CS436 Project Report

Ryan Shendler, Timothy Shores, Christopher Dimeglio, and Joseph Ferring

Abstract

In this paper, we attempt to classify mushrooms as either poisonous or edible based on their physical attributes. To do this, we implemented the Decision Tree, Perceptron, and Naive Bayes learning algorithms on a dataset consisting of descriptions of poisonous and edible mushrooms. By altering parameters of each of these three algorithms, we attempted to determine the optimal algorithm for correctly classifying mushrooms of this dataset. Through our experiment, we were able to identify several configurations of these algorithms that are capable of achieving 100% test accuracy on this dataset.

Introduction and Related Works

It can be difficult to tell whether a mushroom is poisonous or edible just by looking at it. Therefore, our group wanted to determine which machine learning algorithm would be the most effective at determining whether a mushroom is poisonous or not, given a description of that mushroom's attributes. We decided to implement the Naive Bayes, decision tree, and perceptron classifiers and compare the accuracy of the three algorithms on our dataset. For our project, we used a dataset from the UCI Machine Learning Repository which records the physical attributes of various mushrooms from the Agaricus and Lepiota families and classifies them as either poisonous or edible. This dataset contains 8124 instances, each which has 22 features. The class of each instance is either poisonous or edible.

Several other researchers have worked on this dataset. Verma and Dutta implemented the Naive Bayes classification algorithm on this dataset and reported an accuracy of 96.817%

(Verma & Dutta, 2018, 96-97). Ismail, Zainal, and Mustapha implemented a decision tree learning algorithm on this dataset and recorded an accuracy of 100% (Ishmail et al., 2018, 414). Hall and Smith implemented both a Naive Bayes classifier and a C4.5 decision tree classifier, which resulted in accuracies of 94.75% and 99.59% respectively (Hall & Smith, 1999, 238). Our experiment managed to produce a more accurate Naive Bayes classifier than Verma's and Hall's and our decision tree classifier managed to achieve 100% test accuracy, much like Ismail's model.

Dataset and Experiment

As previously mentioned, our dataset has 8124 instances, 22 features and 2 classes. The features record certain physical attributes about a particular mushroom, with some of the features being cap-color, odor, population, and habitat. Most features have one or more character values, with the character representing the value of that feature for that particular mushroom. For example, the feature gill-size has two possible values, b and n, which represent broad and narrow gills respectively. Two particularly important features are stalk-root and veil-type. Stalk-root is important because 2480 instances in the dataset are missing a value for stalk-root. This missing value is indicated by a ? character in the dataset. Meanwhile, veil-type is important because every instance has the same value for veil-type, that being p or partial veil type.

During our preprocessing phase, we first used Scikit-learn's One Hot Encoder to encode class and feature values from characters to numeric values. This is necessary because most of Scikit-learn's classifier models only work on numeric data. After encoding our data, we then used Scikit-learn's train_test_split() function to split our dataset into 70% training data and 30% test data. This 30-70 split is considered to be our base dataset. In addition to our base dataset, we also created three alternative datasets which are modifications of our base data. For the

“noMissing” dataset, we removed the 2480 instances that had missing values for the stalk-root feature. For the “noVeilType” dataset, we removed the veil-type feature from our dataset because every instance had the same value for veil-type, which effectively made it a redundant feature. Finally, for the “replace” dataset, we replaced all of the missing values of the stalk-root feature with the most common value of that feature, which is b or bulbous. In our base and “noVeilType” datasets, we treat a missing value for the stalk-root feature as just another possible value for stalk-root.

Testing of the decision tree algorithm on the dataset was done using the DecisionTreeClassifier from the Scikit-learn library, which uses the CART decision tree algorithm. The algorithm is configured in our testing to use entropy as the information gain criterion. Pruning is done using the minimal cost-complexity pruning algorithm. To test the effects of pruning on the accuracy of the decision tree, the classifier is run using different values for the cost-complexity pruning coefficient a, where $a = 0$ results in no pruning and a larger a results in more aggressive pruning of the tree. As can be seen in Figure 3, pruning using CCP actually decreases the accuracy of the decision tree on this data. For most shuffles of the dataset, the decision tree algorithm achieves 100% accuracy on the test set, even without feature selection or pruning. The decision tree algorithm is generally able to build a tree which perfectly classifies the training set before the tree becomes too complex, precluding the need for pruning.

For our Naive Bayes algorithm, we assumed that our features are conditionally independent and follow a categorical distribution. For our implementation, we used the CategoricalNB() classifier from the Scikit-learn library. In our experiment, we ran our CategoricalNB classifier three times on each of our four datasets, with each run using a different level of LaPlace smoothing. For k=0 LaPlace smoothing, we set the value of the alpha parameter

of our CategoricalNB() classifier to 0.0. However, since using a LaPlace smoothing value of 0 can result in computational errors, Scikit-learn automatically replaces an alpha value of 0.0 with an extremely small non-zero value. Therefore, while our k=0 trials don't actually use a k value of 0, the value of k is so small that the effect is practically identical. In addition to testing with k=0, we also tried running our Naive Bayes classifier with k=1 and k=2 LaPlace smoothing on each of our datasets.

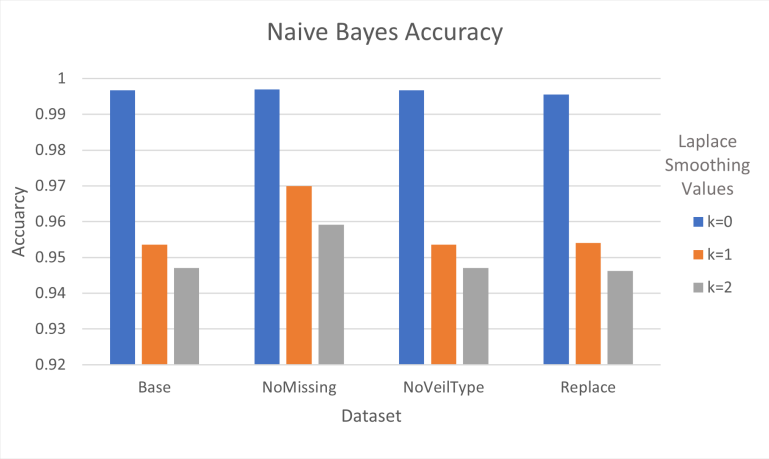
Our single layer perceptron model was implemented using the Perceptron module from the Scikit-learn library. Tests were performed with many different combinations of learning rates and iterations. The most accurate results were achieved when training the model with a learning rate of 0.001 for 100000 iterations. Training in this manner took about 40 to 50 seconds to produce a classifier. Tests were also performed with no regularization, L1 regularization, L2 regularization, and a combination of L1 and L2 regularization.

Results and Conclusion

As seen in Figure 1, our Naive Bayes classifier achieved maximum accuracy when run on the noMissing dataset with k=0 LaPlace smoothing. However, that is not the only interesting part of our data. One interesting trend is that higher values of k seem to result in lower accuracy for our Naive Bayes classifier. This suggests that the optimal value of k is an extremely small value that is close to but not equal to 0. Another interesting feature of our results is that our Naive Bayes classifier results in the same accuracy for both the base dataset and the noVeilType dataset. This is to be expected, as removing a redundant feature like veil-type should keep P(y=poisonous|x) proportional to P(y=edible|x), which means that the results of each classification should be the same.

Figure 1

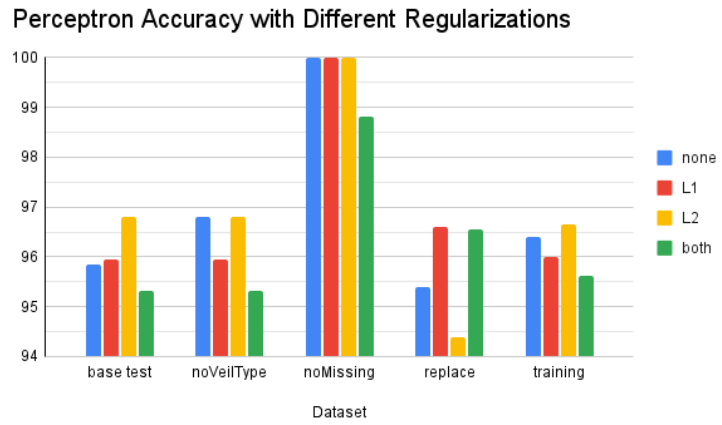
Accuracy of our Naive Bayes classifier



For our perceptron model, the noMissing dataset proved to be linearly separable, as our model was able to classify it with 100% accuracy. L2 regularization seemed to perform marginally better than any other combination of regularization, most likely due to the data having many codependent features. Additionally, the model with L2 regularization seemed to have the best ability to generalize and prevent overfitting based on its improvement in accuracy between the training and test data. Interestingly, training with fewer iterations resulted in a small decrease in accuracy of no greater than 2% while decreasing the training time tenfold to less than 10 seconds.

Figure 2

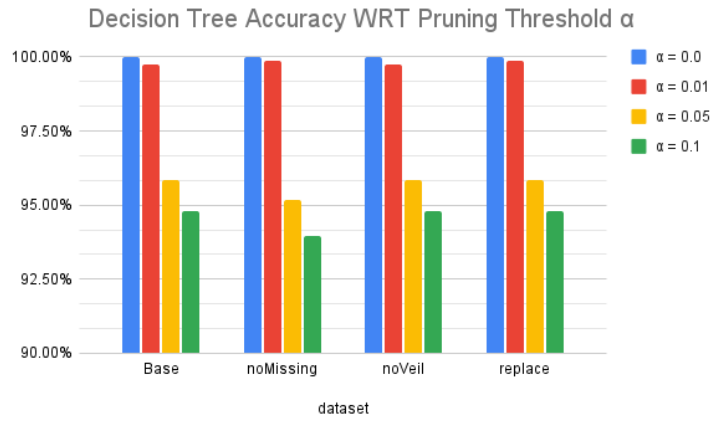
Accuracy of our Perceptron classifier



With the CART decision tree classifier it is found that the tree built is able to classify the test data with 100% accuracy, even with zero preprocessing or filtering of the dataset. The decision tree algorithm used here is able to select the features which are relevant early and categorize the data perfectly before the tree becomes too complex or overfits. As a result, pruning actually decreases the accuracy of the algorithm in the case of this data as the tree generated has very little extra complexity to begin with. This suggests that the data is easily categorized and has little to no noise on the relevant features, making the decision tree classifier a good fit.

Figure 3

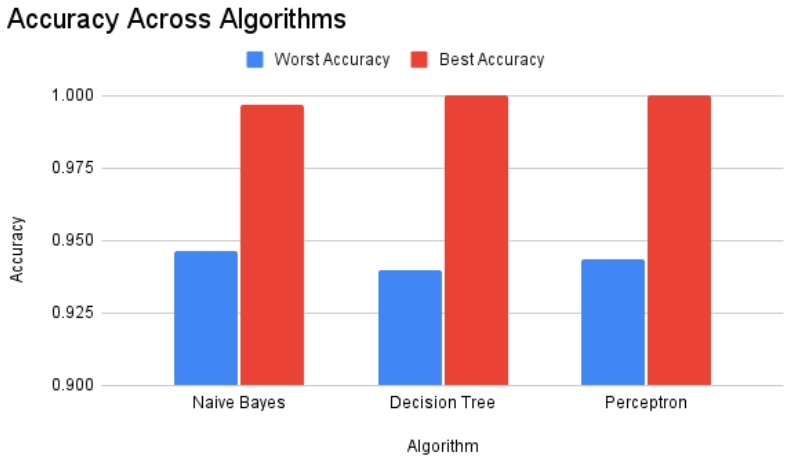
Accuracy of our Decision Tree classifier



As seen in Figure 4, the Decision Tree and Perceptron classifiers both achieved a maximum accuracy of 100%. This exceeds the Naive Bayes classifier's maximum accuracy of 99.7%. At first glance, these results seem to indicate that the Decision Tree and Perceptron are the optimal classifiers for this dataset. However, the Naive Bayes classifier has a higher minimum accuracy than the other two algorithms. This suggests that the Naive Bayes classifier is more consistent than the other two algorithms. This means that if we used a different dataset that had more noise, then the Naive Bayes classifier would likely perform better than the other two algorithms. However, for this particular dataset, either a Decision Tree with no pruning or a Perceptron with L1 or L2 regularization seems to be the optimal choice for correctly classifying mushrooms as either poisonous or edible.

Figure 4

Accuracy of all three classifiers



References

Hall, M. A., & Smith, L. A. (1999). Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper. *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, 235-239. 10.5555/646812.707499

Ishmail, S., Zainal, A. R., & Mustapha, A. (2018). Behavioural Features for Mushroom Classification. *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 412-415. 10.1109/ISCAIE.2018.8405508

Verma, S. K., & Dutta, M. (2018). Mushroom Classification Using ANN and ANFIS Algorithm. *IOSR Journal of Engineering (IOSRJEN)*, *08*(01), 94-100.