CSI 2300: Introduction to Data Science

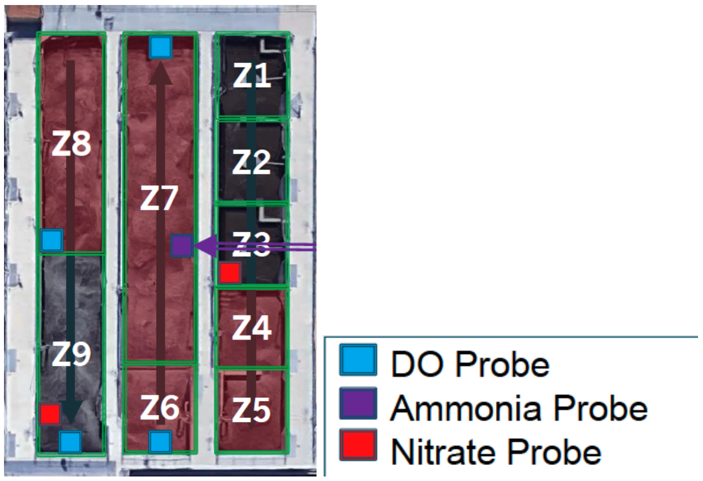
Exam 03

Exam Guidelines:

- Show your code for every question.
- Submit your solutions as a .pdf knitted from a Markdown document.
- You may use your notes and any online resources.
- You may not confer with other students or people.
- If you have a question, Dr. Wilkerson will be on hand to answer them. Email directly to rachel_wilkerson@baylor.edu (not through Canvas).

The data that you will use for this exam come from the Boulder Water Resource and Recovery Facility, and they were also used for the Lecture 6 in-class exercises. Here is a short review of the data.

First is a picture of the facility where the data are collected. It shows three aeration basins together, and the next plot shows a diagram of the flow of water through one aeration basin. The red highlighted basins are "aerated," meaning that oxygen is being pumped by blowers into the sludge, and the other basins are not aerated. Two goals in this exam are to (1) predict ammonia in Zone 7 and (2) classify observations as occurring on weekends versus weekdays using the measured numeric variables.



Run this code first. It loads the data, adds two columns that are a function of the hour of the day, and removes one redundant column. If you do not have the `mowater` library, then load the `boulder_ammonia.rda` dataset, and run the lines after the comment `Data Wrangling`.

```r
#suppressMessages(library(mowateR))
#data(boulder_ammonia)
suppressMessages(library(lubridate))
load("boulder_ammonia.rda")
```

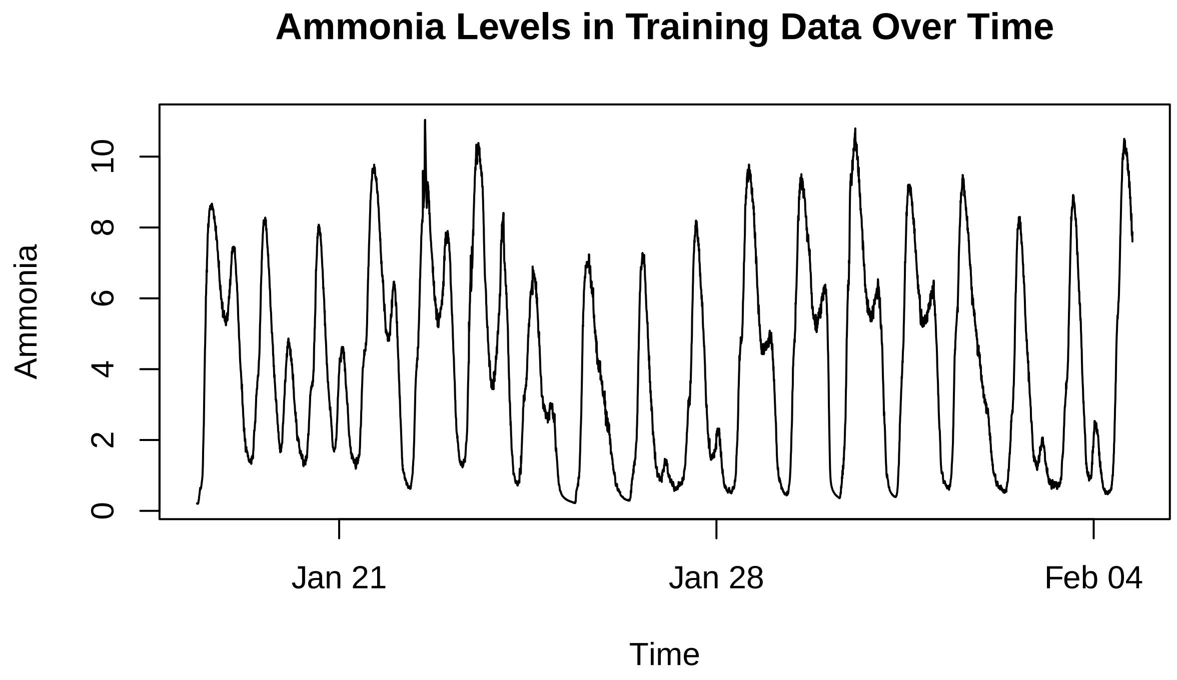
```r
# Data Wrangling
obs_hour <- hour(boulder_ammonia$Date.Time)
obs_minute <- minute(boulder_ammonia$Date.Time)
boulder_ammonia$cosine_hour <- cos((obs_hour + obs_minute/60)*(360/24)*pi/180)
boulder_ammonia$sine_hour <- sin((obs_hour + obs_minute/60)*(360/24)*pi/180)

boulder_ammonia <- boulder_ammonia[ , -15]
```

1. Split the data such that observations 5,000 to 9,999 are in a dataframe representing the training set, and observations 15,000 to 19,999 are in a dataframe representing the testing set. Call these `train` and `test`, respectively. (4 points)

```r
train <- boulder_ammonia[5000:9999, ]
test <- boulder_ammonia[15000:19999, ]
```

- (a) Plot ammonia in the training dataset over time. What pattern(s) do you notice? (4 points)

```r
plot(train$Date.Time, train$AB3.Z7.Ammonia.mg.N.L, type = 'l',
     xlab = "Time", ylab = "Ammonia",
     main = "Ammonia Levels in Training Data Over Time")
```



- (b) If the goal is to build a model for ammonia, what is the purpose of splitting the data into training and testing sets? (4 points)

The purpose of splitting data into training and testing sets when building a model for ammonia is important for evaluating the model's ability to generalize to new data. It can detect overfitting and aid in tuning of the model. The train-test split is used to ensure that the developed model is not just memorizing the data it was built on but can make accurate predictions on new data.

2. One possibility is to use the cosine and sine variables to model ammonia. Fit a model using only these two variables on the training dataset. It will be referred to as the `trig` model for the remainder of the exam. (4 points)

```r
trig_model <- lm(AB3.Z7.Ammonia.mg.N.L ~ cosine_hour + sine_hour, data = train)
summary(trig_model)
#
# Call:
# lm(formula = AB3.Z7.Ammonia.mg.N.L ~ cosine_hour + sine_hour,
#     data = train)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -4.2060 -1.2511 -0.3339  1.2713  6.1358
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  4.05955    0.02762  146.99   <2e-16 ***
# cosine_hour -1.00396    0.03894  -25.79   <2e-16 ***
# sine_hour   -2.74227    0.03917  -70.01   <2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.952 on 4997 degrees of freedom
# Multiple R-squared:  0.5275,  Adjusted R-squared:  0.5273
# F-statistic:  2789 on 2 and 4997 DF,  p-value: < 2.2e-16
```

`- (a) Is the model still linear even though using the cosine and sine predictors produce`

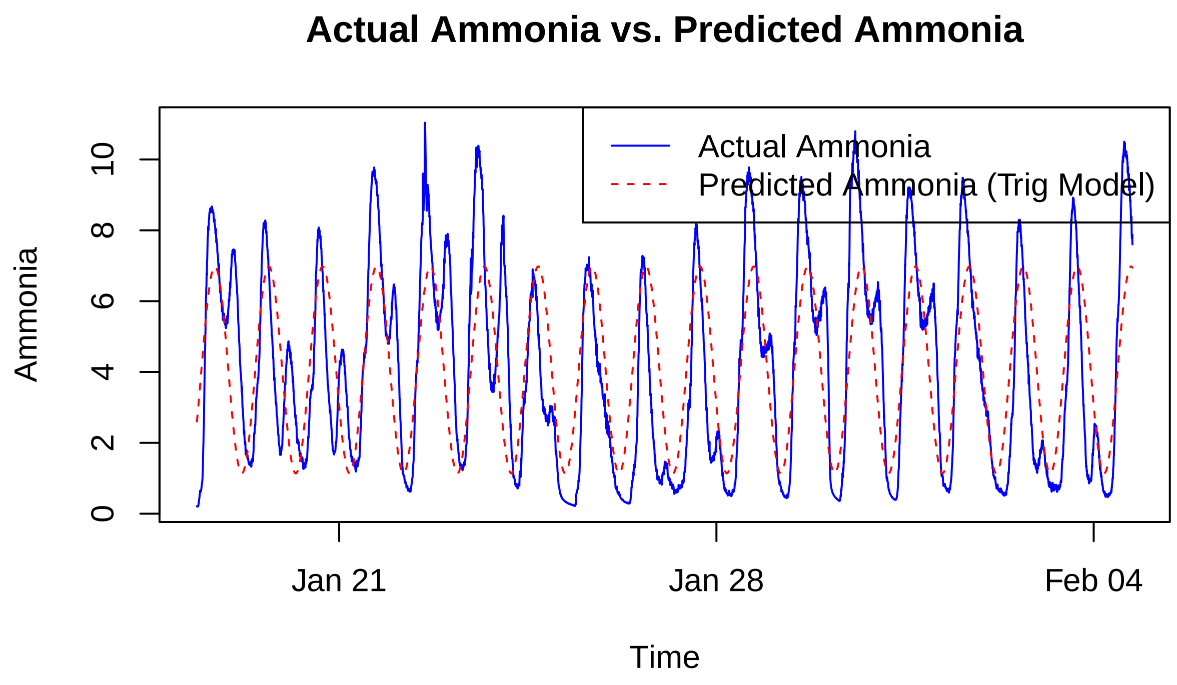
Yes, the model is still considered a linear model. Linear means that the model is linear in its parameters. The resulting fit of ammonia over the original time variable will appear non-linear, but the statistical model itself is linear in its parameters.

`- (b) What is the $R^2$ of this model for the training data? (4 points)`

Multiple R-squared: 0.5275, Adjusted R-squared: 0.5273

```
- (c) Create a plot of the ammonia values over time from the training dataset with the p
```

```r
train_predictions_trig <- predict(trig_model, newdata = train)

plot(train$Date.Time, train$AB3.Z7.Ammonia.mg.N.L, type = 'l',
     xlab = "Time", ylab = "Ammonia",
     main = "Actual Ammonia vs. Predicted Ammonia",
     col = "blue", ylim = range(c(train$AB3.Z7.Ammonia.mg.N.L, train_predictions_trig),

lines(train$Date.Time, train_predictions_trig, col = "red", lty = 2)

legend("topright", legend = c("Actual Ammonia", "Predicted Ammonia (Trig Model)"),
       col = c("blue", "red"), lty = c(1, 2))
```



3. Build a model using the training dataset and backward selection with the BIC criteria. (4 points)

```r
full_mod <- lm(AB3.Z7.Ammonia.mg.N.L ~ . , data = train[ , -1])

n_train <- nrow(train)
```

```
backward_model_bic <- step(full_mod, direction = "backward", k = log(n_train))
# Start:  AIC=2378.62
# AB3.Z7.Ammonia.mg.N.L ~ AB3.Z6.DO.mg.L + AB3.Z7.DO.mg.L + AB3.Z8.DO.mg.L +
#     AB3.Z9.DO.mg.L + AB3.Z6.Header.Flow.SCFM + AB3.Z7.Header.Flow.SCFM +
#     AB3.Z8.Header.Flow.SCFM + AB3.Zone.6.Valve.Position + AB3.Zone.7.Valve.Position
#     AB3.Zone.8.Valve.Position + AB3.Z3.Nitrate.mg.N.L + AB3.Z3.NO2.mg.N.L +
#     AB3.Z9.NO2.mg.N.L + cosine_hour + sine_hour
#
#                             Df Sum of Sq     RSS    AIC
# - AB3.Z9.DO.mg.L             1      1.98  7831.5 2371.4
# <none>                                    7829.6 2378.6
# - AB3.Zone.7.Valve.Position  1     43.94  7873.5 2398.1
# - AB3.Zone.6.Valve.Position  1     48.92  7878.5 2401.2
# - AB3.Zone.8.Valve.Position  1     53.75  7883.3 2404.3
# - sine_hour                  1     64.52  7894.1 2411.1
# - AB3.Z8.DO.mg.L             1     72.15  7901.7 2416.0
# - AB3.Z6.Header.Flow.SCFM    1    132.97  7962.5 2454.3
# - AB3.Z7.DO.mg.L             1    135.58  7965.2 2455.9
# - AB3.Z9.NO2.mg.N.L          1    224.17  8053.7 2511.2
# - AB3.Z7.Header.Flow.SCFM    1    339.16  8168.7 2582.1
# - AB3.Z3.NO2.mg.N.L          1    516.12  8345.7 2689.3
# - AB3.Z6.DO.mg.L             1    593.03  8422.6 2735.2
# - AB3.Z3.Nitrate.mg.N.L      1    656.70  8486.3 2772.8
# - cosine_hour                1   2207.45 10037.0 3612.0
# - AB3.Z8.Header.Flow.SCFM    1   2226.47 10056.0 3621.4
#
# Step:  AIC=2371.37
# AB3.Z7.Ammonia.mg.N.L ~ AB3.Z6.DO.mg.L + AB3.Z7.DO.mg.L + AB3.Z8.DO.mg.L +
#     AB3.Z6.Header.Flow.SCFM + AB3.Z7.Header.Flow.SCFM + AB3.Z8.Header.Flow.SCFM +
#     AB3.Zone.6.Valve.Position + AB3.Zone.7.Valve.Position + AB3.Zone.8.Valve.Positio
#     AB3.Z3.Nitrate.mg.N.L + AB3.Z3.NO2.mg.N.L + AB3.Z9.NO2.mg.N.L +
#     cosine_hour + sine_hour
#
#                             Df Sum of Sq     RSS    AIC
# <none>                                    7831.5 2371.4
# - AB3.Zone.7.Valve.Position  1     44.10  7875.6 2390.9
# - AB3.Zone.6.Valve.Position  1     48.99  7880.5 2394.0
# - AB3.Zone.8.Valve.Position  1     53.74  7885.3 2397.0
# - sine_hour                  1     65.44  7897.0 2404.5
# - AB3.Z8.DO.mg.L             1     73.01  7904.6 2409.3
# - AB3.Z6.Header.Flow.SCFM    1    132.66  7964.2 2446.8
# - AB3.Z7.DO.mg.L             1    137.06  7968.6 2449.6
# - AB3.Z9.NO2.mg.N.L          1    224.30  8055.9 2504.0
# - AB3.Z7.Header.Flow.SCFM    1    339.59  8171.1 2575.1
# - AB3.Z3.NO2.mg.N.L          1    517.19  8348.7 2682.6
# - AB3.Z6.DO.mg.L             1    593.39  8424.9 2728.0
# - AB3.Z3.Nitrate.mg.N.L      1    656.04  8487.6 2765.1
# - cosine_hour                1   2209.92 10041.5 3605.7
# - AB3.Z8.Header.Flow.SCFM    1   2227.79 10059.3 3614.6

summary(backward_model_bic)
#
# Call:
# lm(formula = AB3.Z7.Ammonia.mg.N.L ~ AB3.Z6.DO.mg.L + AB3.Z7.DO.mg.L +
#     AB3.Z8.DO.mg.L + AB3.Z6.Header.Flow.SCFM + AB3.Z7.Header.Flow.SCFM +
#     AB3.Z8.Header.Flow.SCFM + AB3.Zone.6.Valve.Position + AB3.Zone.7.Valve.Position
#     AB3.Zone.8.Valve.Position + AB3.Z3.Nitrate.mg.N.L + AB3.Z3.NO2.mg.N.L +
#     AB3.Z9.NO2.mg.N.L + cosine_hour + sine_hour, data = train[,
#     -1])
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -5.5053 -0.7766 -0.0509  0.7367  7.3945
#
# Coefficients:
#                             Estimate Std. Error t value Pr(>|t|)
# (Intercept)               -2.3038923  0.3553905  -6.483 9.88e-11 ***
# AB3.Z6.DO.mg.L            -2.0359956  0.1047602 -19.435  < 2e-16 ***
# AB3.Z7.DO.mg.L             1.0268078  0.1099321   9.340  < 2e-16 ***
# AB3.Z8.DO.mg.L            -0.9215204  0.1351749  -6.817 1.04e-11 ***
# AB3.Z6.Header.Flow.SCFM    0.0020209  0.0002199   9.189  < 2e-16 ***
# AB3.Z7.Header.Flow.SCFM   -0.0069473  0.0004725 -14.702  < 2e-16 ***
# AB3.Z8.Header.Flow.SCFM    0.0192562  0.0005114  37.657  < 2e-16 ***
# AB3.Zone.6.Valve.Position  0.0275034  0.0049251   5.584 2.47e-08 ***
# AB3.Zone.7.Valve.Position  0.0294585  0.0055603   5.298 1.22e-07 ***
# AB3.Zone.8.Valve.Position -0.0421166  0.0072009  -5.849 5.27e-09 ***
# AB3.Z3.Nitrate.mg.N.L     -0.5391960  0.0263859 -20.435  < 2e-16 ***
# AB3.Z3.NO2.mg.N.L          0.4070598  0.0224350  18.144  < 2e-16 ***
# AB3.Z9.NO2.mg.N.L          2.0709463  0.1733177  11.949  < 2e-16 ***
# cosine_hour               -2.1714545  0.0578966 -37.506  < 2e-16 ***
# sine_hour                 -0.2895152  0.0448597  -6.454 1.19e-10 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 1.253 on 4985 degrees of freedom
# Multiple R-squared:  0.8057,  Adjusted R-squared:  0.8052
# F-statistic:  1477 on 14 and 4985 DF,  p-value: < 2.2e-16
```

- (a) To fit the full model needed for the `step` function, the code needed is: `full_mod <- lm(AB3.Z7.Ammonia.mg.N.L ~ . , data = train[,-1])` What is the purpose of the `-1` in the `data=train[,-1]`? (4 points)

In this dataset, the first column is `Date.Time`. This column is removed because it is not directly suitable as a numeric predictor in the linear model, and its temporal aspects are intended to be captured by the `cosine_hour` and `sine_hour` variables.

- (b) Which variables, if any, are removed from the model? (4 points)

The variable removed from the model during backward selection is AB3.Z9.DO.mg.L.

- (c) Does it appear that including the additional variables improves upon the `trig` model from the prior question? Explain your answer. (6 points)

Yes, the `backward_model_bic` shows a substantial improvement over the `trig_model`. The `backward_model_bic`, which includes 14 predictors, has a Multiple R-squared of 0.806 and an Adjusted R-squared of 0.805.This significant increase in both R-squared values indicates that the additional variables selected by the backward selection process with BIC account for a much larger proportion of the variability in ammonia levels.

4. Now, build a model for ammonia using the training dataset and lasso. (6 points)

```r
suppressMessages(library(glmnet))

x_train_lasso <- model.matrix(AB3.Z7.Ammonia.mg.N.L ~ . -1, data = train[, !names(train)
y_train_lasso <- train$AB3.Z7.Ammonia.mg.N.L

set.seed(123)
cv_lasso_model <- cv.glmnet(x_train_lasso, y_train_lasso, alpha = 1)

optimal_lambda <- cv_lasso_model$lambda.min
cat("Optimal lambda (lambda.min):", optimal_lambda, "\n")
# Optimal lambda (lambda.min): 0.2416674

lasso_coeffs <- coef(cv_lasso_model, s = optimal_lambda)
print(lasso_coeffs)
# 16 x 1 sparse Matrix of class "dgCMatrix"
#                                     s1
# (Intercept)                 0.20394221
# AB3.Z6.DO.mg.L              .
# AB3.Z7.DO.mg.L              .
# AB3.Z8.DO.mg.L              .
# AB3.Z9.DO.mg.L               .
# AB3.Z6.Header.Flow.SCFM      .
# AB3.Z7.Header.Flow.SCFM      .
# AB3.Z8.Header.Flow.SCFM      0.01125650
# AB3.Zone.6.Valve.Position    .
# AB3.Zone.7.Valve.Position    .
# AB3.Zone.8.Valve.Position    0.01595412
# AB3.Z3.Nitrate.mg.N.L       -0.05237439
# AB3.Z3.NO2.mg.N.L            .
# AB3.Z9.NO2.mg.N.L            0.70918007
# cosine_hour                 -1.17202125
# sine_hour                   -0.90218887

final_lasso_model <- glmnet(x_train_lasso, y_train_lasso, alpha = 1, lambda = optimal_la
```

- (a) Which variables are chosen for the model? (6 points)

```
chosen_lasso_vars <- lasso_coeffs[which(lasso_coeffs != 0), , drop = FALSE]
cat("Variables chosen by Lasso (non-zero coefficients):\n")
# Variables chosen by Lasso (non-zero coefficients):
print(rownames(chosen_lasso_vars)[-1])
# [1] "AB3.Z8.Header.Flow.SCFM"   "AB3.Zone.8.Valve.Position"
# [3] "AB3.Z3.Nitrate.mg.N.L"     "AB3.Z9.NO2.mg.N.L"
# [5] "cosine_hour"               "sine_hour"
cat("\nCoefficients:\n")
#
# Coefficients:
print(chosen_lasso_vars)
# 7 x 1 sparse Matrix of class "dgCMatrix"
#                                      s1
# (Intercept)                  0.20394221
# AB3.Z8.Header.Flow.SCFM      0.01125650
# AB3.Zone.8.Valve.Position    0.01595412
# AB3.Z3.Nitrate.mg.N.L       -0.05237439
# AB3.Z9.NO2.mg.N.L            0.70918007
# cosine_hour                 -1.17202125
# sine_hour                   -0.90218887
```

- (b) What is the $R^2$ for this model on the training data? You have to calculate it "

```
predictions_lasso_train <- predict(final_lasso_model, newx = x_train_lasso, s = optimal_

SSE_lasso_train <- sum((y_train_lasso - predictions_lasso_train)^2)
```

```
SST_lasso_train <- sum((y_train_lasso - mean(y_train_lasso))^2)

R_squared_lasso_train <- 1 - (SSE_lasso_train / SST_lasso_train)

cat("SSE for Lasso (training):", SSE_lasso_train, "\n")
# SSE for Lasso (training): 10833.92
cat("SST for Lasso (training):", SST_lasso_train, "\n")
# SST for Lasso (training): 40308.15
cat("R-squared for Lasso (training):", R_squared_lasso_train, "\n")
# R-squared for Lasso (training): 0.7312226
```

5. Now, use the 3 models that you have fit (trig, backward, and lasso) to make predictions for the **testing** set. Fill in the table below: (10 points)

Model	R^2	RMSE	MAE
Trig	0.5893	1.6346	1.2673
Back	0.1369	2.3696	1.8631
Lasso	0.5487	1.7136	1.1704

6. If you had to recommend one model to the Boulder Water Resource and Recovery Facility, which one would it be? Explain your answer. (5 points)

I would recommend the Trig model as the trig mode had the highest R^2 indicating it explains the largest proprtion of variance. The trig model had the lowest RMSE suggesting its predictions were cloest to the actual values. The model had a lower MAE. The Trig model is the simplest using only two predictors. The backward model overfitted making the model test poorly on test data. While the Lasso model showed the best MAE and offers automatic variable selection, the Trig model demonstrated superior performance in terms of R^2 and RMSE on the test data, and it is the most parsimonious.

BONUS: There is a glaring problem with the backward model's predictions. What is it? (5 points)

The most glaring problem with the backward model's predictions is severe overfitting to the training data. The backward model had a Multiple R-squared of approximately 0.806 on the training data, suggesting it explained a large portion of the variance in that specific dataset. On the unseen test data, the same model had an R-squared of only 0.1369. This is a massive drop in explanatory power.

7. The goal in this question is to classify whether it is a weekend day (Fri/Sat/Sun) or a weekday (Mon/Tues/Wed) using `new_train` and `new_test`, which will be the training and testing sets, respectively, for this problem. Include the `{r}` after the triple tick

marks below, and run the chunk of code. It sets up these two new frames with a new column called `class` that indicates whether the observation occurs on "FSS" or "MTW".

```r
week_index_train <- 1913:3640
new_train <- train[week_index_train, -1]
new_train_col <- c(rep("MTW", 864), rep("FSS", 864))
new_train$class <- as.factor(new_train_col)

week_index_test <- 1993:3720
new_test <- test[week_index_test, -1]
new_test_col <- c(rep("MTW", 864), rep("FSS", 864))
new_test$class <- as.factor(new_test_col)
```

- (a) Normalize the scale of the variables (except for `class`). See the code for Question 3, Lecture 24's exercises. Why is it important to remove the influence of scale in a clustering or classification problem? (5 points)

```r
predictor_cols <- setdiff(names(new_train), "class")

train_means <- sapply(new_train[, predictor_cols], mean, na.rm = TRUE)
train_sds <- sapply(new_train[, predictor_cols], sd, na.rm = TRUE)

new_train_scaled <- new_train
for(col_name in predictor_cols){
  new_train_scaled[, col_name] <- (new_train[, col_name] - train_means[col_name]) / trai
}

new_test_scaled <- new_test
for(col_name in predictor_cols){
  new_test_scaled[, col_name] <- (new_test[, col_name] - train_means[col_name]) / train_
}
```

It is important to remove the influence of scale in many clustering and classification algorithms, particularly those that are distance-based or rely on variance. In the context of k-NN, which is used later in this question, distances between data points are fundamental.

- (b) Fit k-nearest neighbors with the **training** data (and $k = 1$) to make predictions on the **test** data. Report the confusion matrix and accuracy. (5 points)

```r
suppressMessages(library(class))

knn_train_predictors <- new_train_scaled[, predictor_cols]
```

```r
knn_train_labels <- new_train_scaled$class

knn_test_predictors <- new_test_scaled[, predictor_cols]
knn_test_labels_actual <- new_test_scaled$class

set.seed(123)
knn_predictions_k1 <- knn(train = knn_train_predictors,
                          test = knn_test_predictors,
                          cl = knn_train_labels,
                          k = 1)

confusion_matrix_k1 <- table(Predicted = knn_predictions_k1, Actual = knn_test_labels_ac
cat("Confusion Matrix (k=1):\n")
# Confusion Matrix (k=1):
print(confusion_matrix_k1)
#          Actual
# Predicted FSS MTW
#       FSS 704 472
#       MTW 160 392

accuracy_k1 <- sum(diag(confusion_matrix_k1)) / sum(confusion_matrix_k1)
cat("\nAccuracy (k=1):", sprintf("%.4f", accuracy_k1), "\n")
#
# Accuracy (k=1): 0.6343
```

The confusion matrix is:

```
         Actual
Predicted FSS MTW
      FSS 704 472
      MTW 160 392
```

The accuracy of this model is 0.6343.

- (c) Is this model able to distinguish between weekday (MTW) and weekend (FSS) days? (4 points)

The model shows some ability to distinguish between weekday and weekend days, but its performance is modest. An accuracy of 63.43% is better than random guessing, indicating some level of discriminative power. So while there's some ability there are many errors.

- (d) Which type of day has the greater proportion of errors–MTW or FSS? (5 points)

To determine which type of day has a greater proportion of errors, we need to calculate the error rate for each class.

For FSS: Proportion of errors for FSS = 160 / 864

For MTW: Proportion of errors for MTW = 472 / 864

Therefore, MTW days have a significantly greater proportion of errors.

```r
actual_fss_total <- confusion_matrix_k1["FSS", "FSS"] + confusion_matrix_k1["MTW", "FSS"
misclassified_fss <- confusion_matrix_k1["MTW", "FSS"]
error_prop_fss <- misclassified_fss / actual_fss_total

actual_mtw_total <- confusion_matrix_k1["FSS", "MTW"] + confusion_matrix_k1["MTW", "MTW"
misclassified_mtw <- confusion_matrix_k1["FSS", "MTW"]
error_prop_mtw <- misclassified_mtw / actual_mtw_total

cat(sprintf("Proportion of errors for FSS: %.4f (%.2f%%)\n", error_prop_fss, error_prop_
# Proportion of errors for FSS: 0.1852 (18.52%)
cat(sprintf("Proportion of errors for MTW: %.4f (%.2f%%)\n", error_prop_mtw, error_prop_
# Proportion of errors for MTW: 0.5463 (54.63%)
```