Decision Trees and Random Forests

Damian Franco
Dept. of Computer Science
University of New Mexico
dfranco24@unm.edu

Meiling Traeger
Dept. of Computer Science
University of New Mexico
meilingt@unm.edu

Trey Sampson
Dept. of Computer Science
University of New Mexico
tsampson@unm.edu

***Abstract*—This project is intended to teach the ins and outs of decision trees and random forests. The decision tree will be constructed using information gain as split criteria and chi square as an alternate termination rule. Information gain is calculated with entropy, gini index, and misclassification error.**

***Index Terms*—gini index, entropy, information gain, decision tree, random forest, chi-square, validation set**

I. Introduction

Machine learning is a subsection of artificial intelligence (AI) that focuses on data and algorithms to classify and understand information. Implementation of machine learning techniques uses training data sets and models to make predictions for various tasks.

Given a specified set of features for mushrooms, the information is used to classify whether an instance of a mushroom is poisonous or edible. This task of classification will primarily focus on accuracy and precision to classify the most mushrooms correctly. Examples of the features are odor, cap-color, ring-number, and stalk-shape. In total there are twenty-two features and seven thousand, one hundred twenty-four instances of mushrooms. In stalk-root there are two thousand one hundred fifty-nine missing data points that will be accounted for.

The data will be classified using decision trees to generate a random forest. A decision tree is a graphical representation of the decision making process that involves making a sequence of choices based on a set of conditions. In the decision tree, each node represents a decision or event, and each branch represents an outcome or possible path. At each decision node, a condition is evaluated and the tree branches off into different paths based on the outcome of that condition. The process will recurse until a final decision or outcome is reached at the end of the tree. The random forest is generated from multiple decision trees. A final prediction is made by combining the predictions of all the trees in the forest. The main advantage of the decision tree is that it minimizes over fitting, identifies important features in the data set, and handles the missing data more effectively.

II. Methods

The decision tree is constructed using multiple methods to determine information gain. In machine learning, entropy is a measure of the impurity or randomness of a set of data. It is commonly used in decision tree algorithms to determine the best attribute to split the data on, and to build a tree that can classify or predict new instances. The equation for entropy is:

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{1}$$

The next method of measuring the impurity of our data set is by calculating the gini index. This metric determines homogeneity based on the likelihood of randomly selecting two identical characteristics from a class (with replacement). This calculation is powerful even when there is a large number of classes to choose from. Our data set only had two possible classes, so we were able to use other metrics for impurity, such as misclassification error. Classifying data into many groups, the gini index is the best method.

$$Gini = 1 - \sum_{i=1}^{n} p(x_i)^2 \tag{2}$$

Misclassification error is effective in binary classification models. This method of determining impurity is viable even in the case of highly imbalanced classes. Our training data was relatively balanced, having 3692 edible mushrooms and 3433 poisonous mushrooms. However, if we were given training data that was significantly skewed toward one class or the other, misclassification error would still give accurate information on determining which characteristic to create our split from.

$$M_E = 1 - max(p_1, p_2, ..., p_n) \tag{3}$$

Gini index, misclassification error, and entropy are all metrics used in the context of decision trees to evaluate the quality of a split. Differences between the three is prevalent when calculating for the highest index of information gain. Generally, entropy and impurity are the best methods to use because misclassification error can lead to overfitting in the data model.

Information gain is a measure of the improvement in entropy or impurity obtained by splitting a node in a decision tree algorithm. The goal of a decision tree algorithm is to find the attribute that provides the most information gain, which helps to create a more accurate and efficient model. The information gain is calculated by subtracting the expected entropy or impurity of the child nodes from the entropy or impurity of the parent node, weighted by the proportion of

instances in each child node. The formula for information gain is as follows:

$$Gain(S, A) = H(x) - \sum_{v \in val(A)} \frac{|S_v|}{|S|} H(x) \tag{4}$$

The chi-squared test is a statistical test that can be used to determine the independence of two variables. This test is to determine if a given characteristic has a strong correlation with our class characteristic, given an alpha value to determine the confidence interval. Using a specified method (entropy, Gini index, misclassification error) to find which potential split gives the highest information gain, ensures that the selected characteristic is correlated with our 'edibility' class before performing the actual split.

$$x^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{5}$$

Algorithm 1 ID3 Decision Tree Algorithm

Input: D = dataset, X = Current Attribute
if all instances in D have same class c **then**
  Label(T) = c
  Return T
end if
if No attribute has no positive information gain **then**
  Label(T) = most common classification in D
  Return T
end if
Label(T) = X
for each value x of X **do**
  D_x = instances in D
  if D_x is empty **then**
    Let T_x be a new Tree
    Label(T_x) = most common classification in D
  else
    T_x = ID3(D_x, Next attribute)
  end if
  Add branch from T to T_x labeled by x
end for
Return T

III. Implementation

Our first objective was to implement a method to fill in the missing data within our data sets. We noticed that for non-numeric values, that we had to either convert our data into numeric values or take a different approach. The method that we decided to use to fill the missing values is the "backfill" approach that was introduced by the Pandas library. When using backfill to fill in missing values, we will fill the missing values with the next available value from the data set. In other words, it will take the value from the next row and "backfill" it into the missing value. We had to first make sure that we had a semi-even distribution of values within our data set and that there was no discrepancy of large chunks of missing values that all get converted to the same value when the backfill method is applied. This approach allowed us to keep even distribution of values that existed previously, before the backfill method was applied and approach the data set fairly and accurately.

Next, we pre-processed our data sets for correct and accurately methods of measurement of accuracy for our model. We decided on a 80-20-20 split for our data where 80% of our data set was chosen at random, 20% of our testing data set and 20% for a validation set were all extracted from our input data sets that were given to us. The purpose of this technique is to prevent overfitting and ensure that the model generalizes well to new data. In this approach, the training data set is used to build the decision tree model, while the validation data set is used to tune the model by selecting the optimal hyperparameters and evaluating the performance of different models. Finally, the testing data set is used to evaluate the performance of the final model, which is selected based on the performance on the validation data set. By using this approach, we can avoid the risk of overfitting the model to the training data and obtain a more accurate estimate of the performance of the model on new, unseen data. Therefore, the use of a training, validation, and testing data set split is an essential step in building robust and accurate decision tree classification models.

We approached the overfitting problem with branch pruning. Branch pruning is a technique that involves removing some of the branches or nodes from a decision tree after it has been constructed. The goal is to create a simpler and more generalizable model that is less likely to overfit the training data. There are several types of branch pruning techniques, including reduced error pruning and cost complexity pruning. This is a useful technique for reducing overfitting in decision tree algorithms, and can help to create simpler and more generalizable models that perform better on new, unseen data. However, it is important to balance the complexity and accuracy of the model, as removing too many branches can result in a loss of accuracy. In our random forest model, we implemented a pruning function that takes an argument to the max-depth a tree branch can achieve before pruning the tree once that depth is reached. This allowed us to compare different values of tree sizes in our random forest generation and accuracy of different tree sizes.

The evaluation metric for this competition is Categorization Accuracy, which is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition in the equation below. We also accounted for binary classification accuracy by calculating in terms of positives and negatives.

$$Accuracy = \frac{Total \quad num \quad of \quad correct \quad predictions}{Total \quad num \quad of \quad predictions} \tag{6}$$

The classification technique itself had two parameters, the tree that was generated and an instance of the data set that

would classify edible or poisonous. The values of the instance will first be applied to the tree that is being evaluated on. This function will then recurse until a leaf node is reached then classification occurs. For example, since we found that odor was our most important feature, then the value of odor will be taken and parsed through the tree continuing down the branch the value leads to until a leaf node is hit stating the classification. This is also done in the random forest by the amount of trees created and returns the majority classification from the various trees.

IV. Results

Although we used three different methods of computing information gain amongst our attributes, We found that all of them had similar values that were computed. The most important attribute was *odor*, the second most important was *spore-print-color* and third was *gill-color* and then differences started occurring within the three. There were attributes that were considered more or less important than others when calculating the information gain, but we noticed that this did not affect our accuracy or tree-building methods. In Fig [1], Fig [2], and Fig [3], you can view our calculations for each of our attributes and their importance.

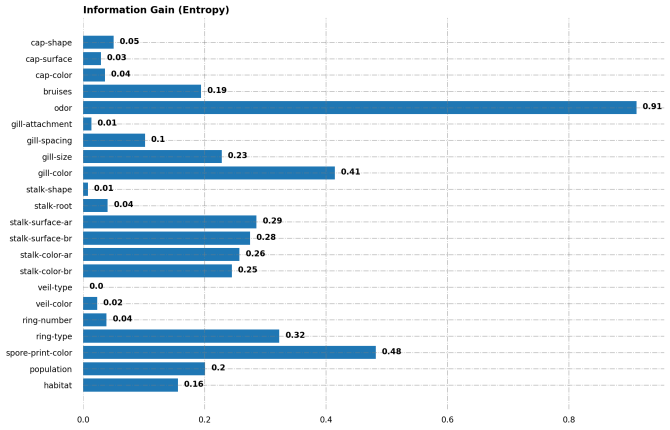

Fig. 1. Findings for our most important attribute while using *entropy* to compute the information gain.

Our accuracy for our model was substantially higher than expected. We were getting 100% for all the tested we conducted with our parameters set. The best max-depth parameter for the most accurate score was 5 with a total of 10 random tree generated in our forest. Below in the confusion matrix, you can view our true-positives and true-negatives. We did not encounter any false-positive or false-negatives in our experiment as shown in confusion matrix in Table I.

Findings for our chi-squared tests were similar to the findings of information gain. We learned that splitting on the top three attributes calculated by each approach was appropriate to building a successful tree for classifying the mushrooms. The α values that were tested consisted of 0, 0.01, 0.05, 0.5. Accuracy never ceased to be high for all of our chi-squared test which is shown in the table below. Once again, this was

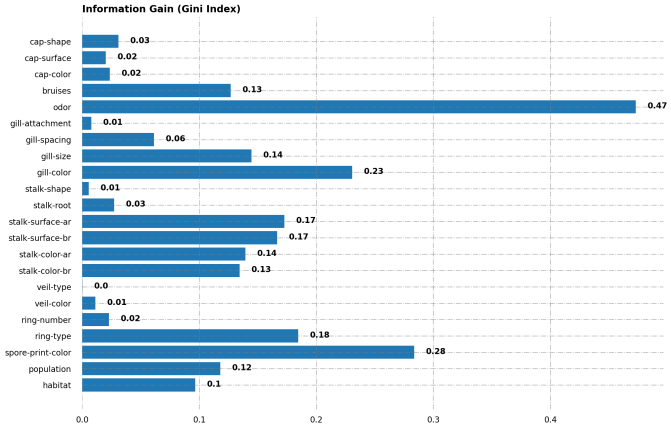

Fig. 2. Findings for our most important attribute while using *Gini Index* to compute the information gain.

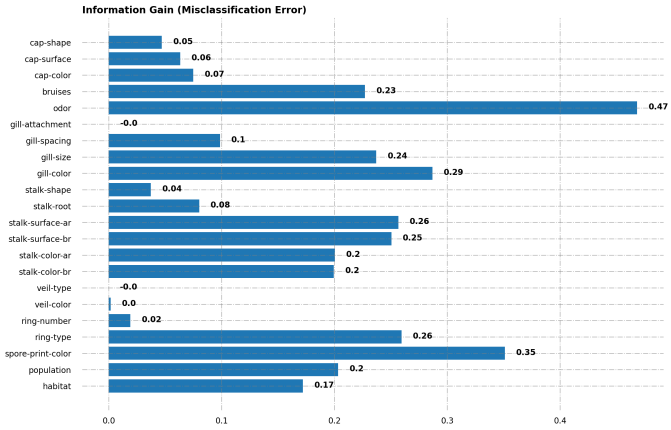

Fig. 3. Findings for our most important attribute while using *misclassifcation error* to compute the information gain.

	Predicted Positives (p)	Predicted Negatives (n)
Actual Positives (p)	**516**	**0**
Actual Negatives (n)	**0**	**483**

TABLE I
True and false positive and negatives of our tests confusion matrix.

all tested with 10 tree generated with the 80-20 data set and validated with the testing and validation data sets.

	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.5$
Entropy	**100%**	**100%**	**100%**	**100%**
Gini Index	**100%**	**100%**	**100%**	**100%**
Misclass Error	**100%**	**100%**	**100%**	**100%**

TABLE II
Chi-squared tests with various values testing for tree building and accuracy correctness.

V. Discussion and Conclusions

Our model classified 100% of the testing data set at the set parameter values that we configured in our Jupyter notebook.

These results were genuinely surprising because our first iteration of our model was only predicting about 49% of our classifications correctly in both the validation data set and the testing data set. You can also see some of our first implementations of the ID3 tree building methods and trees that were only classifying around 49% of the time with our code. Finding the important steps of classification with our model and manipulating them was very interesting to our team. Tuning hyperparamters with branch pruning, using the correct amount of data to training a model with, and creating valid trees to predict a classification result were just some aspects that stuck out to our team members.

As far as the prediction values themselves, we did test with different approaches to the manipulation of different steps in our model. These tests involved changing some steps in the ID3 tree building algorithm such as removing the chi-squared function call to see how that might effect the tree building. We found for that chi-squared values are very important to the validation that a split on an attribute is guaranteed to result in a tree that is very useful, but was not integral to generating great results when you replace the method with the information gain technique. The random forest implementation is excellent and very useful for making correct classifications compared to creating a single decision tree.

Building a model that correctly classified mushrooms based on the data sets given was very challenging. We had challenges understanding some of the various ways that we could split our data sets for training purposes and understanding the tuning of hyperparameters and why that is an important step for classifying. After spending a great amount of time on those topics and the project overall, we feel like we have accomplished and learned a great amount about implementing a great classification model and machine learning in general.

Submitting onto Kaggle, the calculated accuracy of 0.49 was the first submission that we did when our model was actually only producing 49% accuracy. After we edited our model, we were able to calculate our accuracy to 1.0 on Google Colab. For an unexplained reason after we resubmitted our correct csv file, the score on Kaggle did not change even after two additional submissions of a csv file. We are unsure why the score on Kaggle is not recalculating. Although the score on Kaggle is not showing correctly, you can see that on Google Colab, and in Fig 4, our prediction is calculating to exactly 1.0.

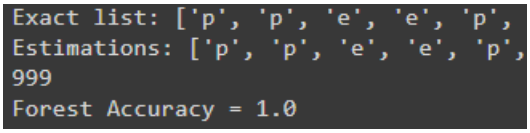

Fig. 4. Our accuracy for the testing data set classification on Google Colab. You may also view this in the bottom of the *Random Forest* section with our Jupyter notebook alongside the trees generated in our random forest.

REFERENCES

[1] T. M. Mitchell et al., Machine learning. wcb, 1997.

[2] Goodfellow, Ian, et al. Deep Learning. The MIT Press, 2017.

[3] Duch W, Adamczak R, Grabczewski K (1996) Extraction of logical rules from training data using backpropagation networks, in: Proc. of the The 1st Online Workshop on Soft Computing, 19-30.Aug.1996, pp. 25-30, available on-line at: http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/

[4] Duch W, Adamczak R, Grabczewski K, Ishikawa M, Ueda H, Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches, in: Proc. of the European Symposium on Artificial Neural Networks (ESANN'97), Bruge, Belgium 16-18.4.1997, pp. xx-xx

[5] Wlodzislaw Duch, Department of Computer Methods, Nicholas Copernicus University, 87-100 Torun, Grudziadzka 5, Poland