API 222 Problem Set 3

Machine Learning and Big Data Analytics: Spring 2024

Oscar Boochever

This problem set is worth 30 points in total. To get full credit, submit your code along with a write-up of your answers. This should either be done in R Markdown or Jupyter Notebook, submitted in one knitted PDF.

Conceptual Questions (11 pts)

1. Dimension Reduction (4 pts)

(a) What is the reason to scale (standardize) your variables when performing dimension reduction? (1 pt)

The reason to scale/standardize variables when performing dimension reduction is to ensure that all variables have the same scale, allowing them to be compared on an equivalent scale and contribute equally to the analysis.

(b) You want to run a Principal Components Regression (PCR). Let's denote two of your features X1 and X2. The correlation between X1 and the outcome variable Y is 0 and the correlation between X2 and Y is 0.9. Which of the two variables will receive more weight in the the first factor loading of PCR regression? Briefly explain. (1 pt)

In PCR, the first factor loading corresponds to the linear combination of variables that explains the most variance in the data. Since the correlation between X2 and the outcome variable Y is 0.9, X2 will receive more weight in the first factor loading compared to X1, which has no correlation with Y. Variables with higher correlations with the outcome variable tend to contribute more to the principal components.

(c) You want to run a Partial Least Squares (PLS) regression. Let's denote two of your features X1 and X2. The correlation between X1 and the outcome variable Y is 0 and the correlation between X2 and Y is 0.9. Which of the two variables will receive more weight in the the first factor loading of PLS regression? Briefly explain. (1 pt)

In Partial Least Squares (PLS) regression, variables are weighted based on their covariance with both the predictors and the outcome variable. Since the correlation between X2 and Y is 0.9, X2 will receive more weight in the first factor loading compared to X1, which has no correlation with Y.

(d) Is PCR a feature selection method? Why or why not? (1 pt)

No, PCR is not a feature selection method. It's a dimension reduction technique that creates new variables (principal components) as linear combinations of the original features. These components capture the most variance in the data, but they may not necessarily correspond directly to the original features (like the Arizona-NY crime example). PCR does not select specific features but rather transforms the original feature space into a smaller set of variables.

2. Shrinkage (3 pts)

(a) What is the reason to scale (standardize) your variables when performing shrinkage methods? (1 pt)

Scaling variables in shrinkage methods ensures balance by preventing those with larger scales or magnitudes from dominating the regularization process. This helps in avoiding biased coefficient estimates and ensures each variable contributes equally to the penalty term.

(b) What are the trade-offs between using LASSO versus Ordinary Least Squares (OLS) in terms of bias and variance? (1 pt)

LASSO tends to introduce more bias but lower variance compared to OLS. This means that LASSO may sacrifice some accuracy in prediction (higher bias) but can provide better generalization to new data (lower variance) by shrinking coefficients toward zero.

(c) What is the main advantage of LASSO over Ridge? (1 pt)

The main advantage of LASSO is its ability to perform feature selection by setting some coefficients exactly to zero. Effectively reducing the number of features with LASSO makes it particularly useful when dealing with high-dimensional datasets with many irrelevant features, also increasing interpretability. In contrast, Ridge regression tends to shrink coefficients towards zero without necessarily making them exactly zero, thus retaining all features in the model.

3. Tree-based Methods (4 pts)

(a) In 2-3 sentences, explain the idea of tree pruning and its importance in terms of the bias-variance trade-off. (1 pt)

Tree pruning involves removing unnecessary parts of a decision tree to simplify its structure and improve interpretability. This process is essential for managing the bias-variance trade-off: by reducing the complexity of the tree, we decrease the model's variance (reducing risk of overfitting), while accepting a slightly higher bias. This balance ensures that the model generalizes well to new data while still capturing important patterns, enhancing both interpretability and performance.

(b) Under what conditions might we expect a tree-based classifier to outperform a linear classifier such as LDA?(1 pt)

A tree-based classifier may outperform LDA when the relationship between features and the outcomes is highly non-linear or when the decision boundary is complex.

(c) Briefly explain how Bagging can lead to both low variance and low bias compared to decision trees. (1 pt)

(c) Bagging reduces variance by averaging predictions from multiple trees trained on bootstrapped samples of the data. This ensemble approach also helps to mitigate bias by combining the predictions of multiple models, each trained on slightly different subsets of the data. As a result, Bagging tends to produce more stable and robust predictions with lower variance and bias compared to individual decision trees.

(d) Briefly explain how Random Forests results in lower variance compared to Bagging. (1 pt)

(d) Random Forests reduce variance compared to Bagging by introducing an additional layer of randomness during the tree-building process. In a Random Forest, each tree is trained on a random subset of features, which reduces the correlation between individual trees and promotes diversity in the ensemble. This diversity further reduces the overall variance of the model, leading to more robust and stable predictions.

Data Questions (19 pts)

The DARWIN dataset contains handwriting data collected and is composed of 25 handwriting tasks. The protocol was specifically designed for the early detection of Alzheimer's disease (AD). The dataset includes data from 174 participants (89 AD patients and 85 healthy people).

The file "DARWIN.csv" contains the acquired data. The file consists of one row for each participant plus an additional header row. The first row is the header row, the next 89 rows collect patients data, whereas the remaining 84 rows collect information from healthy people.

The file consists of 452 columns. The first column shows participants' identifiers, whereas the last column shows the class to which each participant belongs. This value can be equal to 'P' (Patient) or 'H' (Healthy).

The remaining columns report the features extracted from a specific task. The tasks performed are 25, and for each task 18 features have been extracted. The column will be identified by the name of the features followed by a numeric identifier representing the task the feature is extracted. E.g., the column with the header "total_time8" collects the values for the "total time" feature extracted from task #8.

```r
# Load the data
handwriting <- read.csv("data/DARWIN.csv")
```

1. Data preparation (3pts)

(a) Remove the columns that contain the participant's identifier and convert `class` to a factor type. (1 pt)

```r
# Remove the columns that contain the participant's identifier
handwriting <- handwriting[, -1]

# Convert 'class' to a factor type
handwriting$class <- as.factor(handwriting$class)
```

(b) Using the random seed 222, split the data (70/30) into a training set and a test set. (1 pt)

```r
set.seed(222)
train_indices <- sample(seq(nrow(handwriting)), round(0.7 * nrow(handwriting)))
train_data <- handwriting[train_indices, ]
test_data <- handwriting[-train_indices, ]
```

(c) How many observations are in the training data set? How many observations are in the test data set? (1 pt)

```r
# Number of observations in the training data set
train_obs <- nrow(train_data)

# Number of observations in the test data set
test_obs <- nrow(test_data)

train_obs
```

```
## [1] 122
```

```
test_obs
```

```
## [1] 52
```

For the next set of questions, we will consider the prediction problem of classifying Alzheimer's disease patients from healthy people.

2. Decision tree (6 pts)

(a) Run a decision tree on your training data and use cross-validation to choose the optimal size. For your cross-validation use $k = 10$ and `prune.misclass` as the `FUN` argument. Hint: if you are not sure how to do this read the help file for `?cv.tree`. (1 pt)

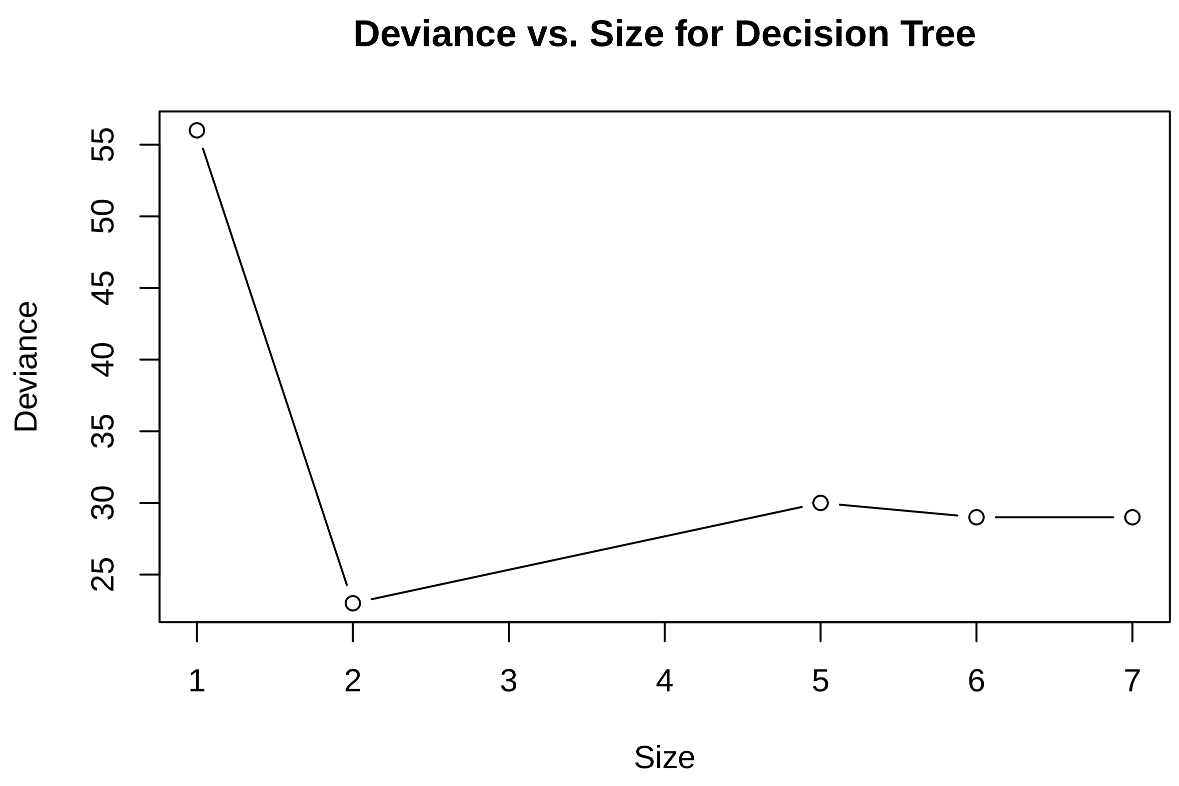
```
# Run a decision tree
library(tree)

# Fit a decision tree model
tree_model <- tree(class ~ ., data = train_data)

# Perform cross-validation to choose the optimal size
cv_tree <- cv.tree(tree_model, FUN = prune.misclass, K = 10)
```

(b) Show a plot of the deviance compared to the size. (2 pts)

```
# Plot deviance against size
plot(cv_tree$size, cv_tree$dev, type = "b",
     xlab = "Size", ylab = "Deviance",
     main = "Deviance vs. Size for Decision Tree")
```



(c) What is the optimal size? (1 pts)

```
# Find the optimal size
optimal_size <- cv_tree$size[which.min(cv_tree$dev)]
optimal_size
```

```
## [1] 2
```

We also see this visually with a minimum at 2.

(d) What is the misclassification error rate on the test set? Show this both for the original tree and for the pruned tree using the optimal size. (2 pts)

```
# Original tree misclassification error rate
tree_pred_orig <- predict(tree_model, newdata = test_data, type = "class")
error_rate_orig <- mean(tree_pred_orig != test_data$class)

# Pruned tree misclassification error rate
pruned_tree <- prune.misclass(tree_model, best = optimal_size)
tree_pred_pruned <- predict(pruned_tree, newdata = test_data, type = "class")
error_rate_pruned <- mean(tree_pred_pruned != test_data$class)

error_rate_orig
```

```
## [1] 0.25
```

```
error_rate_pruned
```

```
## [1] 0.1923077
```

The misclassification error rate for the original tree is 0.25, while for the pruned tree with the optimal size, it is 0.192.

3. Random forest (5 pts)

(a) Run a random forest with 5000 trees using the training data. (1 pt)

```
# Run a random forest
library(randomForest)

# Fit a random forest model
rf_model <- randomForest(class ~ ., data = train_data, ntree = 5000)
```

(b) What is your test misclassification error? (2 pts)

```
# Predict using the random forest model
rf_pred <- predict(rf_model, newdata = test_data)

# Calculate the misclassification error
error_rate_rf <- mean(rf_pred != test_data$class)
error_rate_rf
```

```
## [1] 0.1538462
```

The test misclassification error for the random forest model is 0.154.

(c) What are the three most important variables in the random forest? (2 pts)

```
# Extract variable importance
importance_rf <- importance(rf_model)

# Get the names of the top three most important variables
top_three_names <- rownames(importance_rf)[order(importance_rf, decreasing = TRUE)][1:3]
top_three_names
```

```
## [1] "total_time23" "air_time23"   "paper_time23"
```

The three most important variables in the random forest model are: "total_time23", "air_time23", and "paper_time23".

4. Boosting (4 pts)

(a) Recode the class variable so that it is 0 for healthy people and 1 for patients. (1 pt)

```r
# Recode the class variable
train_data$class <- ifelse(train_data$class == "P", 1, 0)
test_data$class <- ifelse(test_data$class == "P", 1, 0)
```

(b) Run a boosting model with 5000 trees and interaction depth of 4 using the training data. Hint: make sure you specify in your model that the distribution is "bernoulli". (1 pt)

```r
# Run a boosting model
library(gbm)
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```r
# Fit a boosting model
boost_model <- gbm(class ~ ., data = train_data, distribution = "bernoulli",
                    n.trees = 5000, interaction.depth = 4)
```

(c) What is your test misclassification error? Hint: You will need to determine a threshold for which you decide a predicted value to be 1 (ALZ) or 0 (Healthy). Use 0 as this threshold. (2 pts)

```r
# Predict using the boosting model
boost_pred <- predict(boost_model, newdata = test_data, n.trees = 5000, type = "response")

#Set Threshold
threshold <- 0

# Convert predicted probabilities to predicted classes using threshold of 0
boost_pred_class <- ifelse(boost_pred > threshold, 1, 0)

# Calculate misclassification error
error_rate_boost <- mean(boost_pred_class != test_data$class)
error_rate_boost
```

```
## [1] 0.5576923
```

The test misclassification error for the boosting model is 0.558.

5. Which model would you choose to predict Alzheimer's disease? Why? (1 pt)

I would choose the random forest model due to its superior performance with the lowest test misclassification error rate of 0.154. However, if interpretability is a priority, especially for discussing with patients, the pruned decision tree would be a valuable option despite its slightly higher error rate of 0.192.

6. Fun extra question (0 pt)

(a) Try running the threshold analysis for the boosting model to determine a better threshold than 0 to minimize misclassification error? (0 pts)

```r
# Threshold analysis for boosting model
thresholds <- seq(0, 1, by = 0.01)
error_rates <- numeric(length(thresholds))

for (i in seq_along(thresholds)) {
  boost_pred_class <- ifelse(boost_pred > thresholds[i], 1, 0)
  error_rates[i] <- mean(boost_pred_class != test_data$class)
}

# Find the threshold with the lowest misclassification error
best_threshold <- thresholds[which.min(error_rates)]
best_threshold
```

```
## [1] 0.01
```

```r
boost_pred_class <- ifelse(boost_pred > best_threshold, 1, 0)
error_rate_best_threshold <- mean(boost_pred_class != test_data$class)
error_rate_best_threshold
```

```
## [1] 0.1923077
```