Logo

Description automatically generated

**College of Professional Studies**

**Northeastern University San Jose**

**MPS Analytics**

**Course: ALY6040 – Data Mining Application**

**Assignment:**

Module 2 – Technique Practice

**Submitted on:**

April 23rd, 2023

**Submitted to:**  **Submitted by:**

Professor: BEHZAD AHMADI NIKSHITA RANGANATHAN

**ABSTRACT**

Large datasets can be analyzed for insights and patterns using powerful data mining techniques. In order to find significant relationships in the data that can be used to inform business decisions, enhance workflows, and gain a competitive edge, computational and statistical methods must be used.

Several methods are used for data mining, including:

* **Classification:** This involves categorizing data into predefined classes based on specific criteria. A bank, for instance, might use classification to decide, based on a loan applicant's credit score, income, and other factors, whether they are high or low risk.
* **Clustering:** This involves grouping data points based on their similarity. For example, retailers can employ clustering to recognize customer segments based on their purchasing patterns.
* **Regression analysis:** This means modeling how several independent variables and a dependent variable interact with each other. For instance, based on a customer's age, gender, and other characteristics, an insurance company can use regression analysis to forecast the likelihood of a claim.

Financial, medical, marketing, and manufacturing sectors among many others use data mining. It enables companies to target their markets, enhance pricing and promotions, reduce expenses, and boost efficiency in operations.

It is important to remember that data mining requires careful planning and execution in order to yield reliable and useful outcomes. Data quality, selection of appropriate methods, and interpretation of results are key factors to consider. Privacy and ethical issues must also be considered when working with sensitive data.

**INTRODUCTION**

**About this Dataset:**

The mushroom classification dataset is a popular dataset available on Kaggle, which provides details on various types of mushrooms and whether they are poisonous or edible. The dataset was obtained from the UCI Machine Learning Repository and has been widely used for educational and research purposes.

The dataset contains **8124** instances of mushrooms, each of which is a single mushroom. Each instance is characterized by **23** attributes that describe various physical characteristics of the mushroom. The attributes include features such as cap shape, cap surface, gill size, stalk color, and more. The remaining attribute, the target variable, is a binary classification indicating whether the mushroom is edible or poisonous.

One of the challenges of working with this dataset is that some of the attributes are categorical and have multiple possible values, such as cap shape, which can be bell, conical, convex, flat, knobbed, or sunken. In order to use the data for modeling, it must first undergo careful preprocessing and encoding.

Despite this challenge, the mushroom classification dataset has been widely used for machine learning and data mining research, particularly in the area of classification algorithms. Researchers have used various techniques such as decision trees, logistic regression, and support vector machines to build models that can accurately predict the edibility of mushrooms based on their characteristics.

Below are the basic statistics of the Raw dataset:

Graphical user interface, application

Description automatically generated

***Figure 1 – Basic Statistics of the Dataset***

Below is the Data Structure of the Mushroom dataset:

**Chart

Description automatically generated**

***Figure 2 – Data Structure***

The dataset's attributes and its description are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Attribute** | **Description** | **Possible Values** |
| 1. | class | Edibility of the mushroom | edible (e) or poisonous (p) |
| 2. | cap-shape | The shape of the cap | bell (b), conical (c), flat (f), knobbed (k), sunken (s), or convex (x) |
| 3. | cap-surface | The surface texture of the cap | fibrous (f), grooves (g), smooth (s), or scaly (y) |
| 4. | cap-color | Color of the cap | buff (b), cinnamon (c), red (e), gray (g), brown (n), pink (p), green (r), purple (u), white (w), or yellow (y) |
| 5. | bruises | Presence of bruises | no bruises (f) or bruises (t) |
| 6. | odor | The odor of the mushroom | almond (a), creosote (c), foul (f), anise (l), musty (m), none (n), pungent (p), spicy (s), or fishy (y) |
| 7. | gill-attachment | Attachment of gills to the stalk | attached (a) or free (f) |
| 8. | gill-spacing | Spacing between gills | close (c) or crowded (w) |
| 9. | gill-size | Size of the gills | broad (b) or narrow (n) |
| 10. | gill-color | Color of the gills | buff (b), red (e), gray (g), chocolate (h), black (k), brown (n), orange (o), pink (p), green (r), purple (u), white (w), or yellow (y) |
| 11. | stalk-shape | The shape of the stalk | enlarging (e) or tapering (t) |
| 12. | stalk-root | Type of root at the base of the stalk | bulbous (b), club (c), equal (e), rooted (r), or missing (?) |
| 13. | stalk-surface-above-ring | The surface texture of the stalk above the ring | fibrous (f), silky (k), smooth (s), or scaly (y) |
| 14. | stalk-surface-below-ring | The surface texture of the stalk below the ring | fibrous (f), silky (k), smooth (s), or scaly (y) |
| 15. | stalk-color-above-ring | Color of the stalk above the ring | buff (b), cinnamon (c), red (e), gray (g), brown (n), orange (o), pink (p), white (w), or yellow (y) |
| 16. | stalk-color-below-ring | Color of the stalk below the ring | buff (b), cinnamon (c), red (e), gray (g), brown (n), orange (o), pink (p), white (w), or yellow (y) |
| 17. | veil-type | Type of veil covering the gills | partial (p) |
| 18. | veil-color | Color of the veil covering the gills | brown (n), white (w), yellow (y), or orange (o) |
| 19. | ring-number | The number of rings on the stalk | none (n), one (o), or two (t) |
| 20. | ring-type | Type of ring on the stalk | evanescent (e), large (l), none (n), pendant (p), or flaring (f) |
| 21. | spore-print-color | The spore print's color | buff (b), chocolate (h), black (k), brown (n), orange (o), green (r), purple (u), white (w), or yellow (y) |
| 22. | population | Relative abundance of the mushroom in the area where it was found | abundant (a), scattered (s), clustered (c), several (v), numerous (n), or solitary (y) |
| 23. | habitat | Type of environment the mushroom was found in (e.g., woods, grasses, leaves, etc.) | grasses (g), leaves (l), woods (d), meadows (m), urban (u), waste (w), or paths (p), |

*Table 1: Dictionary with the Features of the Mushroom classification dataset*

**CODE WALK-THROUGH**

* **Understanding the dataset:**

Table

Description automatically generated

***Figure 3 – Mushroom dataset***

The structure of the "mushrooms" data frame is printed using the str function. The dataset contains information about different types of mushrooms with a "class" variable that appears to be the target variable, which indicates whether the mushroom is poisonous (p) or edible (e).

* **Checking the missing values**

Using the nrow and complete.cases functions, we can determine how many rows of the data set have missing values. The complete.cases function returns a logical vector that indicates whether a given row of the data set has any missing values. This step is important to handle missing data appropriately in order to avoid biased results and incorrect conclusions in data analysis.

* **Deleting redundant variable ‘viel.type’**

This command deletes the column "veil.type" from the data frame "mushrooms". The variable is considered redundant, which means that it does not add any significant value to the analysis and can be removed to simplify the data set. This variable contains only one unique value in the dataset, which is "p". Since it does not contain any useful information for the analysis, it is removed from the dataset

* **Analyzing odor variable**

The frequency count for the "class" and "odor" variables in the mushroom dataset is displayed in the table.

This relationship between the "class" and "odor" variables can be used to accurately predict the class of mushrooms based on their odor type.

* **Plotting the number of perfect splits vs feature**
  + **Calculation of the number of perfect splits**

This R code computes the number of perfect splits for each feature in the dataset, which is a measure of how well a feature can be used to classify the mushrooms into edible or poisonous.

* **Sorting the number of perfect splits in descending order**

Here, we have reordered the previously calculated number.perfect.splits vector in descending order and assigned it to the same vector

* **Bargraph**

This code creates a bar plot using the barplot() function to visualize the number of perfect splits for each feature in the mushrooms dataset. The par() function sets the margins for the plot. The main, xlab, and ylab arguments define the plot's title and axis labels.

* **Splitting the dataset**

Data splitting is a machine learning technique that divides a dataset into subsets for the purposes of training and testing a model. A machine learning model is trained using a training set, and its performance is assessed using a test set.

The dataset has been split into 80% (training set) and 20% (testing set) in this case.

* **Creating a Penalty matrix**

The penalty matrix is a 2x2 matrix used in decision tree models to handle class imbalance problems.

* **Building a Classification tree**

The rpart algorithm is a popular method used to create decision trees for classification and regression problems. In this specific code, we are using it to build a classification tree for the mushroom data set. The "class" variable is the target variable that we are trying to predict.

* **Visualizing the Decision tree**

By examining the tree, we can gain insight into the model’s decision-making process and identify which variables are most important in predicting the target variable.

* **Tree Pruning**

This code is used to perform tree pruning on a decision tree model. The aim of tree pruning is to reduce the complexity of the decision tree and increase its accuracy by eliminating branches that do not contribute significantly to improving the model's predictive performance.

The first line of the code selects the optimal value of the "cp" (complexity parameter) that will be used for pruning. The second line of the code prunes the decision tree using the optimal "cp" value selected in the first step.

* **Testing the model**

We have evaluated the predictive performance of the decision tree model and compared it to other models or baseline methods.

* **Evaluating the accuracy**

In this last step, we calculate the accuracy of the model by comparing the predicted classes to the actual classes in the test dataset using a confusion matrix.

**ANALYSIS**

Below are some key observations:

A picture containing text

Description automatically generated

***Figure 4 – Missing values***

* There are no missing values in the data set.

Graphical user interface, application, table, Excel

Description automatically generated

***Figure 5 – Basic statistics***

* Descriptive statistics summary of the dataset shows that most of the attributes have a similar mean and median value indicating a normal distribution. The range of values varies between attributes and is highest for the gill color attribute. It can be also observed that the veil-type attribute has a standard deviation of zero indicating that it has only one value

Chart

Description automatically generated

***Figure 6 – Table between odor and class***

* Based on the observations in the above table between class and odor, it is obvious that the mushrooms with odor values ‘c’, ‘f’, ‘s’, ‘m’, ‘y’, and ‘p’ are poisonous as they all have non-zero counts for the poisonous class and zero counts for the edible class.

Chart, histogram

Description automatically generated

***Figure 7 – Bargraph***

* The "odor" variable has the highest number of perfect splits, indicating that it is the most important variable for classifying mushrooms as edible or poisonous. Other variables, such as "stalk.color-above-ring" " stalk.color-below-ring" and "spore-print-color" also have a relatively high number of perfect splits.
* Splitting the dataset into testing and training sets helped us in preventing overfitting, which occurs when a model is too closely tailored to the training data and performs poorly on new, unseen data.

**Table

Description automatically generated**

***Figure 8 – Penalty matrix***

* In the penalty matrix, we assigned a penalty of 1 for misclassifying an edible mushroom as poisonous and a penalty of 10 for misclassifying a poisonous mushroom as edible. This is because misclassifying poisonous mushrooms as edible is more dangerous.

**Timeline

Description automatically generated**

***Figure 9 – Decision tree***

* While building the decision tree, the loss parameter is set to the penalty matrix. This is used to reduce the number of poisonous mushrooms misclassified as edible. Also, the method argument is set to "class" indicating that the tree is to be used for classification purposes.
* The tree is pruned to make the resulting tree simpler, more generalizable, and easier to understand.

A screenshot of a computer

Description automatically generated with low confidence

***Figure 10 – Confusion matrix***

* The output of the confusionMatrix() function shows that the model has correctly classified all the samples in the test dataset, resulting in an accuracy of 100% with a 95% confidence interval of (0.9977, 1). This indicates that the model is very accurate in predicting whether a mushroom is edible or poisonous. The output also shows that the sensitivity and specificity of the model are both 1, indicating that the model has correctly identified all the positive and negative samples, respectively.

**INTERPRETATION AND RECOMMENDATIONS**

Based on the analysis, it can be concluded that the odor of a mushroom is the most important factor in determining whether it is poisonous or edible. Other attributes, such as "spore print color" and "stalk surface above the ring" should also be given high weightage, as they are also strong indicators of the mushroom's class.

While the classification model can be a useful tool in identifying whether a mushroom is edible or poisonous, it is important to exercise caution when consuming wild mushrooms. Some poisonous mushrooms can closely resemble edible ones, and misidentifying a mushroom can have serious consequences.

It is recommended to gather more data on the physical characteristics of mushrooms and their edibility to improve the accuracy of the classification model.

Future research can also focus on collecting data on other variables that may be important in predicting the class of the mushroom, such as the geographic location and habitat of the mushroom.

**REFERENCES**

Holtz, Y. (n.d.). *The R Graph Gallery – Help and inspiration for R charts*. The R Graph Gallery. <https://r-graph-gallery.com/>

*Home - RDocumentation*. (n.d.). https://www.rdocumentation.org/

|  |
| --- |
| **APPENDIX** |

#---------------------- Week\_2\_Module\_2 R Script ----------------------#

print("Module 2 – Technique Practice")

print("Course Name - ALY6040 – Data Mining Application")

# Installing libraries

install.packages('rpart')

install.packages('caret')

install.packages('rpart.plot')

install.packages('rattle')

install.packages('readxl')

# Loading libraries

library(rpart,quietly = TRUE)

library(caret,quietly = TRUE)

library(rpart.plot,quietly = TRUE)

library(rattle)

library(readxl)

library(formattable)

library(psych)

# Reading the data set as a dataframe

mushrooms <- read\_excel("mushrooms.xlsx")

# Structure of the data

str(mushrooms)

# Number of rows with missing values

nrow(mushrooms) - sum(complete.cases(mushrooms))

# Descriptive Statistics for entire dataset

formattable(describe(mushrooms),

caption = "Descriptive statistics summary of the Mushroom Classification Dataset")

# Data Profiling Report

report <- create\_report(mushrooms)

plot\_report(report)

# Deleting redundant variable `veil.type`

mushrooms$veil.type <- NULL

# Analyzing the odor variable

table(mushrooms$class,mushrooms$odor)

number.perfect.splits <- apply(X=mushrooms[-1], MARGIN = 2, FUN = function(col){

t <- table(mushrooms$class,col)

sum(t == 0)

})

# Descending order of perfect splits

order <- order(number.perfect.splits,decreasing = TRUE)

number.perfect.splits <- number.perfect.splits[order]

# Plot graph

par(mar=c(10,2,2,2))

barplot(number.perfect.splits,

main="Number of perfect splits vs feature",

xlab="",ylab="Feature",las=2,col="wheat")

# Data splicing

set.seed(12345)

train <- sample(1:nrow(mushrooms),size = ceiling(0.80\*nrow(mushrooms)),replace = FALSE)

# Training set

mushrooms\_train <- mushrooms[train,]

# Test set

mushrooms\_test <- mushrooms[-train,]

# Penalty matrix

penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)

# Building the classification tree with rpart

tree <- rpart(class~.,

data=mushrooms\_train,

parms = list(loss = penalty.matrix),

method = "class")

# Visualize the decision tree with rpart.plot

rpart.plot(tree, nn=TRUE)

# Choosing the best complexity parameter "cp" to prune the tree

cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]

# Tree pruning using the best complexity parameter. For more in

tree <- prune(tree, cp=cp.optim)

# Testing the model

pred <- predict(object=tree,mushrooms\_test[-1],type="class")

# Calculating accuracy

t <- table(mushrooms\_test$class,pred)

confusionMatrix(t)