ISYE 7406 – Homework 3

Shehzad Anwar

September 28, 2025

Introduction

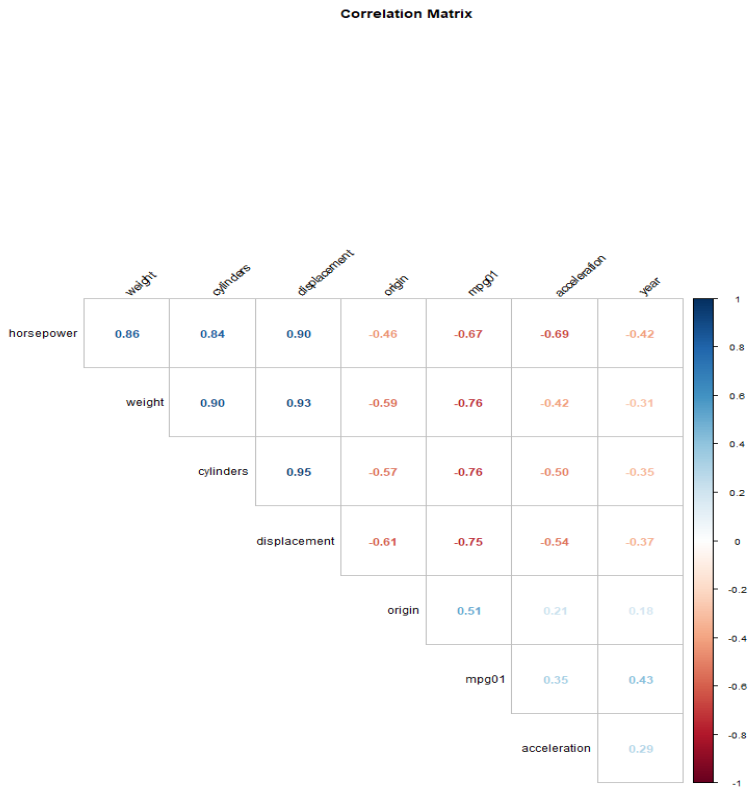
For this assignment, the *Auto.csv* dataset was studied and analyzed. The goal of this assignment was to familiarize ourselves with different methods of classification, including LDA, QDA, Naive Bayes, Logistic Regression and KNN. This was done by testing and determining the training and testing error, the cross validation error, and the variance and comparing each of them against each other.

Exploratory Data Analysis

The *Auto.csv* dataset consists of 398 rows and 9 columns, i.e. characteristics: mpg, cylinders, displacement, horsepower, weight, acceleration, year, origin and the car names. The *car names* column was deleted to avoid issues with numerical analysis, leaving us with 8 columns to work with. The mpg column is utilized to create a binary variable *mpg01*, and this is used to indicate whether or not that specific car model has a higher fuel efficiency than the median.

On the next page, an image of a group of boxplots showcase 'How Car Characteristics Relate to MPG'. These boxplots indicate to us which features are going to be useful in predicting the *mpg01*. What we're looking for here is a clear separation between the boxes in each boxplot which would tell us how strong (or weak) a predictor variable is. The features with a clear separation, from the eye test, would be the cylinders, displacement, weight and horsepower. While there is a lack of separation in the *year* plot, it is, maybe unsurprising, to see that the models from later years tended to have better mileage per gallon than the former. Acceleration would be the pick for the worst predictor, seeing as how the two boxes overlap.

To confirm how close the correlation is between each car characteristic, a correlation heatmap was used using the 'number' format to clearly show the value of the correlation and can be seen on the next page. Homing in on the *mpg01* column, we can see that it has a strong negative correlation with, in order from most negative to least, cylinders, weight, displacement and horsepower, which confirms that they are the strongest predictors. It is also interesting how strong the correlation between each of those same features are, ranging from a positive 0.86-0.95, suggesting high colinearity. The correlation between mpg01 and acceleration is 0.35, a low positive index, which indicates that the prior assumption that it is a weak predictor is correct, with the year edging higher at 0.43. Despite a somewhat mediocre correlation value, I still view year as a usable predictor mainly because it is only natural that as there is technological advancements and better cars produced over time, the fuel efficiency would improve as well, so one could make the assumption that a newer car has a higher mileage per gallon than an older car, and they'd be completely justified.



Methodology

The first act that was undertaken (besides loading in the data to a variable name, *auto*) was the finding of the median of the mileage per gallon (mpg) column, which was 22.75. This was done in order to create the binary variable *mpg01*, a column which would essentially make the initial *mpg* column that came with the dataset redundant and would only complicate calculations and observations as another numeric value. As such, it was removed, and mpg01 was converted into a factor to ensure proper categorization. In order to view the group of boxplots, the dataset needed to be converted from a wide dataset to a long format, so that all the information would be displayed. I'd decided to do a correlation heatmap (with the number format) as well to delve further into how closely tied these characteristics are, and as such, needed to convert mpg01 into a numeric value so that the calculations didn't ignore it.

The correlation heatmap and the boxplots will indicate the strong predictors and the weak predictors, and we will take these predictors as our variables for the classification

methods we will working with. Overall, cylinders, weight, displacement and horsepower were chosen as the strong predictors while acceleration was the weakest.

There were five classification methods that were performed, which included (in order as I completed them) Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naive Bayes, and K Nearest Neighbor.

While I made confusion matrices for the models, ultimately I thought they were redundant to include as I'm running a cross validation on the models anyway.

I chose to have the training and test set have a split of 2/3. I chose this number because I feel it provides a decent split for either side to work with, and that a higher number could possibly cause us to miss out on valuable information potentially, either as a result of the training set being too broad or the test set not having enough to go through to find anything meaningful.

The best k value that I decided to consider was k = 3, and that is mainly because (as will be seen in the next section) that was the value that gave me the least average testing error overall at 8.28% out of several different values (going up to 30, increasing by increments of 2).

To make the comparison more rigorous, I decided to run 100 cross validation trials on each of the methods.

Results

After running 100 Monte Carlo cross validation trials on each classification method, the results of the average test error are as follows:

Method	Test Error (Averaged over 100 CV trials)	Standard Deviation
LDA	0.1089	0.0163
QDA	0.1133	0.0147
Naive Bayes	0.1098	0.0141
Logistic Regression	0.1034	0.0217
KNN ($k = 3$)	0.0828	0.0264

As can be seen from the table, the KNN classification method brought the least amount of test error overall, followed by Logistic Regression, LDA, Naive Bayes and lastly QDA. It is important to note that LDA, QDA and Logistic Regression make assumptions about the

data being analyzed, while the KNN doesn't. Below is the *k* values and their average test error amount.

K Val	Average Test Error (over 100 CV trials)	Standard Deviation
1	0.0833	0.026
3	0.0828	0.0264
5	0.0863	0.0284
7	0.0882	0.0265
9	0.0923	0.0273
11	0.0979	0.0264
13	0.0998	0.0242
15	0.1015	0.023
17	0.1003	0.0237
19	0.1002	0.0234
21	0.1	0.024
23	0.1	0.024
25	0.0995	0.0237
27	0.0992	0.0237
29	0.0984	0.0235

As mentioned before, k = 3 is the best k value out of these, and the test error only increases the greater the value is (until it reaches 25 and later, at which point it plateaus, still higher than our chosen value, though).

Conclusion

In conclusion, out of all the results and the analysis we've conducted on the *Auto.csv* dataset, the KNN model would have to be named the best model based on having the lowest average testing error among all the different models that were studied. Seeing as how all four of the other models have a test error average above 0.1, KNN had a significant difference over the alternative models, and while its standard deviation is greater than its counterparts, it would still be the best overall model.

Appendix

hw3.R:

```r
library (ggplot2)
library (dplyr)
library (tidyr)
library (corrplot)
library (MASS)
library (e1071)
library (class)

# Part A: Let's load the data
auto <- read.csv("Auto.csv", header = TRUE)

# Part B: Let's create the binary variable 'mpg01'
median_mpg <- median(auto$mpg)
print(paste("The value of the median mpg is: ",median_mpg))

# We're going to create a new column for mpg01 and set a condition: If mpg > median_mpg, return 1, else return 0
# and since the mpg would be redundant, let's drop it.
auto$mpg01 <- ifelse(auto$mpg > median_mpg, 1, 0)
auto <- auto %>% dplyr::select(-mpg)

# Part C: Exploratory Data Analysis
# Since mpg01's value is either a 1 or a 0, let's not consider it as an integer value,
# but rather a factor.
auto$mpg01 <- as.factor(auto$mpg01)

# Convert the data from wide to long format so that we can plot it on the same graph.
auto_long <- auto %>% pivot_longer(cols = -mpg01, # Grab all columns except for mpg01
    names_to = "characteristic", values_to = "value")

eda_plot <- ggplot(auto_long, aes(x = mpg01, y = value, fill = mpg01)) + geom_boxplot() +
    facet_wrap(~characteristic, scales = "free_y") + labs(title = "How Car Characteristics Relate to MPG",
    x = "MPG (0 = Less than Median, 1 = Greater than Median)", y = "Value") + theme_minimal()

# Now, let's do a correlation plot to compare the variables.
auto$mpg01 <- as.numeric(as.character(auto$mpg01)) # Need to convert mpg01 back to a numeric to include it in the calculation
```

```r
num_val <- auto %>% select_if(is.numeric) # Let's only take the numeric vals.
corr_mat <- cor(num_val)
corrplot(corr_mat, method = "number", type = "upper", order = "hclust", tl.col =
"black", tl.srt = 45, diag = FALSE, title = "Correlation Matrix", mar=c(0,0,1,0))

# Part D: Let's split the data into training and testing
set.seed(7406)

# Implement a 100 trial Monte Carlo Cross Validation
trials <- 100
error_num <- c()

for (i in 1:trials) { # loop to run 100 times
train_idx <- sample(1:nrow(auto), 2/3 * nrow(auto)) # using sample code from
class, using 2/3 because it gives a decent amount of data to either set
train_set <- auto[train_idx,]
test_set <- auto[-train_idx,]

train_set$mpg01 <- as.factor(train_set$mpg01)
test_set$mpg01 <- as.factor(test_set$mpg01)

# Part E: Let's perform classification methods on the data
# 4: Logistic Regression
auto_log <- glm(mpg01 ~ cylinders + weight + displacement + horsepower + year,
data = train_set, family = binomial)

# Test set predictions
test_prob <- predict(auto_log, test_set, type = "response")
test_class <- ifelse(test_prob > 0.5, 1, 0) # if prob > 0.5, return 1, else 0

# Build a confusion matrix
confusion <- table(test_class, test_set$mpg01)

# Test error for Logistic Regression
log_err <- mean(test_class != test_set$mpg01)
error_num <- c(error_num, log_err) # Run CV
}

avg_log_err <- mean(error_num)
std_log_err <- sd(error_num)
print(paste("The average test error for Logistic Regression over", trials,
"trials:",round(avg_log_err, 4)))
```

```r
print(paste("The standard deviation of the test error for Logistic Regression: ",
round(std_log_err, 4)))

# 1: LDA
for (i in 1:trials) { # loop to run 100 times
lda_model <- lda(mpg01 ~ cylinders + weight + displacement + horsepower + year,
data = train_set)
lda_pred <- predict(lda_model, test_set)
lda_err <- lda_pred$class # we're separating the predicted classes from the rest
of the model

# LDA Confusion matrix
lda_confusion <- table(lda_err, test_set$mpg01)

# LDA Test error
lda_test_err <- mean(lda_err != test_set$mpg01)
error_num <- c(error_num, lda_test_err) # Run CV
}

avg_lda_err <- mean(error_num)
std_lda_err <- sd(error_num)
print(paste("The average test error for LDA over", trials,
"trials:",round(avg_lda_err, 4)))
print(paste("The standard deviation of the test error for LDA: ",
round(std_lda_err, 4)))

# 2: QDA
for (i in 1:trials) { # loop to run 100 times
qda_model <- qda(mpg01 ~ cylinders + weight + displacement + horsepower + year,
data = train_set)
qda_pred <- predict(qda_model, test_set)
qda_err <- qda_pred$class

# QDA Confusion matrix
qda_confusion <- table(qda_err, test_set$mpg01)

# QDA Test error
qda_test_err <- mean(qda_err != test_set$mpg01)
error_num <- c(error_num, qda_test_err) # Run CV
}

avg_qda_err <- mean(error_num)
std_qda_err <- sd(error_num)
```

```r
print(paste("The average test error for QDA over", trials,
"trials:",round(avg_qda_err, 4)))
print(paste("The standard deviation of the test error for QDA: ",
round(std_qda_err, 4)))

# 3: Naive Bayes
for (i in 1:trials) { # loop to run 100 times
nb_model <- naiveBayes(mpg01 ~ cylinders + weight + displacement + horsepower +
year, data = train_set)
nb_pred <- predict(nb_model, test_set)

# Naive Bayes Confusion matrix
nb_confusion <- table(nb_pred, test_set$mpg01)

# Naive Bayes Test error
nb_test_err <- mean(nb_pred != test_set$mpg01)
error_num <- c(error_num, nb_test_err) # Run CV
}

avg_nb_err <- mean(error_num)
std_nb_err <- sd(error_num)
print(paste("The average test error for Naive Bayes over", trials,
"trials:",round(avg_nb_err, 4)))
print(paste("The standard deviation of the test error for Naive Bayes: ",
round(std_nb_err, 4)))

# 5: KNN
k_values <- seq(1, 30, 2)
k_error_rate <- matrix(nrow = trials, ncol = length(k_values))
colnames(k_error_rate) <- paste("k = ", k_values)

for (i in 1:trials) { # This is the outer loop which will do the cross validation
trials
    train_idx <- sample(1:nrow(auto), 2/3 * nrow(auto))
    train_set <- auto[train_idx,]
    test_set <- auto[-train_idx,]

    train_pred <- train_set %>% dplyr::select(cylinders, weight, displacement,
horsepower, year)
    test_pred <- test_set %>% dplyr::select(cylinders, weight, displacement,
horsepower, year)
    train_labels <- train_set$mpg01
    test_labels <- test_set$mpg01
```

```
    for (k in 1:length(k_values)) { # This is the inner loop that handles the k
values
        k_val = k_values[k]
        knn_mod <- knn(train = scale(train_pred), test = scale(test_pred), cl =
train_labels, k = k_val)
        k_error_rate[i, k] <- mean(knn_mod != test_labels)
    }
}

avg_knn_err <- colMeans(k_error_rate)
std_knn_err <- apply(k_error_rate, 2, sd)
print(paste("The average test error for KNN over", trials,
"trials:",round(avg_knn_err, 4)))
print(paste("The standard deviation of the test error for KNN: ",
round(std_knn_err, 4)))
```