CS 699 Assignment 2

Katherine Rein

```
# Import libraries
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.3.3
```

```
library(ROSE)
```

```
## Warning: package 'ROSE' was built under R version 4.3.3
```

```
## Loaded ROSE 0.0-4
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(tidyverse)
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.4     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x tidyr::pack()   masks Matrix::pack()
```

```
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
```

Problem 1

Find the distance between P4 and P5 and the distance between P4 and P9. Is P4 closer to P5 or P9? The attributes are nominal, not ordinal. Distances should be between 0 and 1.

```
# Read in data
data = read.csv('Problem1Data.csv')

# Create a list of columns to compare
nominal_cols <- c('job', 'marital', 'education', 'default', 'housing', 'loan', 'contact')

# Create a function to find the distance between nominal attributes
# Scaled Hamming distance = mismatched / total
# Scale: 0 = identical, 1 = all different
hamming <- function(id_a, id_b) {
  i <- match(id_a, data$ID)
  j <- match(id_b, data$ID)
  mean (data[i, nominal_cols] != data[j, nominal_cols])
}

dist_P4_P5 <- hamming("P4", "P5")
dist_P4_99 <- hamming("P4", "P9")
closer_car <- ifelse(dist_P4_P5 < dist_P4_99, "P5", "P9")
```

The closest observation to P4 between P5 and P9 is P9. This is because the hamming distance between P4 and P9 is 0.286 which is less than the distance between P4 and P5 -- 0.571.

Problem 2

(1). Calculate the distance between O1 and O2 using the Manhattan distance. (2). Calculate the distance between O1 and O2 using the Euclidean distance.

```
# Load Data
data = read.csv('Problem2Data.csv')
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## incomplete final line found by readTableHeader on 'Problem2Data.csv'
```

```
# Set Object Column as index
rownames(data) <- data[[1]]
data <- data[, -1]

# Calculate distances
euclidean_O1_O2 <- as.numeric(dist(data)[1])
manhattan_O1_O2 <- as.numeric(dist(data, method = "manhattan")[1])
```

The euvlidean distance is 23.17 and the manhattan distance is 41.

Problem 3

(1). Generate training and holdout partitions on the data set, holding out 1/3 of the data. (2). Fit a k-Nearest Neighbor model. Be sure to center and scale the data. Select an optimal value of k. Use the optimal model to make predictions on the holdout data. Generate a confusion matrix and compute the accuracy. (3). Does this seem to be a good model? Discuss why or why not.

```
# Load data
acc_data = read.csv('accidents1000.csv')

# Ensure the class column is categorical
acc_data$MAX_SEV <- factor(acc_data$MAX_SEV)

# Split 2/3 for train and 1/3 for test
set.seed(42)
split <- initial_split(acc_data, prop = 2/3, strata = MAX_SEV)
train <- training(split)
test <- testing(split)

# Use cross validation to find best k
# Make sure to scale and center data
ctrl <- trainControl(method = "cv", # 10-fold CV
                     number = 10,
                     classProbs = FALSE, summaryFunction = defaultSummary)

knn_mod <- train(MAX_SEV ~ ., data = train, method = "knn",
                 trControl = ctrl, preProcess = c("center", "scale"), # mandated by lecture
                 tuneLength = 30) # search 30 odd k's

best_k <- knn_mod$bestTune$k

# Predict on test data
pred <- predict(knn_mod, newdata = test, type = "raw")

# Create confusion matrix
cm <- confusionMatrix(data = pred, reference = test$MAX_SEV, positive = "fatal")

# Print values
list(best_k = best_k, confusion_table = cm$table, accuracy = cm$overall["Accuracy"])
```

```
## $best_k
## [1] 41
##
## $confusion_table
##            Reference
## Prediction  fatal no-injury non-fatal
##   fatal         0         0         0
##   no-injury     0        66        80
##   non-fatal     2        98        88
##
## $accuracy
##  Accuracy
## 0.4610778
```

The best k value is 41 and the accuracy is 0.461. 0.461 seems a bit low for this

to be a good model. As we know, accuracy isn't a great indicator of a model but that feels too low.

Problem 4

(1). Remove the observations with MAX_SEV = no-injury (2). Generate training and holdout partitions on the remaining data. Use 1/3 of the data in the holdout. (3). Fit a logistic regression model. Use the model to make predictions on the holdout data. Generate a confusion matrix and compute the accuracy and the F-score. (6). Apply the method (over-sampling or under-sampling) that is more likely to be helpful to the training data you already created in step (2). (7). Fit a logistic regression model to the class-balanced data set you just created in step (6). Use the model to make predictions on the original holdout data you created in step (2). Generate a confusion matrix and compute the accuracy and the F-score. (9.) Calculate variable importance for the model you fit in step (7). What are the top 3 most important variables?

```
# Load data
# *** USING 1030 DATA TO MAKE SURE THE SCORING VALUES WORK ***
acc_data2 = read.csv('accidents1030.csv')

# Ensure the class column is categorical
acc_data2$MAX_SEV <- factor(acc_data2$MAX_SEV)

# Remove no injury observations
acc_data2 <- subset(acc_data2, MAX_SEV != "no-injury")

# Train test split
set.seed(42)
split <- initial_split(acc_data2, prop = 2/3, strata = MAX_SEV)
train <- training(split)
test <- testing(split)
table(train$MAX_SEV)
```

```
##
##     fatal no-injury non-fatal
##        25         0       333
```

```
# Logistic model
logit_mod <- glm(MAX_SEV ~ ., data = train, family = 'binomial')

# Predict on test data
prob <- predict(logit_mod, newdata = test, type = "response")
pred <- factor(ifelse(prob >= 0.5, "fatal", "non-fatal"), levels = levels(test$MAX_SEV))

# Generate confusion matrix
cm <- confusionMatrix(data = pred, reference = test$MAX_SEV, positive = "fatal")

# Compute accuracy
acc <- cm$overall["Accuracy"]

# Compute F-score
calc_measures <- function(cm_tbl) {
  tp <- cm_tbl["fatal", "fatal"]
  fp <- cm_tbl["fatal", "non-fatal"]
  fn <- cm_tbl["non-fatal", "fatal"]
  precision <- tp / (tp + fp)
  recall <- tp / (tp + fn)
```

```
  f1 <- 2 * precision * recall / (precision + recall)
  c(precision = precision, recall = recall, F1 = f1)
}

scores <- calc_measures(cm$table)

# Print scores
list(accuracy = acc,
     precision = scores["precision"],
     recall = scores["recall"],
     F1 = scores["F1"])
```

```
## $accuracy
##   Accuracy
## 0.07821229
##
## $precision
##  precision
## 0.07303371
##
## $recall
## recall
##      1
##
## $F1
##        F1
## 0.1361257
```

```
# Oversample
train_over <- ovun.sample(MAX_SEV ~ ., data = train,
                          method = "over", seed = 33, p = 0.5)$data
table(train_over$MAX_SEV)
```

```
##
## non-fatal     fatal
##       333       338
```

```
logit_mod <- glm(MAX_SEV ~ ., data = train_over, family = 'binomial')
prob <- predict(logit_mod, newdata = test, type = "response")
pred <- factor(ifelse(prob >= 0.5, "fatal", "non-fatal"), levels = levels(test$MAX_SEV))
cm <- confusionMatrix(data = pred, reference = test$MAX_SEV, positive = "fatal")
acc <- cm$overall["Accuracy"]
scores <- calc_measures(cm$table)

# Print scores
list(accuracy = acc,
     precision = scores["precision"],
     recall = scores["recall"],
     F1 = scores["F1"])
```

```
## $accuracy
##  Accuracy
## 0.7150838
##
## $precision
```

```
##  precision
## 0.08695652
##
## $recall
##    recall
## 0.3076923
##
## $F1
##        F1
## 0.1355932
```

```
# Look at coeffs
summary(logit_mod)
```

```
##
## Call:
## glm(formula = MAX_SEV ~ ., family = "binomial", data = train_over)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.861927   0.532268  -3.498 0.000469 ***
## RushHour          -1.337225   0.240169  -5.568 2.58e-08 ***
## WRK_ZONE         -13.344322 466.938323  -0.029 0.977201
## WKDY              -1.900726   0.243065  -7.820 5.29e-15 ***
## INT_HWY           -0.060174   0.347131  -0.173 0.862378
## LGTCON_day        -1.566268   0.233931  -6.695 2.15e-11 ***
## LEVEL             -0.388817   0.261303  -1.488 0.136753
## SPD_LIM            0.073242   0.009104   8.045 8.63e-16 ***
## SUR_COND_dry      -1.137963   0.253129  -4.496 6.94e-06 ***
## TRAF_WAY_two_way   3.168811   0.337073   9.401  < 2e-16 ***
## WEATHER_adverse   -1.020278   0.333966  -3.055 0.002250 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 930.17  on 670  degrees of freedom
## Residual deviance: 648.39  on 660  degrees of freedom
## AIC: 670.39
##
## Number of Fisher Scoring iterations: 14
```

(4). Does this seem to be a good model? Discuss why or why not.

```
No this model is relatively terrible. The best f1 score is 1 and we see an f1
score of 0.136. This is most likely because our positive occurances are very rare.
```

(5). This model has class imbalance. Think about what you've learned about over- sampling and under-sampling. One of these techniques is less likely to work when applied to this data set. Which one, and why?

```
The worse sampling technique is undersampling in this scenario. This is because
undersampling would decrease the size to only a few observations if we weight
the classes equally. When there are very few positive cases then it is a bad
idea to undersample.
```

(8.) Compare this model to the one you fit in step (3). Which, if any, seems to perform better? Discuss.

The second model is significantly better. The accuracy and precision are both significantly better with values of 0.0782 to 0.7151 for accuracy and 0.0730 to 0.0870 for precision. The interesting thing is that the f1 score decreased from 0.1361 to 0.1356 so perhaps it's not as infinetly better as the accuracy and precision indicate.

(9.) Calculate variable importance for the model you fit in step (7). What are the top 3 most important variables?

The most important features are the ones with the largest (negative or positive) coeffs. This is because if the model doesn't think the feature is important it will essentially multiply it by 0 to get rid of it. The biggest 3 coeffs are for WRK_ZONE, TRAF_WAY_two_way, and WKDY.

Problem 5

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit a multiple linear regression model. Use the model to make predictions on the holdout data. Compute the MAE and the RMSE. (4). Apply a regularization method of your choosing, such as LASSO or ridge regression. Use the regularized model to make predictions on the holdout data. Compute the MAE and the RMSE.

```
# Load Data
pow_data = read.csv('powdermetallurgy.csv')

# Train test split
set.seed(42)
split <- initial_split(pow_data, prop = 2/3)
train <- training(split)
test <- testing(split)

# Linear model and predictions
fit <- lm(Shrinkage ~ ., data = train)
summary(fit)
```

```
##
## Call:
## lm(formula = Shrinkage ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.26068 -0.15986  0.02331  0.16961  0.60795
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -1.732e+01  4.500e-01 -38.500  < 2e-16 ***
## Formation.MethodExtrusion  -1.337e-01  1.379e-02  -9.698  < 2e-16 ***
## Formation.MethodIsostatic  -5.610e-02  1.014e-02  -5.534 3.32e-08 ***
## Compaction.MethodWarm      -2.072e+00  4.129e-02 -50.180  < 2e-16 ***
## Compaction.Pressure         3.682e-02  4.617e-04  79.734  < 2e-16 ***
## Sintering.Time              3.664e-02  1.813e-03  20.209  < 2e-16 ***
## Sintering.Temp.C            1.138e-02  4.120e-04  27.622  < 2e-16 ***
## Sintering.MethodVacuum     -5.327e-02  1.484e-02  -3.588 0.000337 ***
## Powder.SourceN              7.986e-02  5.732e-02   1.393 0.163644
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2472 on 4159 degrees of freedom
## Multiple R-squared:  0.842,  Adjusted R-squared:  0.8417
## F-statistic:  2770 on 8 and 4159 DF,  p-value: < 2.2e-16
```

```
pred <- predict(fit, new_data = test)

# MAE and RMSE
actual = test$Shrinkage
mae = mean(abs(actual - pred))
```

```
## Warning in actual - pred: longer object length is not a multiple of shorter
## object length
```

```
rmse = sqrt(mean((actual - pred)^2))
```

```
## Warning in actual - pred: longer object length is not a multiple of shorter
## object length
```

```
# Apply regularization
X <- model.matrix(Shrinkage ~ ., data = train)[, -1]
y <- train$Shrinkage

# Perform 10-fold cross-validation to find best lambda
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)

# Setting alpha = 0 implements ridge regression
ridge_cv <- cv.glmnet(X, y, alpha = 0,
                      lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)

# Best cross-validated lambda
lambda_cv <- ridge_cv$lambda.min

# Fit final model with best lambda
best_ridge <- glmnet(X, y, alpha = 0, lambda = lambda_cv,
                     standardize = TRUE)
y_hat_cv <- predict(best_ridge, X)

# MAE and RMSE
mae = mean(abs(y - y_hat_cv))
rmse = sqrt(mean((y - y_hat_cv)^2))
```

(3). Does this seem to be a good model? Discuss why or why not.

```
To me I would call this a good model. The R2 value is 0.842 which means that the
model explains 84.2% of the variance. Additionally all features are stastically
significant (with p values less than 0.05) except for one. The MAE is 0.6795 and
the RMSE is 0.8411.
```

(5). Compare this model to the one you worked with in steps (2) and (3). Which, if any, seems to perform better? Discuss.

```
The better model was the one with the regularization in step 4. The second model
has an MAE of 0.1978 and a RMSE of 0.2471. This is significantly lower than the
first model of a MAE of 0.6795 and RMSE of 0.8411
```

Problem 6

This problem is about the logistic regression we discussed in the class. Consider a dataset that has two independent variables A1 and A2 and a class attribute, which takes on either yes or no. Suppose you ran a logistic regression algorithm on the dataset and obtained the following coefficients for class yes: Coefficient of A1 = 0.045 Coefficient of A2 = 0.003 Intercept = -3.485 Classify the following two unseen objects using the above model: O1: A1 = 47, A2 = 213 O2: A1 = 65, A2 = 276 Assume that the classification threshold is 0.5.

```
The formula: p = 1/(1+e^-(b0+b1*A1+b2*A2))
b0 = -3.485
b1 = 0.045
b2 = 0.003

p_o1 = 0.3250
p_o2 = 0.5666

With a threshold of 0.5, O1 would be classified as no and O2 as yes.
```