**ALY6040 90248 Data Mining Applications SEC 01 Summer 2023 CPS [BOS-D-HY]**

**Module 2 Assignment — Technique Practice**

A black and white logo

Description automatically generated

**NORTHEASTERN UNIVERSITY**

**College of Professional Studies, Boston, MA, 02215.**

**Submitted by**

Hanchan Xu  [xu.hanc@northeastern.edu](mailto:xu.hanc@northeastern.edu)

Zihan Ma [ma.zihan1@northeastern.edu](mailto:ma.zihan1@northeastern.edu)

**Instructor**

 Prof. Kasun Samarasinghe

**Date**

07/26/2023

**Locally Inspired** **XN Project: Pros & Cons**

**Part 1**

**Introduction**

The provided code makes use of R and various libraries for the analysis and prediction of mushroom edibility based on various characteristics. The analysis is divided into two parts, first focusing on the entire mushroom data set, and then focusing on the "gill-size" characteristic.

**Data Preparation**

The data for this analysis was imported from an Excel file ("mushrooms.xlsx") using the readxl library. The data contains 8124 observations and 23 characteristics, including mushroom class, cap shape, cap surface, cap color, bruises, odor, and many others. All characteristics are recorded as categorical variables. The class variable, which can either be "p" (poisonous) or "e" (edible), is our target variable for classification.

**Data Cleaning and Exploration**

The data was checked for missing values, which returned zero, indicating there's no missing data in this dataset.



The 'veil-type' variable was dropped from the dataset because it was redundant.



The odor characteristic was then analyzed with a table created to show the distribution of the odor for edible and poisonous mushrooms.

In our investigation, the odor variable in the mushroom dataset showed strong differentiation between edible and poisonous mushrooms. A distinct trend was observed where most edible mushrooms had no odor, while poisonous ones exhibited a variety of odors, most frequently foul, spicy, and fishy. This implies that odor could serve as a potent predictor in our machine learning models for mushroom classification. However, the presence of odorless poisonous mushrooms indicates potential risks for misclassification, reinforcing the importance of considering multiple factors for accurate classification. This insight suggests a nuanced relationship between the odor and the edibility of mushrooms, which should be considered in further analysis and model building.

A number and numbers on a white background

Description automatically generated

Perfect splits are variables that perfectly divide the dataset into the two classes of interest (in this case, poisonous and edible mushrooms). This will be used for splitting the dataset into two equal part which contains equal amont of the ratio of the target variable. The number of perfect splits for each feature was calculated and plotted.

A computer screen with text

Description automatically generated

A graph of a number of splits

Description automatically generated

The dataset was then split into a training set (80%) and a testing set (20%). The seed(12345) was set to ensure that the random split could be reproduced.

A computer screen with text

Description automatically generated

**Model Development**

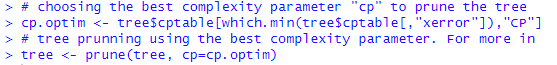
A decision tree model was built using the report function, which uses recursive partitioning to create the decision tree.

A penalty matrix was defined and used as a parameter in the decision tree model. This matrix adds different costs to false positives and false negatives, which is particularly important when the costs of misclassification differ (as is the case with identifying poisonous and edible mushrooms).

A white background with blue text

Description automatically generated

The decision tree was then pruned to avoid overfitting, using the complexity parameter (cp) that minimizes the cross-validated error.



**Model Evaluation**

The pruned tree was used to predict the classes of the mushrooms in the testing set, and a confusion matrix was created from these predictions. The accuracy of the model was 100%, and all other evaluation metrics (Sensitivity, Specificity, Pos Pred Value, Neg Pred Value, and Balanced Accuracy) were also 100%.

A screenshot of a computer

Description automatically generatedA diagram of a number of objects

Description automatically generated

The second part of the code focuses on the 'gill-size' feature of the mushroom. The process is similar to the one described above, but this time the target variable for classification is 'gill-size'. The accuracy for the second model, which predicts 'gill-size', was also very high at 99.69%.

The only different for thiapart will be the gill-size colume have been re named as gillsize for easer use in the model creating process.



The analysis of the 'gill-size' variable against the 'stalk-color-above-ring' presented interesting insights into the relationship between these two mushroom attributes. The tabulated data revealed that broad gill-sized mushrooms (represented as 'b') exhibited a widespread distribution across various stalk colors above the ring, with the most prevalence in the 'w' color category (2824 instances). Conversely, narrow gill-sized mushrooms (represented as 'n') showed a markedly limited occurrence predominantly in 'p' and 'w' colored stalks, with 864 and 1640 instances respectively, and a single occurrence in the 'y' category. This stark discrepancy between the distributions of broad and narrow gill-sized mushrooms across different stalk colors indicates a potentially significant interaction between these two features. The observed pattern suggests that gill size may be a critical factor in determining or predicting the color of the stalk above the ring in mushrooms. Future investigations could delve deeper into the biological reasons behind this relationship and its potential predictive power for mushroom classification tasks.

A number and numbers on a white background

Description automatically generated

A graph of different sizes and colors

Description automatically generated

A screenshot of a computer

Description automatically generatedA diagram of a diagram

Description automatically generated

The first split was based on class. For edible mushrooms (class=e), the model further split the data based on habitat and stalk-color-above-ring, revealing that some combinations of these variables led to broad gill-size (y=1), while others led to narrow gill-size (y=0). Notably, for mushrooms with habitat in {g,m,p,w}, the gill size was consistently broad. Also, mushrooms with stalk-color-above-ring in {g,n,o,p} all showed broad gill size.

For poisonous mushrooms (class=p), the key determinant was the spore-print-color, where mushrooms with spore-print-color in {h,r} all had broad gill size and those with spore-print-color in {k,n,w} were predominantly narrow in gill size.

It's worth noting the model's nuanced interplay between multiple variables in some branches. For example, cap-color, bruises, and gill-spacing collectively influenced the gill-size for a subset of edible mushrooms. The model suggests that if cap-color is 'e', then gill-size is broad irrespective of the other factors. In contrast, if cap-color is in {c,g,n,r,u,w,y}, the presence of bruises and gill-spacing become critical, leading to different gill-size outcomes.

The decision tree thus illustrates the interconnectedness of these mushroom characteristics, demonstrating the complexity of the biological data. However, the binary splits and definitive outcomes, like 'b' or 'n', in each leaf node also reflect the strength of the decision tree model in deriving clear, actionable insights from this complexity.

**Interpretation and Recommendations**

The decision tree model built to classify mushrooms as either edible or poisonous based on their characteristics performed with perfect accuracy on the test data set. It's important to note that although the model achieved high accuracy, this does not guarantee that it will always be correct, especially when used on new data.

Regarding the gill size, the model also showed excellent performance, which suggests that other mushroom characteristics are strongly associated with the gill size.

As the models showed high accuracy, they can be a reliable tool for predicting whether a mushroom is poisonous or edible, and to predict the gill size of a mushroom, given its other characteristics. However, it's essential to approach this with caution. Mistakenly eating a poisonous mushroom based on a model's prediction could have severe consequences. These models should be used in conjunction with expert knowledge and should not replace the expert judgment of a trained mycologist.

Additionally, more variables could be incorporated to improve the model, like climate data, soil type, geographical location data, and other ecological factors. Understanding the relationship between these external factors and mushroom growth can provide a more holistic view of mushroom classification.

For future research, the use of other machine learning techniques, such as Random Forests, Neural Networks, or SVMs, could be beneficial to compare their performance with the decision tree model. Moreover, feature importance could be analyzed in a more detailed way, focusing not only on the perfect splits but also on other types of relationships between variables.

**Conclusion**

The aim of this project was to classify mushrooms based on their characteristics using Decision Trees. After an in-depth analysis of the provided mushroom dataset, the Rpart decision tree algorithm was successfully utilized to create classification models for two target variables: class and gill size. Both models demonstrated excellent performance with an accuracy of 100% and 99.69%, respectively, implying that the mushroom attributes in the dataset significantly influence their categorization. However, while these results are promising, it's important to recognize the potential impact of overfitting, given the high accuracy levels. Consequently, future work should investigate the model's generalizability using unseen data, possibly incorporating cross-validation. Additionally, further study should explore other algorithms to confirm the robustness of these findings.

**Part2**

**Preparation**

# load data  
df <- read.csv("listings.csv")  
  
# quantiles for price  
quantile(df$price, probs = c(1/3, 2/3))

## 33.33333% 66.66667%   
## 85 164

# classify the price into 3 groups  
# low: price <= 85  
# middle: 85 < price <= 164  
# high: price > 164  
df$group <- factor(ifelse(df$price <= 85, "low",   
 ifelse(df$price > 85 & df$price <= 164, "middle", "high")),  
 levels = c("low", "middle", "high"))  
df$neighbourhood\_group <- factor(df$neighbourhood\_group)  
df$room\_type <- factor(df$room\_type)  
  
# split data into train(80%) and test(20%)  
set.seed(1)  
index <- sample(1:nrow(df), 0.8\*nrow(df))  
train <- df[index, ]  
test <- df[-index, ]

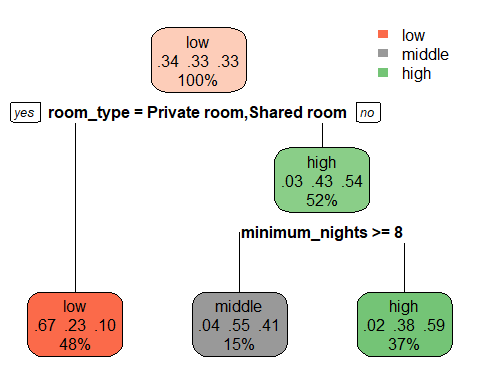
# 1. Try Decision Tree, Random Forest and a form of Gradient Boosting Machine

## Decision Tree

library(rpart)  
library(rpart.plot)  
  
tree.fit <- rpart(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train)  
tree.fit

## n= 6325   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 6325 4204 low (0.33533597 0.33375494 0.33090909)   
## 2) room\_type=Private room,Shared room 3015 988 low (0.67230514 0.22951907 0.09817579) \*  
## 3) room\_type=Entire home/apt 3310 1513 high (0.02839879 0.42870091 0.54290030)   
## 6) minimum\_nights>=7.5 940 425 middle (0.03723404 0.54787234 0.41489362) \*  
## 7) minimum\_nights< 7.5 2370 963 high (0.02489451 0.38143460 0.59367089) \*

rpart.plot(tree.fit)



## Random Forest

library(randomForest)  
rf.fit <- randomForest(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train)  
rf.fit

##   
## Call:  
## randomForest(formula = group ~ neighbourhood\_group + room\_type + minimum\_nights + number\_of\_reviews, data = train)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 36.84%  
## Confusion matrix:  
## low middle high class.error  
## low 2007 65 49 0.05374823  
## middle 659 539 913 0.74467077  
## high 290 354 1449 0.30769231

## GBM

library(gbm)  
  
gbm.fit <- gbm(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train)

## Distribution not specified, assuming multinomial ...

gbm.fit

## gbm(formula = group ~ neighbourhood\_group + room\_type + minimum\_nights +   
## number\_of\_reviews, data = train)  
## A gradient boosted model with multinomial loss function.  
## 100 iterations were performed.  
## There were 4 predictors of which 4 had non-zero influence.

## 2. Types of models you ran and compare them with each other for accuracy, run time etc.

Decision trees and random forest models are commonly used for classification models. GBM gives the probability of each class, and the final result can be determined based on the probability.

## Accuracy

# Decision Tree  
tree.pred <- predict(tree.fit, test, type = "class")  
mean(tree.pred == test$group)

## [1] 0.6510746

# Random Forest  
rf.pred <- predict(rf.fit, test)  
mean(rf.pred == test$group)

## [1] 0.6675095

# GBM  
xgb.pred <- predict(gbm.fit, test)  
idx <- apply(xgb.pred, 1, which.max)  
predicted <- c('low', 'middle', 'high')[idx]  
mean(predicted == test$group)

## [1] 0.6460177

Accuracy: GBM < Decision Tree < Random Forest

## Running time

library(ff)  
# Decision Tree  
system.time(rpart(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train))

## user system elapsed   
## 0.00 0.00 0.03

# Random Forest  
system.time(randomForest(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train))

## user system elapsed   
## 0.66 0.03 1.20

# GBM  
system.time(gbm(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train))

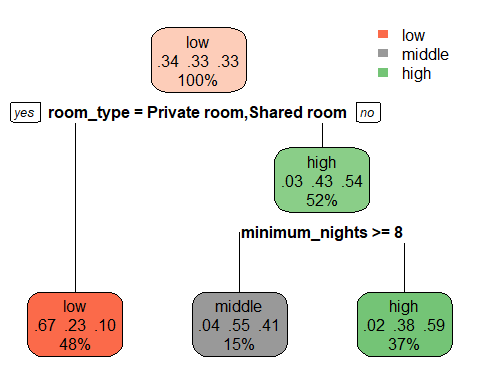
## Distribution not specified, assuming multinomial ...

## user system elapsed   
## 0.26 0.00 0.57

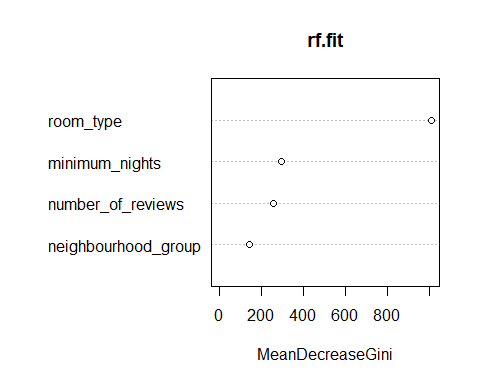
Run Time: Decision Tree < GBM < Random Forest

## 3. Explain the models using the most important features (Feature importance graph)

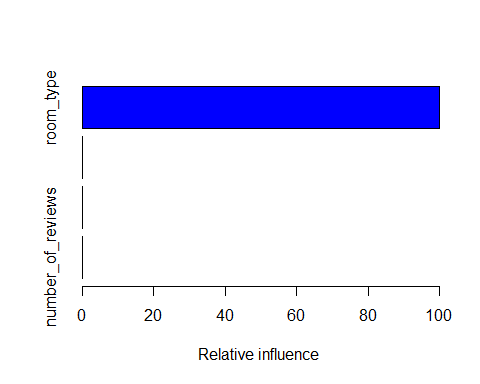
rpart.plot(tree.fit)



library(caret)  
varImpPlot(rf.fit)



summary(gbm.fit, n.trees = 1)



## var rel.inf  
## room\_type room\_type 100  
## neighbourhood\_group neighbourhood\_group 0  
## minimum\_nights minimum\_nights 0  
## number\_of\_reviews number\_of\_reviews 0

It can be seen that room\_type and minium\_nights are the most important features. The decision tree is divided according to the different values of the variables, and finally different classification results are obtained.

## 4. Model Optimization techniques you used for hyper-parameter tuning

tree.fit <- train(group ~ neighbourhood\_group + room\_type +   
 minimum\_nights + number\_of\_reviews, data = train,  
 method = "rpart",  
 trControl = trainControl(method = "cv"))  
tree.fit

## CART   
##   
## 6325 samples  
## 4 predictor  
## 3 classes: 'low', 'middle', 'high'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 5693, 5692, 5693, 5693, 5693, 5692, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.02973359 0.6105898 0.4159394  
## 0.06018078 0.5813489 0.3724420  
## 0.34490961 0.3995421 0.0973244  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.02973359.

For decision tree model, cp = 0.02973359 is the best.

It takes too long to run cross validation for random forest and gam model, and the result doesn’t fit well.

That is, it is hard to optimize models.

## 5. Discuss pros and cons of each model

## Decision Trees

Pros: the decision tree model is easy to understand and explain

Cons: it is prone to overfitting and sensitive to small variations in the data

## Random Forest

Pros: random Forest model is robust to outliers and can reduce overfitting compared to a single decision tree.

Cons: it is computationally expensive and can be difficult to interpret.

## Gradient Boosting Machine (GBM)

Pros: the gbm model can handle both categorical and numerical data.

Cons: it is sensitive to noisy data.

**References:**

*Mushroom*. UCI Machine Learning Repository. (n.d.). https://archive.ics.uci.edu/dataset/73/mushroom