Mushroomia

Takao Oba

I. Introduction

Mushroomia is a land filled with diverse fauna and flora, but it's particularly well-known for its vast variety of mushrooms. Unfortunately, not all of the mushrooms in Mushroomia are edible, and it can be challenging to differentiate between the ones that are and the ones that aren't. To solve this problem, I have employed Shroomster Pro Max, an advanced data collection device that allows us to gather various data points on any mushroom found in the wild, and The National Archives on Mushrooms, a dataset collected by the Mushroomia government over the years.

Our goal is to construct and train machine learning models on The National Archives on Mushrooms dataset to determine which mushrooms I encounter on our journey are poisonous. I have done so through a systematic process of data loading and exploration, data pre-processing, data augmentation, statistical hypothesis testing, constructing models through non-ensemble and ensemble methods, and tuning the hyperparameters.

II. Methodology

A. Data Loading, Splitting, Exploration, and Visualization

The training dataset and testing dataset was loaded into Google Colab from Google Drive. First, the dataset was split into features and labels, where the “class” column represented the label or the variable in which the machine learning model aims to predict. The rest of the columns were used as tentative predictor variables. The testing data was also split into features and labels as well. Furthermore, the label or the “class” column was transformed from categorical variables ('p' and 'e') to numeric variables (1 and 0) using the map function. Additionally, I ensured that there was no skewness to the label values as this can potentially introduce bias to the model. Next, when scoping each of the predictors, the majority of them were categorical and only three of them were numerical. I visualized the numerical predictors through bar graphs and categorical variables as pie charts. Overall, I ensured that the predictors and the labels did not have any major flaws by first visually inspecting the shape and distribution.

B. Data Pre-Processing

The data must be processed prior to analysis. First, the missing values were identified by checking the number of missing values in the dataset for training and testing separately. The 'veil-type' predictor was identified to have 100% missing values in the testing dataset, and thus it was removed from both the training and testing datasets as it did not provide valuable information.

Overall, to process the data into the best way possible for the model to be used, I constructed a transformation pipeline. First, to handle the missing values or NaN values, an imputation strategy was used where missing values were replaced by the mode of the corresponding column. This strategy was chosen as it is a simple and effective way to handle missing values in categorical data.

Next, a numeric pipeline was built using the StandardScaler method to scale the numerical features in the dataset. This way, the numeric value's difference in magnitude will not create bias within the model. Finally, the OneHotEncoder method was used to convert categorical data into numerical data suitable for machine learning algorithms. Many of the predictor variables consisted of more than two distinct values. It will be dangerous to assign numeric values with difference in magnitude to these characters as these numbers will possess meanings. The prepared data was then transformed using the built pipeline. Similarly, the testing data was run through this pipeline as well. Overall, I pre-processed my data this way so that the machine learning model can better capture the fed data and further will not run into errors from missing values.

C. Data Augmentation

I aimed to aid the machine learning model through data augmentation or increasing the size and dimension of the dataset. The first new feature we are creating is called "stem_vol". This feature is constructed by multiplying the "stem-height" and "stem-width" columns of the dataset. This new feature represents the volume of the stem of the plant. This can be useful as certain mushrooms can have a very long stem, but can be skinny and vice versa. Overall, we would like to consider the overall volume so that we are not leaning towards one aspect of the dimension.

The second new feature we are creating is called "diam_stem_ratio". We obtain this new feature through dividing the "cap-diameter" column by the "stem-width" column. This new feature represents the ratio of the diameter of the cap to the overall width of the stem. This predictor illustrates the overall shape of the mushroom while simultaneously considering the cap as well as the stem. All in all, I expect that these two features that involve distinct viewpoints of the mushroom can present invaluable insights regarding the class of the mushroom.

D. Statistical Hypothesis Testing

I utilized a subset of the original dataset to perform statistical hypothesis testing. The chosen subset was the numerical predictors: “cap-diameter”, “stem-height”, and “stem-width”. The results of the logistic regression model showed that all three numerical variables had a statistically significant relationship with the class variable. Two of the variables had p-values less than 0.05 and “stem-height” had a p-value that was very close to 0.05. I should therefore include all of these variables when constructing a model.

Furthermore, I noticed that the pseudo R-squared value is 0.02134 which shows that the model was able to explain a very small amount of variance. This indicates that there may be other factors that contribute to the classification that assess if a mushroom is poisonous or not that is not captured by these three variables. Other categorical variables that I have excluded when performing the hypothesis testing should be included when constructing the machine learning models. This is intuitive as there are definitely more components that we must consider to assess the class of a mushroom than its physical dimension.

E. Models of your choice

I considered multiple models including logistic regression, decision tree, and naive bayes. Logistic Regression was initially used to transform the linear combination of the predictor variables into probability estimates of the binary outcome. Next Decision Trees were considered because they can handle both categorical and continuous data and are simple to understand and interpret. The pruning technique was used to reduce the depth of the tree and make the model less complex. However, pruning the tree led to a drop in accuracy score, highlighting the trade-off between computational cost and performance.

The last non-ensemble method I considered was Naive Bayes because it assumes that the features are independent of each other and that each feature contributes equally to the probability of the class. The Gaussian Naive Bayes model did not yield a high accuracy rate because the normality assumption likely does not hold in these given features. The Bernoulli Naive Bayes model provided a higher accuracy rate because it is most appropriate when working with binary or boolean data, and the data frame had more features involving binary values. Overall, the two distinct Naive Bayes models did not perform very well due to the data having both binary and continuous values, and the overall model assumption that the predictors are independent could have been violated. From the three models that were considered, the decision tree without pruning has performed the best.

F. Ensemble Method

The ensemble method I chose was Random Forest Classifier. The model involves constructing a large number of decision trees on different subsets of the input data and then aggregating their predictions to come up with a prediction. It is an ensemble method because it combines multiple decision trees to generate a more robust and less sensitive model. Random forest showed that it is a great fit for this dataset as it can handle a large number of features and can detect complex relationships between them.Furthermore, by the nature of random forest models, it reduces the risk of overfitting which inevitably implies an increase in testing accuracy. Overall, the random forest classification model has outperformed all of the models that we have constructed thus far.

G. Hyper-parameter Tuning

I aimed to identify the best model through hyper-parameter tuning and finding the best set of hyperparameters. The grid search method is used to explore all possible combinations of hyperparameters by defining a grid of parameter values to search over.

For each model, the corresponding hyperparameters were specified as a list of values to explore using the grid search method. The models were then fitted on the training data, and the function chose the best hyperparameters based on the cross-validation results. The best model was then determined based on the best accuracy score achieved by the model during the cross-validation.

The best hyperparameters found through the evaluations are the following:

- □ **Decision Tree Classifier**: max_depth=30, splitter='random'
- □ **Naive Bayes Classifier**: alpha=0.5, force_alpha=True
- □ **Random Forest Classifier**: max_depth=30, n_estimators=10

III. Results

Implementing the best hyper-parameters that was achieved previously, the following was achieved:

- □ **Decision Tree Classifier**:
    - □ Accuracy: 0.432019
    - □ Precision: 0.792294
    - □ Recall Score: 0.414434
    - □ F1 Score: 0.544205
- □ **Naive Bayes Classifier**:
    - □ Accuracy: 0.498066
    - □ Precision: 1
    - □ Recall Score: 0.460228
    - □ F1 SCore: 0.63051
- □ **Random Forest Classifier**:
    - □ Accuracy: 0.459285
    - □ Precision: 0.956522
    - □ Recall Score: 0.4394778
    - □ F1 Score: 0.602250

Note that above values can fluctuate for random forest classifier based on the specific seed value that is used.

The Naive Bayes Classifier has yielded the best result at classifying whether a mushroom is poisonous or not. It had yielded the highest value for all four of the evaluation metrics. Overall, the three models tend to have a higher precision score compared to the accuracy score, meaning that the model is correctly classifying positive instances more accurately compared to the negative cases.

These values were obtained through utilizing cross validation with K fold splits of 10. This strategy was chosen as it can provide a good trade off in terms of bias and variance. Furthermore, this method is computationally efficient compared to other strategies.

IV. Conclusion

I look forward to using the Naive Bayes Classifier (with hyperparameters alpha - 0.5 and for_alpha = True) during my adventure through Mushroomia. This is because the overall accuracy metrics was the highest amongst the model indicating that it can capture the true class of the mushroom more accurately so that I can identify which mushroom is poisonous and which one is not.

However, we must consider the limitations of this project. The dataset from the National Archives on Mushrooms which may not necessarily capture the entire picture of the current notion as it may be out of date or in reality may involve more predictors than given in the dataset. For example, if there are more predators or mushroom eaters in a specific region, there is a high chance that the mushrooms are not poisonous. Moreover, it is difficult to truly identify if a mushroom is poisonous or not and thus when recording and collecting the data, there could have been misclassification which can heavily impact the machine learning model. Furthermore, the results of our visualization were limited by the quality of the available data. There were a limited number of numerical and categorical variables to accurately visualize the data.

Regarding next steps, numerical variables can be assessed amongst each other through considering multicollinearity and further considering association for categorical variables. Additionally, locations can be addressed as well so that the machine learning model can be used in other contexts of mushroom classification as well.