ILLINOIS TECH

Homework 2

EDUARDO GALEOTE ESCALERA

A20552496

CSP 571

Data Preparation and Analysis

September 2023

Contents

1 Recitation Exercises

1.1 Chapter 4

1.1.1 Excercise 4

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

a) *Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55,0.65]. On average, what fraction of the available observations will we use to make the prediction?*
Assuming $x \in [0.05, 0.95]$, then intervals: $[x - 0.05, x + 0.05]$, so $length = 0.1$. On average 10% of the observations would be available to make a prediction for the test observation.

b) *Now suppose that we have a set of observations, each with measurements on p = 2 features, X1 and X2. We assume that (X1 , X2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10 % of the range of X1 and within 10 % of the range of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1 = 0.6 and X2 = 0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3,0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?*
Assuming $x \in [0.05, 0.95]$, $x1_{length} \times x2_{length} = 0.01$. Therefore, 1% of the available observations would be used to make a prediction.

c) *Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?*
If p=100; $0.1^p \times 100 = 0.1^{100} \times 100\%$ of the observations are available.

d) *Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.*
As the number of predictors increase, the fraction of observations available to make a prediction is reduced exponentially.

e) *Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.*

- $p = 1; d(length) = 0.1^{1/1} = 0.1$

- $p = 2; d(length) = 0.1^{1/2} = 0.32$
- $p = 100; d(length) = 0.1^{1/100} = 0.977$

As p increases the side length converges to 1, and this shows that the hypercube centered around the test observation with 10% of the test observation needs to be nearly the same size as the hypercube with all the observations.

1.1.2 Excercise 6

Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = 6, \hat{\beta}_1 = 0.05, \hat{\beta}_2 = 1$.

a) *Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.*

$$P(X) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}$$

$$P(X) = \frac{\exp(-0.5)}{1 + \exp(-0.5)} = 0.38$$

b) *How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class?*

$$\log(\frac{P(X)}{1 - P(X)}) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

$$\log(\frac{0.5}{1 - 0.5}) = -6 + 0.05X1 + 3.5$$

$$X_1 = 50 hours.$$

1.1.3 Excercise 7

Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$*, while the mean for those that didn't was* $\bar{X} = 0$*. In addition, the variance of X for these two sets of companies was* $\sigma^2 = 36$*. Finally, 80 % of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was* $X = 4$ *last year.*

$$P(Y = yes|X = 4) = \frac{\pi_{yes} f_{yes}(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} = \frac{\pi_{yes} \exp(-1/2\sigma^2(x - \mu_{yes})^2)}{\sum_{l=1}^{K} \pi_l \exp(-1/2\sigma^2(x - \mu_l)^2)}$$

$$P(Y = yes|X = 4) = \frac{0.8 \times \exp(-0.5)}{0.8 \times \exp(-0.5) + 0.2 \times \exp(-16/72)}$$

$$P(Y = yes|X = 4) = 0.75$$

1.1.4 Excercise 9

This problem has to do with odds.

a) *On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?*
$Odds = \frac{P(X)}{1 - P(X)}$
$P(X) = \frac{0.37}{1.37} = 0.27$
27% of people with odds of 0.37 will default.

b) *Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?*
$Odds = \frac{0.16}{1-0.16} = 0.19$

1.2 Chapter 5

1.2.1 Excercise 2

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

a) *What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.*
When using random sampling with replacement to obtain a bootstrap sample from n observations: If n=3, with observations: 1,2,3 we have: Total permutations: $n^n = n^3 = 27$. Total times an observation will appear in a particular order(say as the first observation): $n^{n-1} = n^2 = 9$. So, the probability 1st bootstrap observation is the jth observation is: $1/n$. Therefore, the probability the 1st bootstrap observation is not the jth observation is: $1 - 1/n$.

b) *What is the probability that the second bootstrap observation is not the jth observation from the original sample?*
$1 - 1/n$

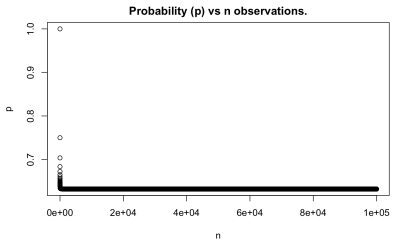
c) *Argue that the probability that the jth observation is not in the bootstrap sample is* $(1 - \frac{1}{n})^n$.
Total times an observation will not appear in a bootstrap sample: $(n-1)^n$. Therefore, the probability that an observation is not in the bootstrap sample: $\frac{(n-1)^n}{n^n} = (1 - 1/n)^n$.

d) *When n = 5, what is the probability that the jth observation is in the bootstrap sample?*
$p = 1 - (1 - 1/5)^5 = 0.672$

e) *When n = 100, what is the probability that the jth observation is in the bootstrap sample?*
$p = 1 - (1 - 1/100)^{100} = 0.634$

f) *When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?*
$p = 1 - (1 - 1/10000)^{10000} = 0.632$

g) *Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.*



Probability decreases rapidly to an asymptote around 0.63 as n increases.

h) *We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.* 0.6338. The result is similar to the value calculated using the formula in e.

1.2.2 Excercise 3

We now review k-fold cross-validation.

a) *Explain how k-fold cross-validation is implemented.*
In k-fold cross validation, a training set is divided into k groups of equal size. The first group is treated as the validation set, and the model is fit on the remaining k-1 groups. The MSE is calculated using the validation set. This procedure is repeated k times and on each occasion the validation set and training sets will be different than the previous one. We then take the average of all the MSE's as the final MSE.

b) *What are the advantages and disadvantages of k-fold cross-validation relative to: i. The validation set approach? and ii. LOOCV?*
i. Less variance in test error estimate. A more accurate test error, as entire dataset is used.
ii. Computational advantage, as less computing resources required than LOOCV. Only k models need to be fitted, unlike with LOOCV where n models need fitting. Higher bias than LOOCV, as fewer observations are used, but tends to have lower variance.

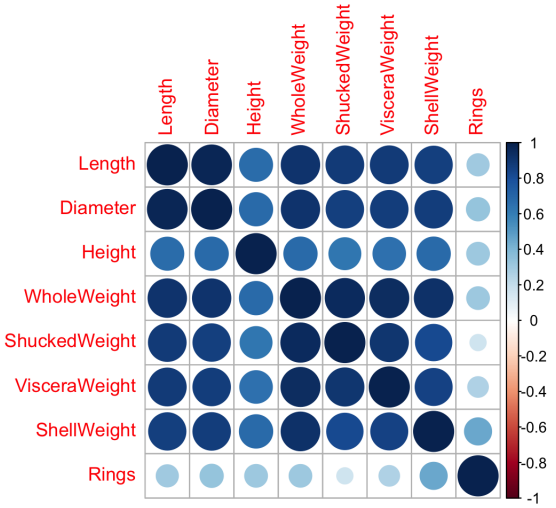
2 Practicum Problems

2.1 Problem 1

Load the abalone sample dataset from the UCI Machine Learning Repository (abalone.data) into R using a dataframe. Remove all observations in the Infant category, keeping the Male/Female classes. Using the caret package, use createDataPartition to perform an 80/20 test-train split (80% training and 20% testing). Fit a logistic regression using all feature variables via glm, and observe which predictors are relevant. Do the confidence intervals for the predictors contain 0 within the range? How does this relate to the null hypothesis? Use the confusionMatrix function in caret to observe testing results (use a 50% cutoff to tag Male/Female) - how does the accuracy compare to a random classifier ROC curve? Use the corrplot package to plot correlations between the predictors. How does this help explain the classifier performance?

```r
# Load required libraries
library(caret)
library(corrplot)
library(pROC)

# Load the abalone dataset from the UCI Machine Learning Repository
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
abalone <- read.csv(url, header = FALSE)

# Add column names to the dataset
colnames(abalone) <- c("Sex", "Length", "Diameter", "Height", "WholeWeight", "ShuckedWeight", "VisceraWeight", "ShellWeight", "Rings")

# Filter out observations in the "Infant" category
abalone <- abalone[abalone$Sex %in% c("M", "F"), ]

# Recode "Sex" to a binary variable
abalone$Sex <- ifelse(abalone$Sex == "M", 1, 0)

# Perform an 80/20 test-train split
set.seed(123)  # for reproducibility
trainIndex <- createDataPartition(abalone$Sex, p = 0.8, list = FALSE)
trainData <- abalone[trainIndex, ]
testData <- abalone[-trainIndex, ]

# Convert "Sex" to a factor with levels 0 and 1
trainData$Sex <- factor(trainData$Sex, levels = c(0, 1))
testData$Sex <- factor(testData$Sex, levels = c(0, 1))

# Fit a logistic regression model
model <- glm(Sex ~ ., data = trainData, family = binomial)

# Summary of the logistic regression model
summary(model)

# Confidence intervals for the predictors
confint(model)

# Check if confidence intervals contain 0
# If the interval contains 0, it suggests that the predictor is not statistically significant
# This relates to the null hypothesis, where the null hypothesis is that the predictor's coefficient is equal to 0
# If the interval contains 0, we fail to reject the null hypothesis, indicating the predictor may not be relevant.

# Predict on the test data using the logistic regression model
predictions <- predict(model, newdata = testData, type = "response")

# Convert probabilities to class predictions using a 50% cutoff
predicted_classes <- ifelse(predictions > 0.5, "1", "0")

predicted_classes <- factor(predicted_classes, levels = c(0, 1))

# Create a confusion matrix
confusion_matrix <- confusionMatrix(predicted_classes, testData$Sex)

# Print the confusion matrix
print(confusion_matrix)

# Calculate accuracy
accuracy <- confusion_matrix$overall["Accuracy"]
```

```r
cat("Accuracy:", accuracy, "\n")

# ROC Curve
roc_curve <- roc(testData$Sex, predictions)
plot(roc_curve)

# Correlation plot
correlation_matrix <- cor(trainData[, -1])  # Exclude the "Sex" column
corrplot(correlation_matrix, method = "circle")
```



High correlations between predictors indicate multicollinearity. It can be problematic for logistic regression models because it can make it challenging to isolate the individual effects of predictors on the response variable. Correlation plots can help you identify which predictors are strongly correlated with the response variable. Variables with strong correlations with the response are likely to be more informative and may have a greater impact on classifier performance.
Accuracy of the logistic regression model is 0.566 while the ROC curve's accuracy is 0.497.

2.2 Problem 2

Load the mushroom sample dataset from the UCI Machine Learning Repository (agaricus-lepiota.data) into R using a dataframe (Note: There are missing values with a ? character, you will have to explain your handling of these). Create a Naive Bayes classifier using the e1071 package, using the sample function to split the data between 80% for training and 20% for testing. With the target class of interest being edible mushrooms, calculate the accuracy of the classifier both in-training and in-test. Use the table function to create a con- fusion matrix of predicted vs. actual classes - how many false positives did the model produce?

```r
# Load the required packages
library(e1071)

# Load the mushroom dataset from the UCI Machine Learning Repository
# Specify the column names as per dataset documentation.
column_names <- c("class", "cap_shape", "cap_surface", "cap_color", "bruises", "
    odor", "gill_attachment", "gill_spacing", "gill_size", "gill_color", "stalk_
    shape", "stalk_root", "stalk_surface_above_ring", "stalk_surface_below_ring"
    , "stalk_color_above_ring", "stalk_color_below_ring", "veil_type", "veil_
    color", "ring_number", "ring_type", "spore_print_color", "population", "
    habitat")
```

```r

# Load the dataset, treating "?" as missing values
mushroom_data <- read.csv("agaricus-lepiota.data", header = FALSE, col.names =
    column_names, na.strings = "?")

# Handle missing values by removing rows with any missing values
mushroom_data <- na.omit(mushroom_data)

# Split the data into training (80%) and testing (20%) sets
set.seed(123)  # for reproducibility
train_indices <- sample(1:nrow(mushroom_data), 0.8 * nrow(mushroom_data))
train_data <- mushroom_data[train_indices, ]
test_data <- mushroom_data[-train_indices, ]

# Create a Naive Bayes classifier
nb_classifier <- naiveBayes(class ~ ., data = train_data)

# Predict the classes for training and testing sets
train_predictions <- predict(nb_classifier, train_data)
test_predictions <- predict(nb_classifier, test_data)

# train_predictions to char vector
train_predictions <- as.character(train_predictions)

# Calculate accuracy train set
train_accuracy <- sum(train_predictions == train_data$class) / nrow(train_data)


# test_predictions to char vector
test_predictions <- as.character(test_predictions)

# Calculate accuracy test set
test_accuracy <- sum(test_predictions == test_data$class) / nrow(test_data)

# Create a confusion matrix for the testing set
conf_matrix <- table(Predicted = test_predictions, Actual = test_data$class)

# Calculate the number of false positives
false_positives <- conf_matrix["e", "p"]

# Print the results
cat("Accuracy on the training set:", train_accuracy, "\n")
cat("Accuracy on the testing set:", test_accuracy, "\n")
cat("Number of false positives:", false_positives, "\n")
```

Accuracy on the training set: 0.9521595.
Accuracy on the testing set: 0.9583702.
Number of false positives: 45.

2.3 Problem 3

Load the Yacht Hydrodynamics sample dataset from the UCI Machine Learning Repository (yacht hydrodynamics.data) into R using a dataframe (Note: The feature labels need to be manually specified). Use the caret package to perform a 80/20 test-train split (via the createDataPartition function), and obtain a training fit for a linear model. (Hint: The model fit should use all available features with the residuary resistance as the target.). What are the training MSE/RMSE and R^2 results? Next, use the caret package to perform a bootstrap from the full sample dataset with N=1000 samples for fitting a linear model (via the trainControl method), resulting in a

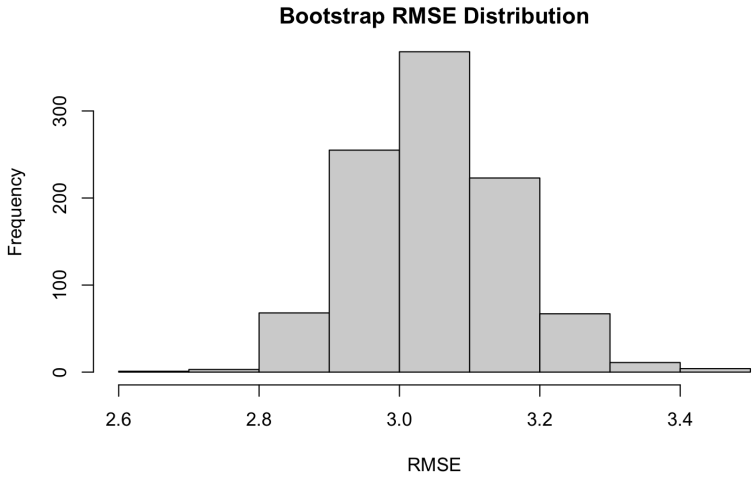
training MSE/RMSE and R^2 for each resample. Plot a histogram of the RMSE values, and provide a mean RMSE and R^2 for the fit. How do these values compare to the basic model? How does the performance on the test set for the original and boostrap model compare?

```r
# Load required libraries
library(caret)
library(boot)

# Load the dataset and specify feature labels
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht_hydrodynamics.data"
yacht_data <- read.table(url, header = FALSE, col.names = c("Longitudinal_position", "Prismatic_coefficient", "Length_displacement_ratio", "Beam-draught_ratio", "Length-beam_ratio", "Froude_number", "Residuary_resistance"))

# Perform an 80/20 train-test split
set.seed(123)  # For reproducibility
splitIndex <- createDataPartition(yacht_data$Residuary_resistance, p = 0.8, list = FALSE)
train_data <- yacht_data[splitIndex, ]
test_data <- yacht_data[-splitIndex, ]

# Fit a linear model using all features
lm_model <- lm(Residuary_resistance ~ ., data = train_data)

# Calculate MSE and R-squared for the training set
train_pred <- predict(lm_model, newdata = train_data)
train_mse <- mean((train_data$Residuary_resistance - train_pred)^2)
train_rmse <- sqrt(train_mse)
train_r_squared <- 1 - (sum((train_data$Residuary_resistance - train_pred)^2) / sum((train_data$Residuary_resistance - mean(train_data$Residuary_resistance))^2))

# Print results
cat("Training MSE:", train_mse, "\n")
cat("Training RMSE:", train_rmse, "\n")
cat("Training R-squared:", train_r_squared, "\n")
```

Training MSE: 79.91971.
Training RMSE: 8.939783.
Training R-squared: 0.6606717.

```r
# Set up bootstrap resampling
ctrl <- trainControl(method = "boot", number = 1000)

# Fit a linear model using bootstrap resampling
boot_results <- train(Residuary_resistance ~ ., data = train_data, method = "lm", trControl = ctrl)

# Extract RMSE and R-squared for each resample
boot_rmse <- sqrt(boot_results$resample$RMSE)
boot_r_squared <- boot_results$resample$Rsquared

# Calculate mean RMSE and R-squared
mean_boot_rmse <- mean(boot_rmse)
mean_boot_r_squared <- mean(boot_r_squared)

# Plot histogram of RMSE values
hist(boot_rmse, main = "Bootstrap RMSE Distribution", xlab = "RMSE")

# Print mean RMSE and R-squared for bootstrap model
cat("Mean Bootstrap RMSE:", mean_boot_rmse, "\n")
```

```r
cat("Mean Bootstrap R-squared:", mean_boot_r_squared, "\n")
```



Mean Bootstrap RMSE: 3.051137.
Mean Bootstrap R-squared: 0.6412813.
The Bootrap RMSE is lower than the basic model, the R-squared is a also a lower number in the Bootstrap model.

2.4 Problem 4

Load the German Credit Data sample dataset from the UCI Machine Learning Repository (german.data-numeric) into R using a dataframe (Note: The final column is the class variable coded as 1 or 2). Use the caret package to perform a 80/20 test-train split (via the createDataPartition function), and obtain a training fit for a logistic model via the glm package. (Hint: You may select a subset of the predictors based on exploratory analysis, or use all predictors for simplicity.). What are the training Precision/Recall and F_1 results? Next, use the trainControl and train functions to perform a k=10 fold cross-validation fit of the same model, and obtain cross-validated training Precision/Recall and F_1 values. How do these values compare to the original fit? How does the performance on the test set for the original and cross-validated model compare?

```r
library(caret)
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data-numeric"
data <- read.table(url, header = FALSE)
colnames(data)[ncol(data)] <- "Class"
set.seed(123)  # For reproducibility
splitIndex <- createDataPartition(data$Class, p = 0.8, list = FALSE)
trainData <- data[splitIndex, ]
testData <- data[-splitIndex, ]

# Convert the "Class" variable to a factor with levels 1 and 2 in both train and test data
trainData$Class <- factor(trainData$Class, levels = c(1, 2))
testData$Class <- factor(testData$Class, levels = c(1, 2))

# Fit the logistic model using glm
model <- glm(Class ~ ., data = trainData, family = binomial)

# Make predictions on the training data
```

```r
train_preds <- predict(model, newdata = trainData, type = "response")
train_preds <- ifelse(train_preds > 0.5, 2, 1)  # Convert probabilities to class labels

train_preds <- factor(train_preds, levels = c(1, 2))

# Create a confusion matrix
conf_matrix <- confusionMatrix(data = train_preds, reference = trainData$Class)

# Extract Precision, Recall, and F1-score from the confusion matrix
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
f1 <- conf_matrix$byClass["F1"]

# Print the results
cat("Training Precision:", precision, "\n")
cat("Training Recall:", recall, "\n")
cat("Training F1 Score:", f1, "\n")
```

Training Precision: 0.8084772.
Training Recall: 0.9035088.
Training F1 Score: 0.8533554

```r
# Convert the "Class" variable to a factor with valid R variable names
trainData$Class <- as.factor(trainData$Class)
testData$Class <- as.factor(testData$Class)

# Rename the levels to "Class_1" and "Class_2" (valid R variable names)
levels(trainData$Class) <- c("Class_1", "Class_2")
levels(testData$Class) <- c("Class_1", "Class_2")

# Define the training control with 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 10, classProbs = TRUE,
    summaryFunction = twoClassSummary)

# Fit the logistic model using cross-validation
cv_model <- train(Class ~ ., data = trainData, method = "glm", trControl = ctrl,
    family = binomial)

# Extract Precision, Recall, and F1-score from the cross-validated results
cv_precision <- cv_model$results$Precision
cv_recall <- cv_model$results$Recall
cv_f1 <- cv_model$results$F1
```