 59 instances each and are, therefore, balanced.

# II. DECISION TREE ANALYSIS

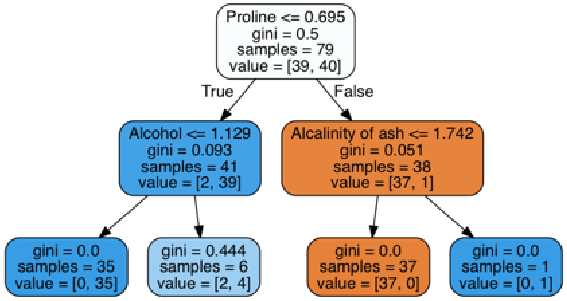
As per my analysis, the best decision tree classifier is the ID3 algorithm using the Gini Index. The reasoning for this conclusion is as follows.

The score charts for both the Gini Index and Entropy decision tree are equal for the optimal tree depth and the scores evaluated on both training and test datasets. For details on the procedures for this analysis please see ‘Decision Tree Procedures’ Section. (please refer to table below for tabulated values)



The Gini index is preferred, this is due to the fact that it requires less computational power. For our intents and purposes, given the size of the dataset being so small, either decision tree is arguably suffice for this application.

The decision tree using the Gini Index,

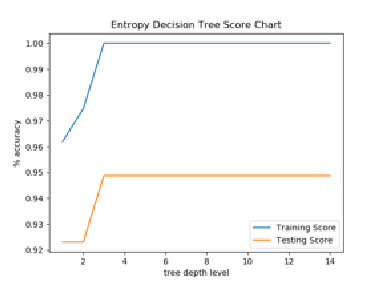


# III. DECISION TREE PROCEDURE

|  |
| --- |
| Faraz Mirza 002-42-5477 ©2018 IEEE |

The whole data set was normalized between the range 0 to 2 via the min max method. The data was then split into a training and testing dataset via Stratified holdout sampling in order to ensure an equal number of instances belonging to each class are present in each dataset. Two-thirds of the data was split into training dataset and the other third was split into the testing dataset.

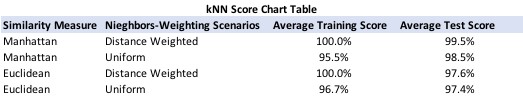
Score charts were developed for both methodologies across a range of max tree depth levels. The following are the score charts of decision tree classifier with Entropy and Gini criterion.



# IV. K NEAREST NEIGHBOR ANALYSIS

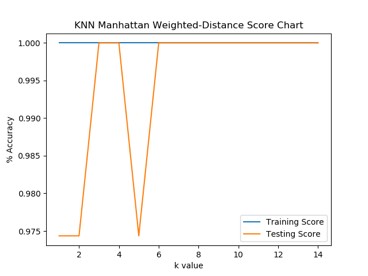
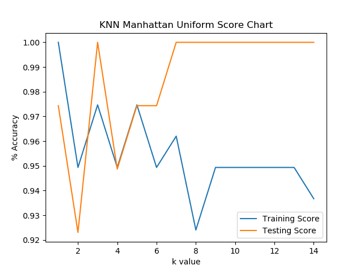
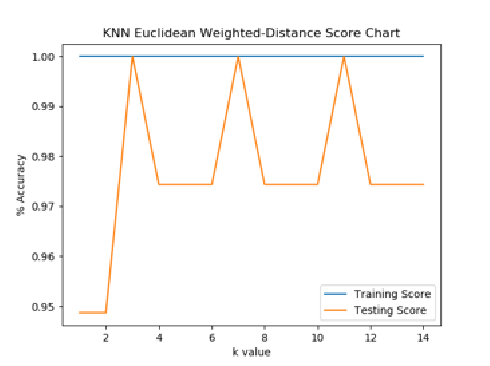
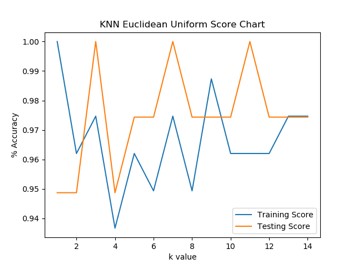
As per my analysis, the best classifier is kNN using the similarity measure, Manhattan. And the neighbor-weighting scenario, weighted-distance. The reasoning for this conclusion is as follows.

The score charts were evaluated across three key parameters: Similarity measures, neighbor-weighted scenarios, and k-values. The optimal kNN classifier stated above had the highest average score with the lowest standard deviation with respect to both the training dataset score and the testing dataset score. The following table summarizes the scores for each methodology investigated in ranked order.



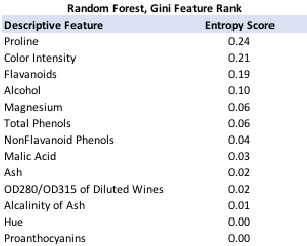
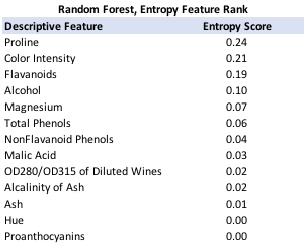
Given that our dataset only contains 118 instances which captures a very small percent of all the possible outcomes, the highest average test score can help us identify which method can generalize the data the best. The average test score is the average accuracy for that method across the range of k-values, 1 to 14. It is important to state here that key criterion to picking the best model is ‘Stability’ (the robustness of the model to generalize the dataset). One way to gage this is to see whether the performance of the metrics for both the training dataset and the test dataset preform similarly. As you can see in the score charts, this observation matches our conclusion.

V. K NEAREST NEIGHBOR PROCEDURE The same training and testing dataset was used for the kNN classifiers. The following are the score charts of kNN classifier with Manhattan/Euclidean distance metric and uniform weights/weighted distances.

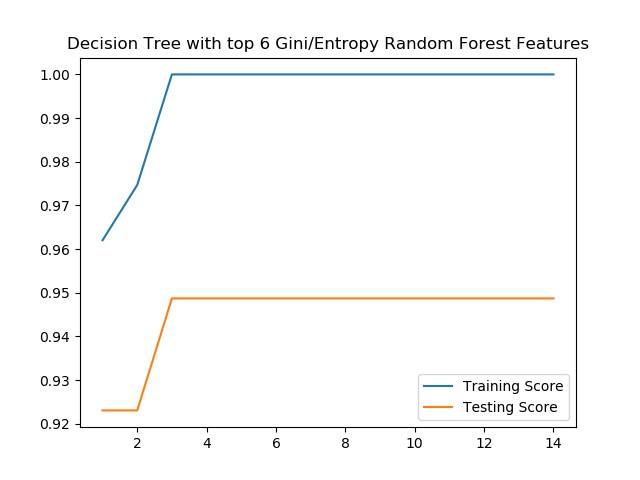


# VI. RANDOM FOREST

The following descriptive feature rankings were generated using the whole dataset via random forest.



One can use these rankings to our advantage by identifying which attributes are the most important relative to the target variable. Our data may contain descriptive features that are either redundant or irrelevant. In turn, certain features may be identified as an optimal test for a node but in fact, may have no relation to the target feature.

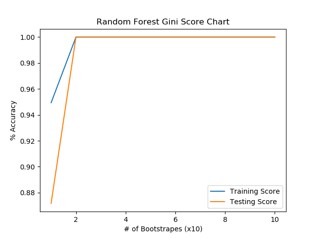
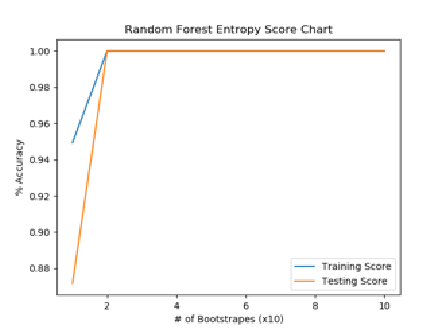


Therefore, the end resulting decision tree may end up with a high training dataset score. It may not be as accurate when evaluated across a new set of stances, the test dataset.

So forth, one way of using this information is to exclude descriptive features from the dataset that have low entropy scores.

Having said that, it would be fair to compare the Random forest classifier against the Decision tree and kNN classifiers because all three models have similar performance scores for the testing and training datasets.

In my opinion, the Random Forest classifier has a better chance to win when compared against the other models because it runs the ID3 algorithm on various bootstrap sample groups which enables it to be more robust to noise in the dataset. Furthermore, it can handle the higher dimensions in a dataset. As you can see, in the score charts below, random forest via entropy and Gini were able to reach 100% accuracy with 20 estimations. This does elude to it overfitting the dataset but strictly speaking, as per our performance metrics, it does win in comparison.

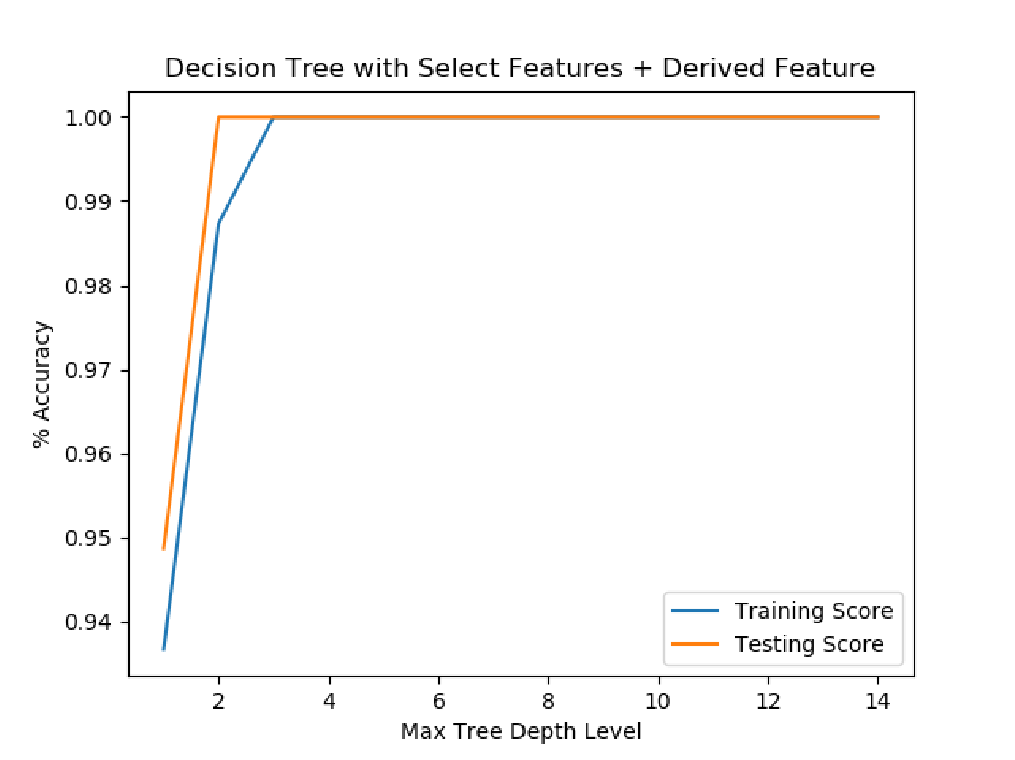


# VII. OPTIMIZATION

Using the rankings generated from the random forests, I tried running the Decision tree classifiers with both the Gini Index and Entropy across a tree depth range of 1 to 15. I did this by running the classifier with an excluded # of descriptive features starting with the feature that had lowest respective ranking. The highest testing dataset score achieved was with the top 6 features resulting in 95% with a tree depth of 3. Therefore, equal but no improvement in performance.

Our Dataframe is consistent of 118 instances and 13 descriptive features. The number of all possible outcomes for each target class is significantly larger than our historical dataset. It’s an ill-posed problem. Given that I’m not a subject matter expert in Wine making. I would need more time in order to develop this expertise for understanding whether a feature has a direct relation to the target feature or not. With time as a constraint, I decided to split the data frame by the target classes and find the average and standard deviation for each feature. Then compare the differences in the averages for each feature for each class. My goal was to find the features with the largest difference in average with the smallest standard deviation and sum them together to make a new derived feature. For my dataset, ‘Proline’ and ‘Magnesium’ fit this criterion. From here, I took the top 6 features and added this new derived feature and ran a decision tree classifier using the Gini Index and was able to out preform my previous decision tree accuracy of 95% with a tree depth of 3 to an accuracy of 100% with a tree depth of

2. (please see below for the score chart)



Therefore, I was able to beat my results from the previous tasks. The reason why this methodology worked is because the decision tree model works on splitting the dataset into the purest target class groups. What I did was identify which of the features were the furthest from each other when grouped by class. So by summing these features together, we could potentially create a feature that could split the dataset with the most amount of information gain which is exactly what happened in our case.

However, having said that, this model may be prone to be overfitting the dataset. If I had more time and/or resources I would investigate this further by confirming whether ‘Proline’ and/or ‘Magnesium’ is directly related to the target feature or maybe they’re indirectly related.

With regards to further optimizing though, I’d first like to extend my analysis on just the summation of two features but the summation X number of features. From here, run each potential derived feature along side Y number of descriptive features through a decision tree classifier model; with both Entropy and the Gini Index.

It is important to note that the kNN model can be potentially optimized by applying this same methodology.