Statistical Learning Project I: Mushroom Classification

Rachael Hawthorne

6/28/2020

Introduction

Dataset

The dataset I am using for this project was obtained from the UIC Machine Learning Repository and contains 8124 hypothetical mushroom samples from 23 species of mushrooms in the *Agaricus* and *Lepiota* Family. Each mushroom sample is described with 22 physical characteristics and belongs to either class poisonous or edible. The features used to describe the mushroom samples are: cap shape, cap surface, cap color, presence of bruises, odor, gill attachment, gill spacing, gill size, gill color, stalk shape, stalk root, stalk surface above the ring, stalk surface below the ring, stalk color above the ring, stalk color below the ring, veil type, veil color, ring number, ring type, spore print color, population, and habitat. Each of these features also include 2 or more levels as described at https://archive.ics.uci.edu/ml/datasets/Mushroom.

Research Question

Using this dataset and three distinct classification techniques, I hope to discover which technique is best used for datasets comprising of all categorical values. I also hope to learn if there are any features that could be easily used in the field to determine if a mushroom is edible or poisonous.

Methods

The three methods I chose to use to analyze my dataset are K-Nearest Neighbors Classification, Logistic Regression, and Linear Discriminant Analysis. Each of these methods are used specifically for classification problems. Though Linear Discriminant Analysis cannot work correctly with categorical variables, I chose to try and implement it anyway because there were no other methods that we learned about that can work with categorical variables. Details on why it will not work will be explained later on. As for KNN and Logistic Regression, both techniques work with categorical variables, so they are fit to use on my dataset.

To compare the performance of these techniques, we must use the same training and testing set for each model. To do this, I used the same seed (10) and the same training/testing sample proportions (75% & 25%) for every model. I also used the same method to analyze the results for each technique.

Preprocessing

Preprocessing my data for classification was a fairly simple process. The dataset indicates that there are missing values in the stalk-root feature due to some mushrooms not having a stalk attached. To deal with this, I first searched the dataset for the missing values to see how many of them there are. Upon searching, I found that there were 2,480 samples where the stalk root feature is listed as missing. Because there were so many rows with missing values in that feature, I decided to remove the entire feature instead of removing the rows, as removing the rows would shrink the dataset too much. I do not believe that removing this feature will affect the outcome of any of the classification techniques simply because it is apparent that there is a good probability that when finding a mushroom in the field, it will have a missing stalk root.

When looking at the data, I also found a problem with another feature: veil type. Unlike all of the other features every mushroom sample a partial veil type, making this feature

completely redundant for classification. To deal with this issue, I removed the entire veil type feature.

Following the removal of the previously stated features. I converted each feature to a factor for the classification techniques to know that they are to be treated as categorical variables. Any other data processing done is specific to the classification technique and will be discussed under the appropriate subheading. After the preprocessing was completed, the variables and their levels were as follows:

```
'data.frame':   8124 obs. of  21 variables:
 $ class                    : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
 $ cap.shape                : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
 $ cap.surface              : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
 $ cap.color                : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
 $ bruises                  : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
 $ odor                     : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
 $ gill.attachment          : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
 $ gill.spacing             : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
 $ gill.size                : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
 $ gill.color               : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
 $ stalk.shape              : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
 $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ veil.color               : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ ring.number              : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ ring.type                : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
 $ spore.print.color        : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
 $ population               : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
 $ habitat                  : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

K-Nearest Neighbors

The first classification technique I chose to implement is k-nearest neighbors classification. The first thing I did was to make sure I separated the class column from the rest of the dataset and put it into its own container. This is because the KNN algorithm does not want to take the class into consideration when classifying the data. This technique also requires the data to be numeric or Boolean variables which posed a problem for my entirely categorical dataset. So, before I could run the KNN function on my dataset, I had to do some more processing. First, I found all of the features that could be coded as Boolean variables (which happened to be only "bruises") and changed them as such: bruises = 1, no bruises = 0. Then for the rest of the features I dummy coded each of their levels into Boolean variables and converted them to data frames. Below is an example of the dummy coded feature, cap shape, with all its levels.

x <dbl>	f <dbl>	k <dbl>	b <dbl>	s <dbl>	c <dbl>
1	0	0	0	0	0
1	0	0	0	0	0
0	0	0	1	0	0
1	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	1	0
0	1	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0

Once all of the features have been dummy-coded, I split the data into training and testing sets with the training set consisting of 6093 samples (about 75% of the dataset) and the testing set

consisting of 2031 samples (about 25% of the dataset). The predictions were then generated with the KNN function given the training set, the testing set, a list of the actual classes of the samples in the training set, and a value *k* which determines how many neighbors the algorithm will consider for classification of a single sample. For my model, I chose $k = 90$ which is approximately the square root of the size of the data set. After the predictions were made, I gathered the data to compare the predictions to the actual classes of the testing set and formulated a confusion matrix. The details will be discussed in the results section.

Logistic Regression

The second classification technique I chose for my dataset is logistic regression. To perform logistic regression on my dataset, I first had to recode my class values to e = 1 and p = 0. I am also no longer using the dummy coded features as I used before since the logistic regression function understands how to work with categorical variables. I created my training and testing sets exactly as I did previously in the KNN technique with 75% of the data in the training set and 25% of the data in the testing set. For this technique, I did not use all of the features in the prediction due to issues with the algorithm not converging. Instead, I selected the features with the most ideal p-values to use in the model. The features I ended up using were gill spacing, gill size, and habitat. After the model was trained, I generated the probabilities for the testing set using the predict function. Once the probabilities were obtained, I specified that any probability greater than or equal to 0.5 would be classified as edible and any probability less than 0.5 would be classified as poisonous. The results were then gathered, and a confusion matrix was constructed.

Linear Discriminant Analysis

The third classification technique I chose to use is linear discriminant analysis. Because linear discriminant analysis is sensitive to collinear variables, the first thing I did was test the variables for collinearity. Because all of my variables are categorical, normal correlation methods cannot be used, so instead preformed a chi-squared test for each of the pairs of features to determine which features are not independent. I specified that if the p-value generated by the chi-squared test was greater than 0.05, the offending features were to be removed from the dataset. The results of these tests showed that the features “veil color” and “ring number” are dependent on each other, so I removed these features from the dataset before performing LDA. Then, the dataset was split again into the same training/testing set ratios as used before. Now, it is important to note that LDA assumes all features have normal distributions, which is a major reason it will not work correctly with categorical variables, because categorical variables are not normally distributed. But for the sake of this project and curiosity, I ran the technique anyway. I used the lda() function to train the model to predict class based on all of the features that were not removed from the dataset. The model was then cross validated on the testing set and the predictions were obtained and put into a confusion matrix.

Results

K-Nearest Neighbors

```
Confusion Matrix and Statistics

          Actual
Predicted    p    e
        p  973   14
        e   11 1033

               Accuracy : 0.9877
                 95% CI : (0.9819, 0.992)
    No Information Rate : 0.5155
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9754

 Mcnemar's Test P-Value : 0.6892

            Sensitivity : 0.9888
            Specificity : 0.9866
         Pos Pred Value : 0.9858
         Neg Pred Value : 0.9895
             Prevalence : 0.4845
         Detection Rate : 0.4791
   Detection Prevalence : 0.4860
      Balanced Accuracy : 0.9877

       'Positive' Class : p
```

Confusion Matrix. As you can see from the accuracy score, the model performed amazingly in predicting the correct classes based on the data. Cohen's kappa value is also very high, suggesting that the model performed much better than by chance. The model had barely any false positives, while the true positive and true negative prediction rates are 98.8% and 98.6%.

Logistic Regression

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.99163  -0.38936   0.00036   0.72663   2.60648

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.19694    0.05673  21.100  < 2e-16 ***
gill.spacingw  3.56673    0.14013  25.454  < 2e-16 ***
gill.sizen    -3.91241    0.12721 -30.754  < 2e-16 ***
habitatg      -1.56634    0.09156 -17.107  < 2e-16 ***
habitatl       0.17396    0.14912   1.167 0.243385
habitatm       0.63831    0.20462   3.119 0.001812 **
habitatp      -2.37762    0.12521 -18.989  < 2e-16 ***
habitatu      -0.64735    0.18990  -3.409 0.000652 ***
habitatw      15.36913  201.36533   0.076 0.939161
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8438.1  on 6092  degrees of freedom
Residual deviance: 4763.6  on 6084  degrees of freedom
AIC: 4781.6

Number of Fisher Scoring iterations: 15
```

Deviance Residuals. When looking at the deviance residuals we can see that they are somewhat symmetrical. Meaning they the model should have a decent amount of accuracy at predicting the response.

P-values. The p-values for gill spacing = w, gill size = n, habitat = g, habitat = p, and habitat = u are all statistically significant, meaning that there is likely a relationship between these predictors and the response. While habitat = m is moderately significant, habitat = l and habitat = w are not significant at all.

Analysis of Variance. I also ran an ANOVA test to analyze the deviance table. As we can see, the difference between the null deviance and the residual deviance is high for all three variables. This indicates that the model tests well against the null model. We also want to look for if adding each variable to the model decreases the residual deviance, meaning that adding the variable improves the model. We can see that each variable added decreases the residual deviance by ~1000-2000.

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: class

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         6092     8438.1
gill.spacing  1   851.58      6091     7586.5 < 2.2e-16 ***
gill.size     1  1989.45      6090     5597.1 < 2.2e-16 ***
habitat       6   833.40      6084     4763.6 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Confusion Matrix and Statistics
```

```
          Actual
Predicted   p   e
        p 797 159
        e 187 888

               Accuracy : 0.8296
                 95% CI : (0.8126, 0.8458)
    No Information Rate : 0.5155
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6587

 Mcnemar's Test P-Value : 0.1466

            Sensitivity : 0.8100
            Specificity : 0.8481
         Pos Pred Value : 0.8337
         Neg Pred Value : 0.8260
             Prevalence : 0.4845
         Detection Rate : 0.3924
   Detection Prevalence : 0.4707
      Balanced Accuracy : 0.8290

       'Positive' Class : p
```

Prediction Results. The prediction results are about as expected from the model analysis. The accuracy is at 82% which is generally considered good accuracy. Cohen’s kappa value is 0.6587, suggesting that the model performed much better compared to predicting class by chance. Overall, the model performed well with the features decided to include.

Linear Discriminant Analysis

```
lda.class    p    e
        p  968    0
        e    0 1063

               Accuracy : 1
                 95% CI : (0.9982, 1)
    No Information Rate : 0.5234
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4766
         Detection Rate : 0.4766
   Detection Prevalence : 0.4766
      Balanced Accuracy : 1.0000

       'Positive' Class : p
```

Confusion Matrix. As stated previously in this report, linear discriminant analysis does not work correctly when using categorical predictors. The reason for this being that LDA assumes that all predictors have a normal distribution. This is not the case with categorical variables. As we can see, the model gave some bizarre results. Somehow, the model came out with a 100% accuracy. I'm not sure why this happened, but I am sure that it is not accurate.

Discussion

Best Performance Model

Based on the accuracy of each model it is clear that the k-nearest neighbors method performed the best out of the three with a 98% accuracy. The next best performing model would be logistic regression with an accuracy of 82% and the worst performing model is, of course, linear discriminant analysis. The reason that the k-nearest neighbors method performed the best is most likely due to the dataset being composed of entirely categorical variables. Even though KNN has a weakness with categorical variables, this dataset has the advantage of being completely categorical. The significance of this is that the algorithm does not have to convert between scales like it would when handling a mix of categorical and continuous data.

Method Performance

Because this is primarily a classification problem, we mostly care about prediction. Consequently, it does not matter all that much that the model is less interpretable. For classification problems, it is better to value high accuracy than high interpretability. The KNN algorithm is hard to interpret, but it provides very accurate results. The logistic regression algorithm on the other hand is easier to interpret but comes at the cost of being less accurate. Therefore, KNN is still the best method to make predictions based on the dataset.

Other Methods

One method that I think would perform very well on this dataset and that I would have loved to use for this project is a decision tree classifier. The decision tree classifier works by learning a set of rules to use for classification. Once the tree consisting of "if else" like rules is built, each sample is sent through the branches of the tree that correspond to its data and ends at a

leaf that classifies it. I think this method would work very well with this dataset because decision trees are very well built for categorical data.

Research Question

As for the research questions mentioned at the beginning of this report, we have learned that the KNN model is the most suitable model for this particular dataset. The KNN model, however, cannot help us determine which features could be easily used to tell if a mushroom is safe to eat in the wild. Thankfully, the logistic regression model shows that gill spacing, gill size, and habitat can be used to determine whether a mushroom is edible most of the time. Though I wouldn't bet being poisoned on 82% accuracy.

References

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.