4442\_Project - Wine Dataset

Multinomial Regression to Predict Grape Variety

Grayson Gunderson and Thomas Beadle

2024-03-13

## The 3 types of Logistic Regression: When to use Multinomial Logistic Regression

Multinomial Logistic Regression is useful when an outcome variable has more than two categories with no inherent ranking and we want to model the probability of each outcome. Binomial Logistic Regression can be used to predict a categorical outcome with only two possibilities. Ordinal Logistic Regression is preferred when the outcome variable has more than 2 categorical outcomes and is logically ordered. The nature of the outcome variable will determine which of the three logistic regression techniques would produce the best model for a given data set. Similar to Binary Logistic Regression, Multinomial Logistic Regression models the probabilities of the outcome categories. However, instead of modeling the probability of one category versus another, Multinomial Logistic Regression models the probability of each category relative to a reference category. 1 or more independent variables are used to predict the probability that an observation belongs to each category of the outcome variable. Generally, Multinomial Logistic Regression is used when categories of the dependent variable have no inherent ranking, such as color of a car, types of drinks, or in this case, wine cultivars (the type of grapes used to make the wine).

Below are some prerequisites that must be met by the data in order to model its outcome using Multinomial Logistic Regression: - The outcome must be categorical with more than two outcomes - The Categorical outcomes must be nominal (not logically ordered) - The data set used to create the model should contain observations that fall into each of the categorical outcomes - The sample size should be large enough to reliably predict each of the unique outcome cases. As the number of categorical outcomes increases, the sample size required to build a robust model to predict each outcome also increases. - Predictor variables should have at least 10-15 observations (more is better) in order for them to be considered as included predictors. - Predictor variables should have a measurable impact on the outcome; if they are unrelated or share a weak relationship, we will not be able to reliably predict the outcome based on the values of the predictors.

It is also important that the number of categories and predictor variables are not too large. The number of fitted parameters in a Multinomial Logistic Regression model is given by the formula P(K-1), where P is the number of predictors and K is the number of categories, a high number of either one can lead to a prohibitively complex model.

## Dataset Information:

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars (grape types). The analysis measured the quantities of 13 constituents found in each of the three cultivars.

The attributes are:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

## Research Question: Can we predict the cultivar of wine (Class 1, 2, or 3) based on the values of chemical properties that were recorded in this data using Multinomial Logistic Regression?

Upload the wine data set to R.

# Specify URL  
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"  
  
# Read the dataset into a data frame  
wine.data <- read\_csv(url, col\_names = FALSE, show\_col\_types = FALSE)  
  
# Add column Names  
colnames(wine.data) <- c("Class", "Alcohol", "MalicAcid", "Ash", "AlcalinityOfAsh", "Magnesium", "TotalPhenols", "Flavanoids", "NonflavanoidPhenols", "Proanthocyanins", "ColorIntensity", "Hue", "OD280/OD315", "Proline")

The response variable “Class” was loaded in as a numeric data type. In Multinomial Logistic Regression, the response variable is categorical, so we will change the “Class” variable from a numeric to a factor. Now we can check the structure of the Data and the Summary to ensure it was uploaded successfully.

wine.data$Class <- as.factor(wine.data$Class)  
str(wine.data)

## spc\_tbl\_ [178 × 14] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ Class : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Alcohol : num [1:178] 14.2 13.2 13.2 14.4 13.2 ...  
## $ MalicAcid : num [1:178] 1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...  
## $ Ash : num [1:178] 2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...  
## $ AlcalinityOfAsh : num [1:178] 15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...  
## $ Magnesium : num [1:178] 127 100 101 113 118 112 96 121 97 98 ...  
## $ TotalPhenols : num [1:178] 2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...  
## $ Flavanoids : num [1:178] 3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...  
## $ NonflavanoidPhenols: num [1:178] 0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...  
## $ Proanthocyanins : num [1:178] 2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ...  
## $ ColorIntensity : num [1:178] 5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...  
## $ Hue : num [1:178] 1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...  
## $ OD280/OD315 : num [1:178] 3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...  
## $ Proline : num [1:178] 1065 1050 1185 1480 735 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. X1 = col\_double(),  
## .. X2 = col\_double(),  
## .. X3 = col\_double(),  
## .. X4 = col\_double(),  
## .. X5 = col\_double(),  
## .. X6 = col\_double(),  
## .. X7 = col\_double(),  
## .. X8 = col\_double(),  
## .. X9 = col\_double(),  
## .. X10 = col\_double(),  
## .. X11 = col\_double(),  
## .. X12 = col\_double(),  
## .. X13 = col\_double(),  
## .. X14 = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

We should also check our data set for any missing values before proceeding.

# search entire data set for any missing values  
missing\_vals <- colSums(is.na(wine.data))  
  
# Display the results  
print(missing\_vals)

## Class Alcohol MalicAcid Ash   
## 0 0 0 0   
## AlcalinityOfAsh Magnesium TotalPhenols Flavanoids   
## 0 0 0 0   
## NonflavanoidPhenols Proanthocyanins ColorIntensity Hue   
## 0 0 0 0   
## OD280/OD315 Proline   
## 0 0

We can see from the output that there is no missing data to be handled.

One of the column names in our data set has a format that could cause problems. Let’s replace the “/” with “\_” in the “OD280/OD315” column to prevent any interpretability issues that could arise later on.

colnames(wine.data)[colnames(wine.data) == "OD280/OD315"] <- "OD280\_OD315"

We can run the summary function to get an overview of our dataset in its current state.

summary(wine.data)

## Class Alcohol MalicAcid Ash AlcalinityOfAsh  
## 1:59 Min. :11.03 Min. :0.740 Min. :1.360 Min. :10.60   
## 2:71 1st Qu.:12.36 1st Qu.:1.603 1st Qu.:2.210 1st Qu.:17.20   
## 3:48 Median :13.05 Median :1.865 Median :2.360 Median :19.50   
## Mean :13.00 Mean :2.336 Mean :2.367 Mean :19.49   
## 3rd Qu.:13.68 3rd Qu.:3.083 3rd Qu.:2.558 3rd Qu.:21.50   
## Max. :14.83 Max. :5.800 Max. :3.230 Max. :30.00   
## Magnesium TotalPhenols Flavanoids NonflavanoidPhenols  
## Min. : 70.00 Min. :0.980 Min. :0.340 Min. :0.1300   
## 1st Qu.: 88.00 1st Qu.:1.742 1st Qu.:1.205 1st Qu.:0.2700   
## Median : 98.00 Median :2.355 Median :2.135 Median :0.3400   
## Mean : 99.74 Mean :2.295 Mean :2.029 Mean :0.3619   
## 3rd Qu.:107.00 3rd Qu.:2.800 3rd Qu.:2.875 3rd Qu.:0.4375   
## Max. :162.00 Max. :3.880 Max. :5.080 Max. :0.6600   
## Proanthocyanins ColorIntensity Hue OD280\_OD315   
## Min. :0.410 Min. : 1.280 Min. :0.4800 Min. :1.270   
## 1st Qu.:1.250 1st Qu.: 3.220 1st Qu.:0.7825 1st Qu.:1.938   
## Median :1.555 Median : 4.690 Median :0.9650 Median :2.780   
## Mean :1.591 Mean : 5.058 Mean :0.9574 Mean :2.612   
## 3rd Qu.:1.950 3rd Qu.: 6.200 3rd Qu.:1.1200 3rd Qu.:3.170   
## Max. :3.580 Max. :13.000 Max. :1.7100 Max. :4.000   
## Proline   
## Min. : 278.0   
## 1st Qu.: 500.5   
## Median : 673.5   
## Mean : 746.9   
## 3rd Qu.: 985.0   
## Max. :1680.0

Let’s review some important assumptions of Multinomial Logistic Regression:

## Independence of Observations

For this demo we will assume that our data is independent. The overview and description of the data set did not give us any reason to believe otherwise, but it is important to note that we are assuming our observations are independent, which is one of the requirements for using Multinomial Logistic Regression.

## Randomness

Based on the data set description, there is no reason to believe that the data was not randomly selected. For the purposes of this demo, we will assume that the data is randomly sampled.

## Linearity

To perform multinomial logistic regression, there must be a linear relationship between the log-odds of the outcome variable and the predictor variables.

## No Multicollinearity:

Multicollinearity occurs when two or more independent variables are highly correlated with one another. This can cause issues in understanding which variables are responsible for the result of the dependent variable.

## Frequency of Categorical Outcomes

It is also important to check the frequencies of occurrences for each outcome category in the data set. If one outcome occurs significantly more than the others (typically 4:1 or 5:1), the model may skew to predict an increased likelihood of the categorical outcome with the most observations. This is a quick check on the assumption of independence of irrelevant alternatives.

# descriptive data  
descrpts <- descriptives(wine.data, freq = TRUE)  
  
descrpts$frequencies

##   
## FREQUENCIES  
##   
## Frequencies of Class   
## ─────────────────────────────────────────────────   
## Class Counts % of Total Cumulative %   
## ─────────────────────────────────────────────────   
## 1 59 33.14607 33.14607   
## 2 71 39.88764 73.03371   
## 3 48 26.96629 100.00000   
## ─────────────────────────────────────────────────

Class 2 has the largest number of observations in our data set but not enough to require adjustment. Although the number of observations for each categorical outcome are not equal, the distribution of the dependent variable does not warrant concern. Since we don’t have a lot of information regarding the three different wine grapes (represented by Class = 1, 2, or 3), we will choose the categorical outcome with the most observations (Class 2) as the reference category.

wine.data$Class <- relevel(wine.data$Class, ref = "2")  
  
# Convert factor levels to valid variable names  
wine.data$Class <- as.factor(wine.data$Class)  
wine.data$Class <- make.names(as.character(wine.data$Class))  
  
# Set levels for the entire dataset  
wine.data$Class <- factor(wine.data$Class, levels = make.names(c("2", "1", "3")))

Before jumping straight to building our Multinomial Logistic Regression model, it is a good practice to separate the data into testing, validation, and training sets. This will enable us to train our model on one subset, and test the model’s performance on a different subset that the model has not yet seen. This practice gives us a better understanding of our model’s generalizability.

In the below code we will randomly assign 60% of our observations to the training subset, 20% to the testing subset, and 20% to the validation subset.

n <- nrow(wine.data)  
  
set.seed(12345)  
  
tvt <- sample(rep(0:2, c(round(n \* 0.2), round(n \* 0.2), n - 2 \* round(n \* 0.2))), n)  
  
dat.train <- wine.data[tvt == 2,]  
dat.valid <- wine.data[tvt == 1,]  
dat.test <- wine.data[tvt == 0,]  
  
# Check that the sum of the three sets is equal to the total number of rows in the data set  
print(n == nrow(dat.train) + nrow(dat.valid) + nrow(dat.test))

## [1] TRUE

table(tvt)

## tvt  
## 0 1 2   
## 36 36 106

To eliminate predictors, we will use stepwise selection. Stepwise selection uses both forward and backward movement to eliminate and add predictors until a final model is produced based upon a specified selection criterion. In this case we will be minimizing AIC.

# Fit multinomial logistic regression model  
model.multinom <- multinom(Class ~ ., data = dat.train)

## # weights: 45 (28 variable)  
## initial value 116.452903   
## iter 10 value 21.399909  
## iter 20 value 1.848586  
## iter 30 value 0.004400  
## final value 0.000075   
## converged

# Create a temporary file to capture output (setting trace = 0 did not suppress the output)  
sink("temp\_output.txt")  
  
# Perform stepwise AIC  
model.stepwise <- stepAIC(model.multinom, direction = "both", trace.lev = 0)  
  
# Display the summary  
summary(model.stepwise)

## Call:  
## multinom(formula = Class ~ Flavanoids + ColorIntensity + Proline,   
## data = dat.train)  
##   
## Coefficients:  
## (Intercept) Flavanoids ColorIntensity Proline  
## X1 -142.39944 0.5257433 14.30966 0.1094293  
## X3 -34.49088 -25.1572390 19.52752 -0.0223283  
##   
## Std. Errors:  
## (Intercept) Flavanoids ColorIntensity Proline  
## X1 0.1514331 2.169064 1.538274 0.003813835  
## X3 0.5347417 1.885681 2.799196 0.030516384  
##   
## Residual Deviance: 2.737849   
## AIC: 18.73785

Stepwise selection produced a model with three predictors: Flavanoids, ColorIntensity, and Proline. The AIC for this model on the training data was 18.7379.

## Interpreting Coefficients:

Coefficients: (Intercept) Flavanoids ColorIntensity Proline X1 -142.39944 0.5257433 14.30966 0.1094293 X3 -34.49088 -25.1572390 19.52752 -0.0223283

In multinomial logistic regression, one category is chosen as the reference category, and the coefficients for the other categories are expressed relative to this reference category. In our model, Class 2 was chosen as the reference category, therefore we will only see coefficient estimates for Classes 1 and 3.

For each Class (X1, X3 = Class 1, Class 3), there are intercepts and coefficients for the predictor variables included in our model (Flavanoids, ColorIntensity, and Proline).

The coefficient values correspond to the estimated effect that each predictor has on the log-odds of each observation belonging to the respective class, relative to the reference class.

# For example, in X1 (Class 1):

Keeping all other predictors constant, if the value of Flavanoids increases by 1 unit, the log-odds of the observation being categorized as X1 (Class 1) increases by 0.5257433 more than the log-odds of the observation being categorized as X2 (Class 2). To put it more generally, higher levels of Flavanoids result in a higher probability of classifying an observation as Class 1 compared to Class 2.

The coefficient for ColorIntensity is 14.30966, indicating that as ColorIntensity increases by one unit, the log-odds of the observation being categorized as X1 (Class 1) increases by 14.30966 more than the log-odds of the observation being categorized as X2 (Class 2).

The coefficient for Proline is 0.1094293, indicating that as Proline increases by one unit, the log-odds of the observation being categorized as X1 (Class 1) increases by 0.1094293 more than the log-odds of the observation being categorized as X2 (Class 2).

# For X3 (Class 3):

Keeping all other predictors constant, if the value of Flavanoids increases by 1 unit, the log-odds of the observation belonging to X3 (Class 3) decreases by 25.1572390 compared to the log-odds of the observation belonging to X2 (Class 2).

If the value of ColorIntensity increases by 1 unit, the log-odds of the observation belonging to X3 (Class 3) increases by 19.52752 compared to the log-odds of the observation belonging to X2 (Class 2).

If the value of Proline increases by 1 unit, the log-odds of the observation belonging to X3 (Class 3) decreases by 0.0223283 compared to the log-odds of the observation belonging to X2 (Class 2).

## Predictor Significance

Showing the p-values for each class and predictor in our model, we can confirm that each variable is significant:

# Convert the model to tidy format  
tidy\_model <- broom::tidy(model.stepwise)  
  
# Select specific columns using dplyr::select  
p\_vals <- tidy\_model %>%  
 dplyr::select(y.level, term, p.value)  
  
p\_vals

## # A tibble: 8 × 3  
## y.level term p.value  
## <chr> <chr> <dbl>  
## 1 X1 (Intercept) 0   
## 2 X1 Flavanoids 8.08e- 1  
## 3 X1 ColorIntensity 1.37e- 20  
## 4 X1 Proline 4.70e-181  
## 5 X3 (Intercept) 0   
## 6 X3 Flavanoids 1.33e- 40  
## 7 X3 ColorIntensity 3.03e- 12  
## 8 X3 Proline 4.64e- 1

The p-values above show the significance level of the effect of each predictor variable in differentiating each category (X1 and X3) from the reference category (X2).

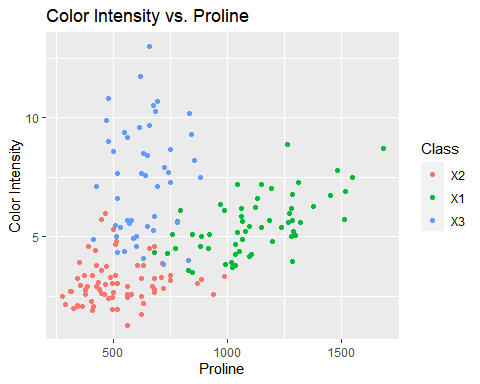
Most of the p-values for the predictors in our model are below the statistical significance threshold of .05. The two exceptions are Flavanoids for X1 (~0.8085) and Proline for X3 (~.4644). These predictors are included in our model because the p-values for their counterparts are significant (Flavanoids for X3 = 1.333028e-40 and Proline for X1 4.702436e-181).

The p-value of Flavanoids for X1 is above .05. This indicates that the level of Flavanoids does not have a significant impact on predicting whether an observation belongs to Class 1 (X1) or Class 2 (reference category). The p-value of Proline for X3 is above .05. This indicates that the level of Proline does not have a significant impact on predicting whether an observation belongs to Class 3 (X3) or Class 2 (reference category).

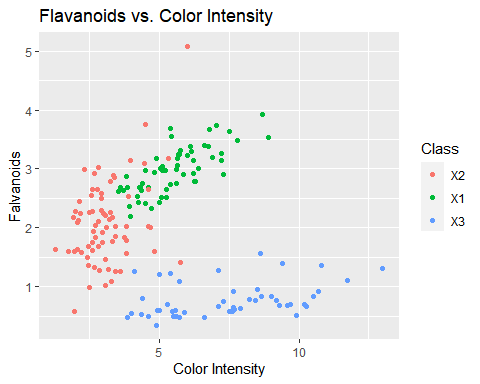
## Graphs

Now that we have checked and cleaned up the data, let’s look at a few graphs of the data using different variables for x and y, coloring by class to see if it appears that class can be predicted by variables.

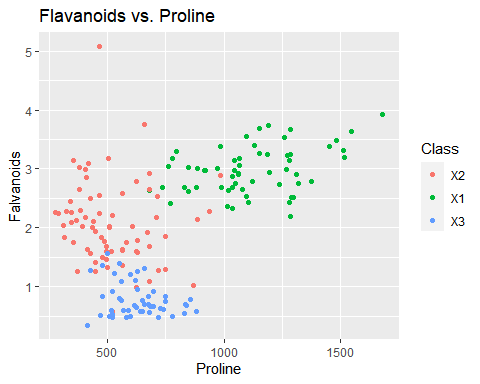
ggplot(data=wine.data,aes(x=Proline,y=ColorIntensity, color=Class))+geom\_point()+  
 labs(  
 title = "Color Intensity vs. Proline",  
 x = "Proline",  
 y = "Color Intensity"  
 )



ggplot(data=wine.data,aes(x=ColorIntensity ,y=Flavanoids, color=Class))+geom\_point() +  
 labs(  
 title = "Flavanoids vs. Color Intensity",  
 x = "Color Intensity",  
 y = "Falvanoids"  
 )



ggplot(data=wine.data,aes(x=Proline,y=Flavanoids, color=Class))+ geom\_point() +  
 labs(  
 title = "Flavanoids vs. Proline",  
 x = "Proline",  
 y = "Falvanoids"  
 )



As displayed in the graphs above, it appears that the class of wine does have separation when plotting the chosen predictor variables from our model. This likely indicates that our multinomial logistic regression model is better at predicting the class of wine than the intercept only model; we will test this directly later on.

## Checking our Model for Multicollinearity

Before we continue on to predictions, let’s use the model we just created to verify that there is no significant multicollinearity in our chosen model:

Variance Inflation Factors (VIFs) can be used to determine the level of multicollinearity that exists between predictors in a regression model. Larger VIFs indicate higher levels of multicollinearity. Generally, VIFs below 5 are acceptable values for the predictors in a multinomial logistic regression.

The vif() function in R can’t handle multinomial logistic regression models, however we can input our model as a multiple linear Regression strictly for the purposes of checking for multicollinearity with the vif() function.

# Convert class variable from a factor to a numeric to test for the VIF values  
dat.train.vif <- dat.train  
dat.train.vif$Class <- as.numeric(dat.train.vif$Class)  
  
# Use same predictors and data as in model.stepwise  
model.vif <- lm(Class ~ Flavanoids + ColorIntensity + Proline, data = dat.train.vif)  
  
# Calculate VIF values for predictors  
vif\_values <- vif(model.vif)  
  
# Display VIF values  
print(vif\_values)

## Flavanoids ColorIntensity Proline   
## 1.426135 1.259120 1.513569

The VIFs for each of the predictors in our model are close to 1, indicating that there is no strong evidence of multicollinearity.

|  |
| --- |
| ## Predictions |
| Now that we have produced our model, we can use it to make predictions on the validation data (which the model has not yet seen) and assess its performance. |
| The function below computes a confusion matrix for a given model and data set: |
| ```r calc\_confusions <- function(model, data) { |
| # Make class predictions pred.class <- predict(model, newdata = data) |
| # Convert factor levels levels(pred.class) <- c(“Class 2”, “Class 1”, “Class 3”) levels(data$Class) <- c(“Class 2”, “Class 1”, “Class 3”) |
| # Calculate accuracy ConfuseMatrix <- table(Actual = data$Class, Predicted = pred.class) |
| return(ConfuseMatrix) } ``` |
| r confusion.valid <- calc\_confusions(model.stepwise, dat.valid) confusion.valid |
| ## Predicted ## Actual Class 2 Class 1 Class 3 ## Class 2 12 1 0 ## Class 1 0 11 0 ## Class 3 0 0 12 |
| # Interpreting a 3 x 3 confusion matrix: |
| A 3 x 3 confusion matrix can be used when the outcome variable has 3 classes as opposed to 2. It extends the concepts of True Positives, True Negatives, False Negatives, and False Positives to 3 classes. |
| The main diagonal represents correct classifications. The counts that are not on the main diagonal represent incorrect classifications. For the validation data, our model made 35 out of 36 correct classifications. There was 1 incorrect classification that is shown in the first row of the second column, indicating there was 1 instance in which the model incorrectly classified a Class 2 observation as a Class 1 observation. |
| The Actual distribution of our 36 observations in dat.valid are as follows: Class 2: 13 Class 1: 11 Class 3: 12 |
| The Predicted distribution of our 36 observations in dat.valid are as follows: Class 2: 12 Class 1: 12 Class 3: 12 |
| # Accuracy Now we can compute the overall accuracy, as well as the accuracy for each class using the confusion matrix that we produced. Accuracy can be easily computed from the 3x3 confusion matrix. |
| Accuracy: The proportion of correctly predicted instances divided by the total instances. |
| ```r # Overall Accuracy accuracy.valid <- sum(diag(confusion.valid))/sum(confusion.valid) |
| # Class 2 accuracy.valid.class2 <- confusion.valid[1, 1] / sum(confusion.valid[1, ]) |
| # Class 1 accuracy.valid.class1 <- confusion.valid[2, 2] / sum(confusion.valid[2, ]) |
| # Class 3 accuracy.valid.class3 <- confusion.valid[3, 3] / sum(confusion.valid[3, ]) |
| # Print or use the accuracy values as needed cat(“Overall Accuracy:”, accuracy.valid, “”) ``` |
| ## Overall Accuracy: 0.9722222 |
| r cat("Accuracy for Class 2:", accuracy.valid.class2, "\n") |
| ## Accuracy for Class 2: 0.9230769 |
| r cat("Accuracy for Class 1:", accuracy.valid.class1, "\n") |
| ## Accuracy for Class 1: 1 |
| r cat("Accuracy for Class 3:", accuracy.valid.class3, "\n") |
| ## Accuracy for Class 3: 1 |
| r #compare accuracy of test to frequencies in data. For a good model, we want the accuracy to be above the percent of occurrences in each class: descrpts$frequencies |
| ## ## FREQUENCIES ## ## Frequencies of Class ## ───────────────────────────────────────────────── ## Class Counts % of Total Cumulative % ## ───────────────────────────────────────────────── ## 1 59 33.14607 33.14607 ## 2 71 39.88764 73.03371 ## 3 48 26.96629 100.00000 ## ───────────────────────────────────────────────── For the Validation Data the overall accuracy was 97.22%. |
| Now we can repeat this process on the testing data for additional metrics that relate to our model’s performance. |
| r confusion.test <- calc\_confusions(model.stepwise, dat.test) confusion.test |
| ## Predicted ## Actual Class 2 Class 1 Class 3 ## Class 2 12 1 2 ## Class 1 1 13 0 ## Class 3 0 0 7 |
| ```r # Overall Accuracy accuracy.test <- sum(diag(confusion.test))/sum(confusion.test) |
| # Class 2 accuracy.test.class2 <- confusion.test[1, 1] / sum(confusion.test[1, ]) |
| # Class 1 accuracy.test.class1 <- confusion.test[2, 2] / sum(confusion.test[2, ]) |
| # Class 3 accuracy.test.class3 <- confusion.test[3, 3] / sum(confusion.test[3, ]) |
| # Print or use the accuracy values as needed cat(“Accuracy:”, accuracy.test, “”) ``` |
| ## ## Overall Accuracy: 0.8888889 |
| r cat("Accuracy for Class 2:", accuracy.test.class2, "\n") |
| ## Accuracy for Class 2: 0.8 |
| r cat("Accuracy for Class 1:", accuracy.test.class1, "\n") |
| ## Accuracy for Class 1: 0.9285714 |
| r cat("Accuracy for Class 3:", accuracy.test.class3, "\n") |
| ## Accuracy for Class 3: 1 |
| r #compare accuracy of test to frequencies in data. For a good model, we want the accuracy to be above the percent of occurences in each class: descrpts$frequencies |
| ## ## FREQUENCIES ## ## Frequencies of Class ## ───────────────────────────────────────────────── ## Class Counts % of Total Cumulative % ## ───────────────────────────────────────────────── ## 1 59 33.14607 33.14607 ## 2 71 39.88764 73.03371 ## 3 48 26.96629 100.00000 ## ───────────────────────────────────────────────── The Actual distribution of our 36 observations in dat.test are as follows: Class 2: 15 Class 1: 14 Class 3: 7 |
| The Predicted distribution of our 36 observations in dat.test are as follows: Class 2: 13 Class 1: 14 Class 3: 9 |
| The Model performed less well on the testing data. It correctly classified 32 out of 36 observations with an overall accuracy of 88.89%. |
| The lowest accuracy score by Class was 80%, for Class 2. It appears our model’s weakest attribute is its tendency to classify a Class 2 observation as a Class 1 or Class 3 observation. However, 80% is still well above the 39.88% of Class 2 occurrences in our data so it appears to be a good fit. |
| ## Misclassification Error |
| We can take 1 - accuracy to compute misclassification rate for the validation and test data. |
| ```r missClass.valid <- 1 - accuracy.valid |
| missClass.test <- 1 - accuracy.test |
| # Print or use the misclassification values as needed cat(“rate for validation data::”, missClass.valid, “”) ``` |
| ## ## Misclassification rate for validation data:: 0.02777778 |
| r cat("\nMisclassification rate for test data::", missClass.test, "\n\n") |
| ## ## Misclassification rate for test data:: 0.1111111 |

# Likelihood Ratio Test

As a final test, we can conduct a likelihood ratio test that compares the performance of our selected multinomial logistic regression model to the null model (the intercept-only model). This is a common test that is used to determine if a multinomial logistic regression model is necessary for predicting a categorical outcome.

# Fit the null model (intercept only)  
null\_model <- multinom(Class ~ 1, data = dat.train)

## # weights: 6 (2 variable)  
## initial value 116.452903   
## final value 115.045102   
## converged

# Perform likelihood ratio test  
lr\_test <- lrtest(null\_model, model.stepwise)  
  
# Display the results  
print(lr\_test)

## Likelihood ratio test  
##   
## Model 1: Class ~ 1  
## Model 2: Class ~ Flavanoids + ColorIntensity + Proline  
## #Df LogLik Df Chisq Pr(>Chisq)   
## 1 2 -115.045   
## 2 8 -1.369 6 227.35 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The results of the likelihood ratio test show that our multinomial logistic regression model is significantly better at classifying the response variable than the null model (the intercept-only model).

With p < .05, we reject the null hypothesis that the more complex model (our multinomial logistic regression model) is not a significantly better fit than the null model. We conclude that our final model is a good fit and does not contain unnecessary predictors.

Final Model: Class ~ Flavanoids + ColorIntensity + Proline

## Conclusion

Using Stepwise selection to minimize AIC, 10 predictors were eliminated from our model. The predictors that we found to be the most significant for predicting the Class of Wine were Flavanoids, ColorIntensity, and Proline.

On the validation data, our model correctly classified 35 out of 36 observations (97.22% accuracy). On the testing data, our model correctly classified 32 out of 36 observations (88.89% accuracy). If we aggregate the results of our testing, our model correctly classified 67 out of 72 observations for an aggregate accuracy of 93.06%.

In general, our model’s weakest attribute was its ability to correctly classify Class 2 observations. Out of 5 total misclassifications, 4 were Class 2 observations that were erroneously classified as Class 1 or Class 3 observations.

The results of the likelihood ratio test showed that our model was a much better fit at predicting the categorical outcome than the null model.

In summary, our multinomial logistic regression model was a fairly strong classifier for our data. We were able to correctly predict the cultivar of wine (grape type) based on the values of Flavanoids, ColorIntensity, and Proline for the overwhelming majority of our tested observations.

To revisit our research question: Can we predict the cultivar of wine (Class 1, 2, or 3) based on the values of chemical properties that were recorded in this data using Multinomial Logistic Regression? Yes - We were able to use Multinomial Logistic Regression to correctly predict the cultivar of wine with an aggregated accuracy of 93.06% using Flavanoids, ColorIntensity, and Proline as predictors.

## Outside References:

Outside Reference 1: <https://www.statstest.com/multinomial-logistic-regression/>

Outside Reference 2: <https://www.kaggle.com/code/saurabhbagchi/multinomial-logistic-regression-for-beginners>

Outside Reference 3: <https://statistics.laerd.com/spss-tutorials/multinomial-logistic-regression-using-spss-statistics.php>

Outside Reference 4: <https://www.ibm.com/docs/en/spss-statistics/29.0.0?topic=regression-multinomial-logistic>

Outside Reference 5: <https://www.mygreatlearning.com/blog/multinomial-logistic-regression/>

Outside Reference 6: <https://stats.oarc.ucla.edu/r/dae/multinomial-logistic-regression/>

Outside Reference 7: <https://online.stat.psu.edu/stat504/lesson/8>

Outside Reference 8 (R reference): <https://bookdown.org/chua/ber642_advanced_regression/multinomial-logistic-regression.html> <https://bookdown.org/sarahwerth2024/CategoricalBook/multinomial-logit-regression-r.html>

Outside Reference 9 (R reference): <https://rpubs.com/rslbliss/r_logistic_ws>

Outside Reference 10 (R reference): <https://www.youtube.com/watch?v=c78eMWw43I0>