Lab 6

Brandon Leslie

```
library(tidymodels)
library(tidyverse)

Heart <- read_csv("data/Heart.csv")
Heart <- Heart %>% select(-...1)
```

logistic regression, data splitting, overfitting, tidymodels

Using the Heart data on Brightspace, we predict "all heart disease", AHD, based on selected other variables.

Split the data into training and test sets. Do this "manually". (Note that you can also use tidymodels rsample package for many different types of data splitting.)

```
iris <- iris %>%
 mutate(setosa_condition = ifelse(Species == "setosa", 1, 0))

Heart$AHD <- as.factor(Heart$AHD)
Heart$Fbs <- as.factor(Heart$Fbs)
Heart$Sex <- as.factor(Heart$Sex)
iris$setosa_condition <- as.factor(iris$setosa_condition)


set.seed(88)

Heart_Splitting <- sample(1:nrow(Heart), size = 0.75 * nrow(Heart))


Heart_Test <- Heart[-Heart_Splitting, ]
Heart_Train <- Heart[Heart_Splitting, ]
```

Fit a logistic regression model on the training set, predict on both training and test sets and find the training and test set accuracies Do this manually, then use metric_set from the yardstick package in tidy-models

Training Set Logistic Reg. Model - AHD

```
Heart_Mod1 <- logistic_reg() %>% fit(AHD ~., data = Heart_Train, family = binomial)
  summary(Heart_Mod1)
```

```
             Length Class        Mode
lvl           2     -none-       character
spec          7     logistic_reg list
fit          31     glm          list
preproc       1     -none-       list
elapsed       1     -none-       list
censor_probs  0     -none-       list
```

AHD Training Set Pred./Accuracy

```
# Probability of each observation (person) having heart disease or not
Heart_Train_Pred1 <- predict(Heart_Mod1, new_data = Heart_Train, type = "prob")


# Creates a dataframe that combines AHD probability with AHD status
 Heart_Train_Results1 <- bind_cols(Heart_Train_Pred1 ,
  Heart_Train %>% select(AHD))

 #Creating a column that checks to see whether prediction was right or not.
Heart_Train_Results1 <- Heart_Train_Results1 %>%
  mutate(
   pred_class = factor(ifelse(.pred_Yes > 0.5, "Yes", "No"), levels = c("No", "Yes"))
   )

#Changes pred_class & AHD ot factor so it can interact with the rest f this code smoothly
Heart_Train_Results1$pred_class <- as.factor(Heart_Train_Results1$pred_class)

#Checks metrics of code based on AHD and pred class. Actually numeircal anayysis of predictio
Heart_Train_Results1 %>% sensitivity(truth = "AHD", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.897
```

```
#manually checks for accuracy
mean(Heart_Train_Results1$AHD == Heart_Train_Results1$pred_class, na.rm = TRUE)
```

```
[1] 0.8642534
```

Metrics - AHD Training Set

```
metrics1 <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Train_Results1 %>% metrics1(truth = "AHD", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.864
2 precision   binary         0.854
3 ppv         binary         0.854
4 sensitivity binary         0.897
5 recall      binary         0.897
6 f_meas      binary         0.875
```

AHD Test Set Pred./Accuracy

```
Heart_Test_Pred1 <- predict(Heart_Mod1, new_data = Heart_Test , type = "prob")


 Heart_Test_Results1 <- bind_cols(Heart_Test_Pred1 ,
  Heart_Test %>% select(AHD))

Heart_Test_Results1 <- Heart_Test_Results1 %>%
  mutate(
   pred_class = factor(ifelse(.pred_Yes > 0.5, "Yes", "No"), levels = c("No", "Yes"))
   )

Heart_Test_Results1$pred_class <- as.factor(Heart_Test_Results1$pred_class)
```

```
Heart_Test_Results1 %>% sensitivity(truth = "AHD", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.977
```

```
mean(Heart_Test_Results1$AHD == Heart_Test_Results1$pred_class, na.rm = TRUE)
```

```
[1] 0.9210526
```

Metrics - AHD Testing Set

```
metrics1 <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Test_Results1 %>% metrics1(truth = "AHD", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.921
2 precision   binary         0.894
3 ppv         binary         0.894
4 sensitivity binary         0.977
5 recall      binary         0.977
6 f_meas      binary         0.933
```

Experiment with different predictors and note if the Heart_Train_Results1 change. In particular, so some models result in much lower test set accuracy? This is called overfitting.

Important to keep in mind that levels for truth & estimate may be different when trying to create the "pred_class" column for prediction accuracies. Columns like **Sex** and **Fbs** contains **Dichotomous** data, while they cannot take on the term "true" or "false" because they are classifications, there are two distinct classifications that come into play in those columns. Whereas in AHD, they are both referring to the possession of one thing, rather then the difference of 2 distinct things under 1 category.

With that being said, this code was made so that it could be replicated across multiple different classification metrics with little change. The **mutate()** section should be the one that is manipulated.

Training Set Logistic Reg. Model - Sex

```
Heart_Mod2 <-  logistic_reg() %>% fit(Sex ~., data = Heart_Train, family = binomial)
  summary(Heart_Mod2)
```

```
             Length Class        Mode
lvl           2     -none-       character
spec          7     logistic_reg list
fit          31     glm          list
preproc       1     -none-       list
elapsed       1     -none-       list
censor_probs  0     -none-       list
```

Sex Training Set Pred./Accuracy

```
Heart_Train_Pred2 <- predict(Heart_Mod2, new_data = Heart_Train, type = "prob")


 Heart_Train_Results2 <- bind_cols(Heart_Train_Pred2 ,
  Heart_Train %>% select(Sex))

 #Take note on how .pred_Yes changed to .pred_1, this is indicative of the different classifi

Heart_Train_Results2 <- Heart_Train_Results2 %>%
  mutate(
   pred_class = factor(ifelse(.pred_1 > 0.5, "1", "0"), levels = c("0", "1"))
   )
Heart_Train_Results2$pred_class <- as.factor(Heart_Train_Results2$pred_class)

Heart_Train_Results2 %>% sensitivity(truth = "Sex", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.543
```

```
mean(Heart_Train_Results2$Sex == Heart_Train_Results2$pred_class, na.rm = TRUE)
```

```
[1] 0.7737557
```

Metrics - Sex Training Set

```
metrics <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Train_Results2 %>% metrics(truth = "Sex", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.774
2 precision   binary         0.679
3 ppv         binary         0.679
4 sensitivity binary         0.543
5 recall      binary         0.543
6 f_meas      binary         0.603
```

Sex Test Set Pred./Accuracy

```
Heart_Test_Pred2 <- predict(Heart_Mod2, new_data = Heart_Test , type = "prob")


 Heart_Test_Results2 <- bind_cols(Heart_Test_Pred2 ,
  Heart_Test %>% select(Sex))

Heart_Test_Results2 <- Heart_Test_Results2 %>%
  mutate(
   pred_class = factor(ifelse(.pred_1 > 0.5, "1", "0"), levels = c("0", "1"))
   )

Heart_Test_Results2$pred_class <- as.factor(Heart_Test_Results2$pred_class)

Heart_Test_Results2 %>% sensitivity(truth = "Sex", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.462
```

```
mean(Heart_Test_Results2$Sex == Heart_Test_Results2$pred_class, na.rm = TRUE)
```

```
[1] 0.7236842
```

Metrics - Sex Testing Set

```
metrics <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Test_Results2 %>% metrics(truth = "Sex", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.724
2 precision   binary         0.632
3 ppv         binary         0.632
4 sensitivity binary         0.462
5 recall      binary         0.462
6 f_meas      binary         0.533
```

Training Set Logistic Reg. Model - Fasting Blood Sugar (Fbs)

```
Heart_Mod3 <-logistic_reg() %>% fit(Fbs ~ . , data = Heart_Train, family = binomial )
summary(Heart_Mod3)
```

```
             Length Class        Mode
lvl           2     -none-       character
spec          7     logistic_reg list
fit          31     glm          list
preproc       1     -none-       list
elapsed       1     -none-       list
censor_probs  0     -none-       list
```

Fbs Training Set Pred./Accuracy

```
Heart_Train_Pred3 <- predict(Heart_Mod3, new_data = Heart_Train, type = "prob")


 Heart_Train_Results3 <- bind_cols(Heart_Train_Pred3 ,
  Heart_Train %>% select(Fbs))

Heart_Train_Results3 <- Heart_Train_Results3 %>%
  mutate(
```

```
   pred_class = factor(ifelse(.pred_1 > 0.5, "1", "0"), levels = c("0", "1"))
   )
Heart_Train_Results3$pred_class <- as.factor(Heart_Train_Results3$pred_class)

Heart_Train_Results3 %>% sensitivity(truth = "Fbs", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.979
```

```
mean(Heart_Train_Results3$Fbs == Heart_Train_Results3$pred_class, na.rm = TRUE)
```

```
[1] 0.8642534
```

Metrics - Fbs Training Set

```
metrics <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Train_Results3 %>% metrics(truth = "Fbs", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.864
2 precision   binary         0.877
3 ppv         binary         0.877
4 sensitivity binary         0.979
5 recall      binary         0.979
6 f_meas      binary         0.925
```

Fbs Test Set Pred./Accuracy

```
Heart_Test_Pred3 <- predict(Heart_Mod3, new_data = Heart_Test , type = "prob")


 Heart_Test_Results3 <- bind_cols(Heart_Test_Pred3 ,
  Heart_Test %>% select(Fbs))
```

```
Heart_Test_Results3 <- Heart_Test_Results3 %>%
  mutate(
   pred_class = factor(ifelse(.pred_1 > 0.5, "1", "0"), levels = c("0", "1"))
   )

Heart_Test_Results3$pred_class <- as.factor(Heart_Test_Results3$pred_class)

Heart_Test_Results3 %>% sensitivity(truth = "Fbs", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.969
```

```
mean(Heart_Test_Results3$Fbs == Heart_Test_Results3$pred_class, na.rm = TRUE)
```

```
[1] 0.8157895
```

Metrics - Fbs Testing Set

```
metrics <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
Heart_Test_Results3 %>% metrics(truth = "Fbs", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.816
2 precision   binary         0.838
3 ppv         binary         0.838
4 sensitivity binary         0.969
5 recall      binary         0.969
6 f_meas      binary         0.899
```

Accuracy is roughly similar to all three Predictors, if anything, training set accuracy is higher then predictors.

Repeat the above with the iris data set. There are three species, but we predict whether species is setosa or not, making this into a binary classification problem.

Data Splitting - iris

```
#Creates variable that is equal to one if species in row is equal to setosa, and returns 0 i


iris_Splitting <- sample(1:nrow(iris), size = 0.75 * nrow(iris))


iris_test <- iris[-iris_Splitting, ]
iris_train <- iris[iris_Splitting, ]
```

Training Set Model - iris

```
iris_mod <-logistic_reg() %>% fit(Species ~ . , data = iris_train, family = binomial )
summary(iris_mod)
```

```
             Length Class        Mode
lvl           3     -none-       character
spec          7     logistic_reg list
fit          30     glm          list
preproc       1     -none-       list
elapsed       1     -none-       list
censor_probs  0     -none-       list
```

Training Set Pred./Accuracy - iris

```
iris_train_pred <- predict(iris_mod, new_data = iris_train, type = "prob")

#note that for the minority, (virginica), there is no pred column. This would be an issue, h
 iris_train_results <- bind_cols(iris_train_pred ,
  iris_train %>% select(setosa_condition))

 iris_train_results <- iris_train_results %>%
  mutate(pred_class = factor(ifelse(.pred_setosa > 0.5, "1", "0"), levels = c("0", "1"))
 iris_train_results %>% sensitivity(truth = "setosa_condition", estimate = pred_class)
```

```
# A tibble: 1 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary             1
```

```
mean(iris_train_results$setosa_condition == iris_train_results$pred_class, na.rm = TRUE)
```

```
[1] 1
```

iris training set metrics

```
metrics <- metric_set(accuracy, precision,  ppv, sensitivity, recall, f_meas)
iris_train_results %>% metrics(truth = "setosa_condition", estimate =  pred_class)
```

```
# A tibble: 6 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary             1
2 precision   binary             1
3 ppv         binary             1
4 sensitivity binary             1
5 recall      binary             1
6 f_meas      binary             1
```

This data is weird because all the metrics add up to be one, meaning that there might not be enough data for a "true" regression model. Maybe this involves some type of deeper analysis, but for now, I would like to keep my sanity.

K nearest neighbors

K nearest neighbors classification simply predicts the probability as the proportion of nearest neighbors with the outcome. Since it is measuring distances, the predictors must be numerical. Binary categorical variables, such as 1, 0 for student or not, are okay.

Repeat the above examples but use k nearest neighbors to fit. Make sure to scale the data using recipes from tidymodels, and use only the numerical variables or binary coded 0, 1. Try k = 1, k = 3, k = 5, k = 10. You should set mode = "classification" inside the argument; for example nearest_neighbor(neighbors = 3, mode = "classification")

AHD - Heart Data

```
rec_knn <- recipe(AHD ~ ., data = Heart_Train) %>%
  step_normalize(all_numeric(), -all_outcomes())  # Normalize all numeric variables except th

knn_1 <- nearest_neighbor(neighbors = 1, mode = "classification")
knn_3 <- nearest_neighbor(neighbors = 3, mode = "classification")
knn_5 <- nearest_neighbor(neighbors = 5, mode = "classification")
knn_10 <- nearest_neighbor(neighbors = 10, mode = "classification")

workflow_knn_1 <- workflow() %>%
  add_recipe(rec_knn) %>%
  add_model(knn_1)

workflow_knn_3 <- workflow() %>%
  add_recipe(rec_knn) %>%
  add_model(knn_3)

workflow_knn_5 <- workflow() %>%
  add_recipe(rec_knn) %>%
  add_model(knn_5)

workflow_knn_10 <- workflow() %>%
  add_recipe(rec_knn) %>%
  add_model(knn_10)

knn_fit_1 <- fit(workflow_knn_1, data = Heart_Train)
knn_fit_3 <- fit(workflow_knn_3, data = Heart_Train)
knn_fit_5 <- fit(workflow_knn_5, data = Heart_Train)
knn_fit_10 <- fit(workflow_knn_10, data = Heart_Train)

knn_preds_1 <- predict(knn_fit_1, new_data = Heart_Test, type = "class")
knn_preds_3 <- predict(knn_fit_3, new_data = Heart_Test, type = "class")
knn_preds_5 <- predict(knn_fit_5, new_data = Heart_Test, type = "class")
knn_preds_10 <- predict(knn_fit_10, new_data = Heart_Test, type = "class")

metrics <- metric_set(accuracy, precision, sensitivity, f_meas)

knn_results_1 <- bind_cols(knn_preds_1, Heart_Test %>% select(AHD))
knn_results_3 <- bind_cols(knn_preds_3, Heart_Test %>% select(AHD))
knn_results_5 <- bind_cols(knn_preds_5, Heart_Test %>% select(AHD))
knn_results_10 <- bind_cols(knn_preds_10, Heart_Test %>% select(AHD))
```

```r
knn_results_1 %>%
  metrics(truth = AHD, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.776
2 precision   binary         0.825
3 sensitivity binary         0.767
4 f_meas      binary         0.795
```

```r
knn_results_3 %>%
  metrics(truth = AHD, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.776
2 precision   binary         0.825
3 sensitivity binary         0.767
4 f_meas      binary         0.795
```

```r
knn_results_5 %>%
  metrics(truth = AHD, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.816
2 precision   binary         0.854
3 sensitivity binary         0.814
4 f_meas      binary         0.833
```

```r
knn_results_10 %>%
  metrics(truth = AHD, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
```

```
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.855
2 precision   binary         0.864
3 sensitivity binary         0.884
4 f_meas      binary         0.874
```

Fbs - Heart Data

```
rec_knn_class <- recipe(Fbs ~ ., data = Heart_Train) %>%
  step_normalize(all_numeric(), -all_outcomes())

knn_class_1 <- nearest_neighbor(neighbors = 1, mode = "classification")
knn_class_3 <- nearest_neighbor(neighbors = 3, mode = "classification")
knn_class_5 <- nearest_neighbor(neighbors = 5, mode = "classification")
knn_class_10 <- nearest_neighbor(neighbors = 10, mode = "classification")

workflow_knn_class_1 <- workflow() %>%
  add_recipe(rec_knn_class) %>%
  add_model(knn_class_1)

workflow_knn_class_3 <- workflow() %>%
  add_recipe(rec_knn_class) %>%
  add_model(knn_class_3)

workflow_knn_class_5 <- workflow() %>%
  add_recipe(rec_knn_class) %>%
  add_model(knn_class_5)

workflow_knn_class_10 <- workflow() %>%
  add_recipe(rec_knn_class) %>%
  add_model(knn_class_10)

knn_class_fit_1 <- fit(workflow_knn_class_1, data = Heart_Train)
knn_class_fit_3 <- fit(workflow_knn_class_3, data = Heart_Train)
knn_class_fit_5 <- fit(workflow_knn_class_5, data = Heart_Train)
knn_class_fit_10 <- fit(workflow_knn_class_10, data = Heart_Train)

knn_class_preds_1 <- predict(knn_class_fit_1, new_data = Heart_Test, type = "class")
knn_class_preds_3 <- predict(knn_class_fit_3, new_data = Heart_Test, type = "class")
knn_class_preds_5 <- predict(knn_class_fit_5, new_data = Heart_Test, type = "class")
knn_class_preds_10 <- predict(knn_class_fit_10, new_data = Heart_Test, type = "class")
```

```r
metrics_class <- metric_set(accuracy, precision, sensitivity, f_meas)

knn_class_results_1 <- bind_cols(knn_class_preds_1, Heart_Test %>% select(Fbs))
knn_class_results_3 <- bind_cols(knn_class_preds_3, Heart_Test %>% select(Fbs))
knn_class_results_5 <- bind_cols(knn_class_preds_5, Heart_Test %>% select(Fbs))
knn_class_results_10 <- bind_cols(knn_class_preds_10, Heart_Test %>% select(Fbs))

knn_class_results_1 %>%
  metrics_class(truth = Fbs, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.763
2 precision   binary         0.838
3 sensitivity binary         0.891
4 f_meas      binary         0.864
```

```r
knn_class_results_3 %>%
  metrics_class(truth = Fbs, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.763
2 precision   binary         0.838
3 sensitivity binary         0.891
4 f_meas      binary         0.864
```

```r
knn_class_results_5 %>%
  metrics_class(truth = Fbs, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.816
2 precision   binary         0.847
3 sensitivity binary         0.953
4 f_meas      binary         0.897
```

```r
knn_class_results_10 %>%
  metrics_class(truth = Fbs, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 accuracy    binary         0.829
2 precision   binary         0.84
3 sensitivity binary         0.984
4 f_meas      binary         0.906
```

Can knn be used for regression; that is to predict a continuous outcome? Yes it can. Instead of the proportion of nearest neighbors, it simply takes the average; that is, the mean, of the nearest neighbors. You must set the mode = regression". Pick a simple data set like mtcars and perform knn regression.

```r
set.seed(88)
data_split <- initial_split(mtcars, prop = 0.7)
train_data <- training(data_split)
test_data <- testing(data_split)

rec_knn_reg <- recipe(mpg ~ ., data = train_data) %>%
  step_normalize(all_numeric(), -all_outcomes())

knn_reg <- nearest_neighbor(neighbors = 5, mode = "regression")

workflow_knn_reg <- workflow() %>%
  add_recipe(rec_knn_reg) %>%
  add_model(knn_reg)

knn_reg_fit <- fit(workflow_knn_reg, data = train_data)

knn_reg_preds <- predict(knn_reg_fit, new_data = test_data)

knn_reg_results <- bind_cols(knn_reg_preds, test_data %>% select(mpg))

metrics_reg <- metric_set(rmse, rsq)

knn_reg_results %>%
  metrics_reg(truth = mpg, estimate = .pred)
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       4.66
2 rsq     standard       0.721
```

Multiclass prediction

Can either of logistic regression or Knn be used for multiclass problems? In tidymodels, we use multinom_reg() for multinomial regression. We dont have to make any adjustments for nearest_neighbors()

```
library(caret)
library(nnet)

wine_data <- read.csv("wine.csv")
wine_data$Class.label <- as.factor(wine_data$Class.label)

set.seed(42)
train_index <- createDataPartition(wine_data$Class.label, p = 0.7, list = FALSE)
train_data <- wine_data[train_index, ]
test_data <- wine_data[-train_index, ]

preproc <- preProcess(train_data[, -1], method = c("center", "scale"))
train_scaled <- predict(preproc, train_data[, -1])
test_scaled <- predict(preproc, test_data[, -1])
train_scaled$Class.label <- train_data$Class.label
test_scaled$Class.label <- test_data$Class.label

set.seed(42)
knn_model <- train(Class.label ~ ., data = train_scaled, method = "knn",
                   tuneLength = 5,
                   trControl = trainControl(method = "cv", number = 5))

set.seed(42)
multinom_model <- multinom(Class.label ~ ., data = train_data)
```

```
# weights:  45 (28 variable)
initial  value 138.425148
iter  10 value 24.884058
iter  20 value 3.585849
```

```
iter  30 value 0.553449
iter  40 value 0.032580
iter  50 value 0.008028
iter  60 value 0.002621
iter  70 value 0.000923
iter  80 value 0.000379
final  value 0.000094
converged
```

```r
knn_preds <- predict(knn_model, newdata = test_scaled)
multinom_preds <- predict(multinom_model, newdata = test_data)

knn_conf_matrix <- confusionMatrix(knn_preds, test_data$Class.label)
multinom_conf_matrix <- confusionMatrix(multinom_preds, test_data$Class.label)

print("KNN Model Performance:")
```

```
[1] "KNN Model Performance:"
```

```r
print(knn_conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 17  1  0
         2  0 19  0
         3  0  1 14

Overall Statistics

               Accuracy : 0.9615
                 95% CI : (0.8679, 0.9953)
    No Information Rate : 0.4038
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9419

 Mcnemar's Test P-Value : NA

Statistics by Class:
```

```
                     Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9048   1.0000
Specificity            0.9714   1.0000   0.9737
Pos Pred Value         0.9444   1.0000   0.9333
Neg Pred Value         1.0000   0.9394   1.0000
Prevalence             0.3269   0.4038   0.2692
Detection Rate         0.3269   0.3654   0.2692
Detection Prevalence   0.3462   0.3654   0.2885
Balanced Accuracy      0.9857   0.9524   0.9868
```

```
print("Multinomial Logistic Regression Performance:")
```

```
[1] "Multinomial Logistic Regression Performance:"
```

```
print(multinom_conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 17  1  0
         2  0 20  0
         3  0  0 14

Overall Statistics

               Accuracy : 0.9808
                 95% CI : (0.8974, 0.9995)
    No Information Rate : 0.4038
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9708

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9524   1.0000
Specificity            0.9714   1.0000   1.0000
```

```
Pos Pred Value         0.9444   1.0000   1.0000
Neg Pred Value         1.0000   0.9687   1.0000
Prevalence             0.3269   0.4038   0.2692
Detection Rate         0.3269   0.3846   0.2692
Detection Prevalence   0.3462   0.3846   0.2692
Balanced Accuracy      0.9857   0.9762   1.0000
```