UMD Methods Workshop: Introduction to Machine Learning

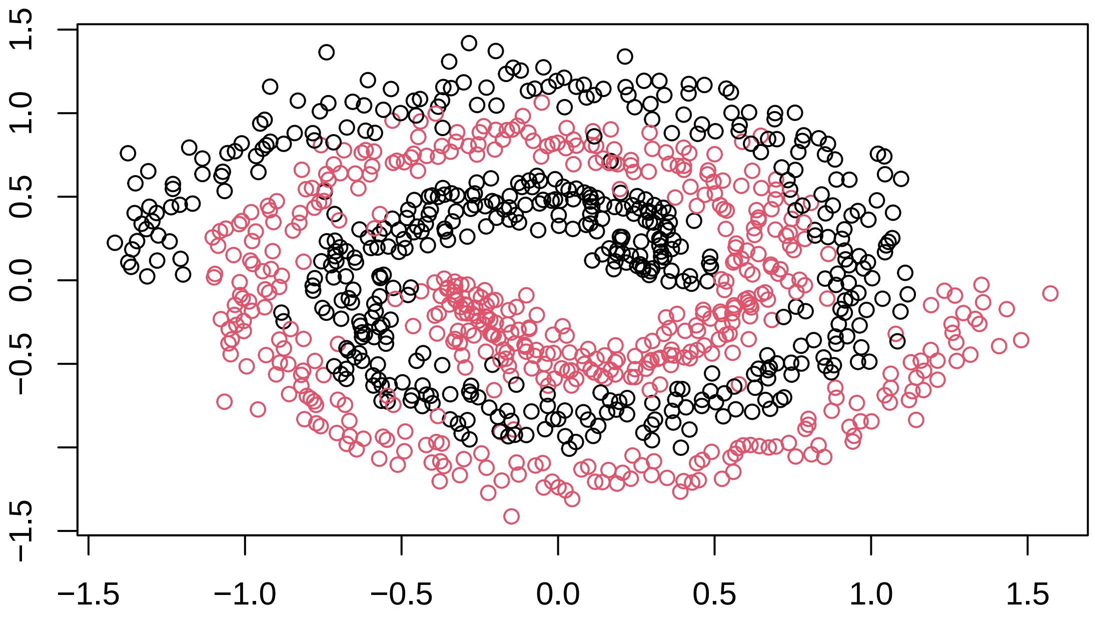
Wenqing Huangfu

11 October 2023

Classifying a dataset

Pick a dataset from the `mlbench` package. Experiment with classifying the data using KNN at different values of k. Use cross-validation to choose your best model.

```r
library(mlbench)
set.seed(777)
new_data <-mlbench.spirals(1000, 1.5, 0.1)
plot(new_data)
```



Splitting the data

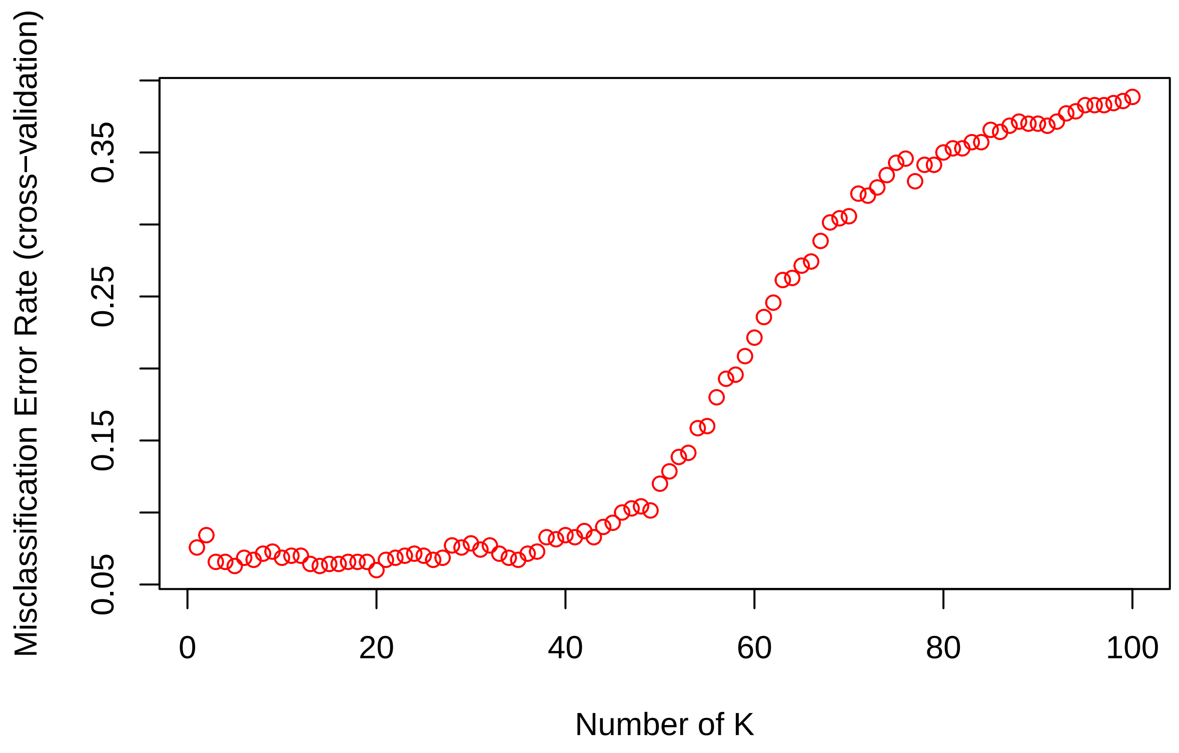
```r
new_data_x <- new_data$x
new_data_y <- new_data$classes

set.seed(123)
indices <- sample(1:nrow(new_data_x), nrow(new_data_x)*0.7)
train_x <- new_data_x[indices, ]
train_y <- new_data_y[indices]
test_x <- new_data_x[-indices, ]
test_y <- new_data_y[-indices]
```

Plot misclassification error rate at different values of k.

```r
# Cross-validation on training set
library(class)
knn.cv.error <- rep(0,100)
for (i in 1:100){
    knn.cv.results <- knn.cv(train_x, train_y, k = i)
    #This uses leave-one-out cross validation.
    #For each row of the training set train, the k nearest (in Euclidean distance)
    #other training set vectors are found, and the classification is decided by majority vote,
    #with ties broken at random. If there are ties for the kth nearest vector,
    #all candidates are included in the vote.
    knn.cv.error[i] <- mean(knn.cv.results!=train_y)
}

best_k <- which.min(knn.cv.error)
best_k
```

```
## [1] 20
```

```r
# Plotting misclassification rate
plot(1:100, knn.cv.error, type='b', col='red', xlab="Number of K",
     ylab="Misclassification Error Rate (cross-validation)")
```

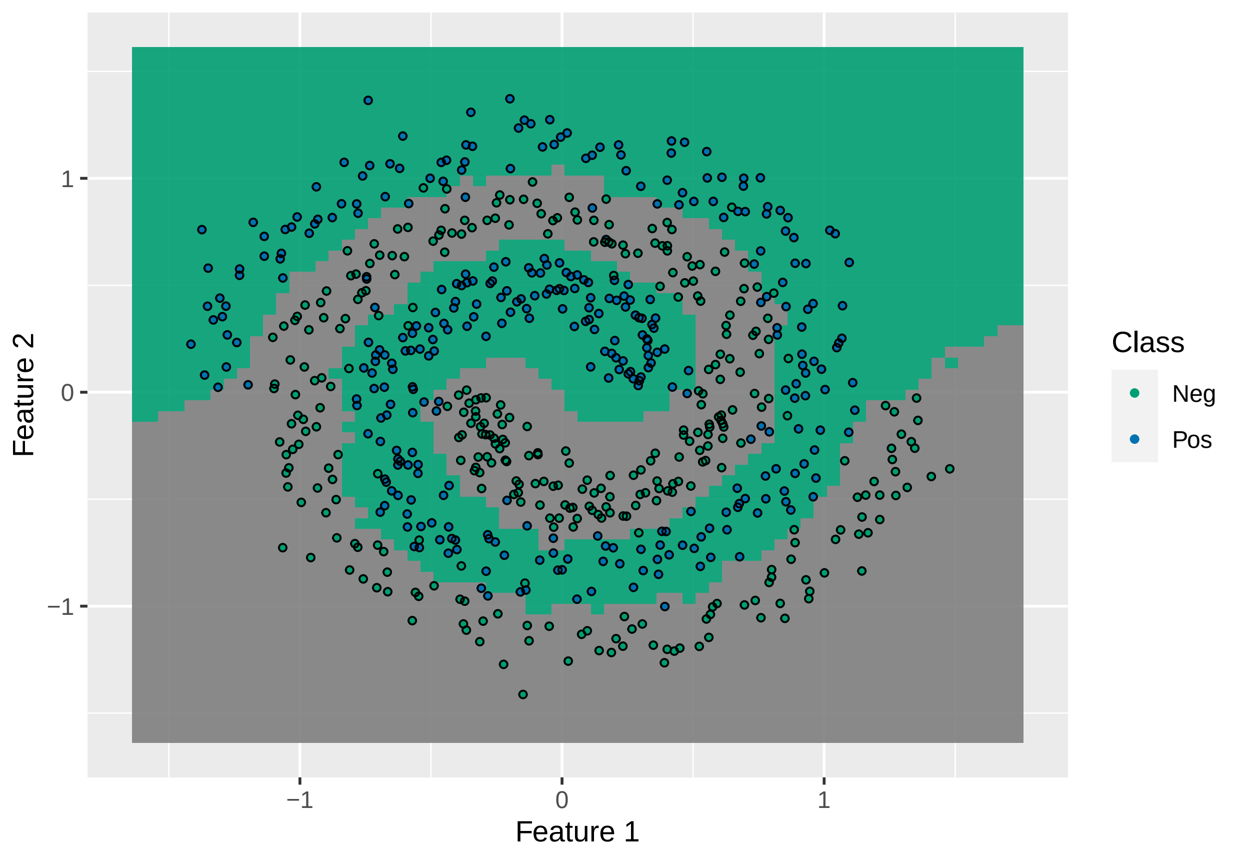


Plot the decision boundary

Plot the decision boundary for your classifier using the function at the top code block, `plot_decision_boundary()`. Make sure you load this function into memory before trying to use it.

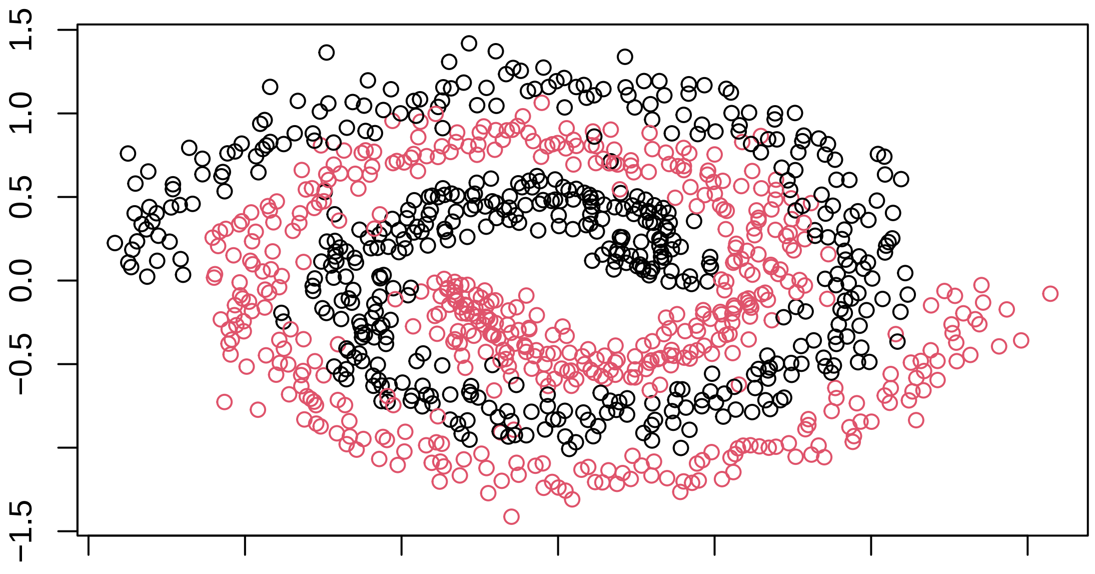
```
# Plotting decision boundaries
grid <- expand.grid(x_1=seq(min(new_data_x[,1]-0.2), max(new_data_x[,1]+0.2), by=0.05),
                    x_2=seq(min(new_data_x[,2]-0.2), max(new_data_x[,2]+0.2), by=0.05))

pred_grid_best_k <- as.numeric(knn(train = train_x, test = grid, cl = train_y, k = best_k))
plot_decision_boundary(train_x, train_y, pred_grid_best_k, grid)
```

```
## Warning: The ‘guide‘ argument in ‘scale_*()‘ cannot be ‘FALSE‘. This was deprecated in
## ggplot2 3.3.4.
## i Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

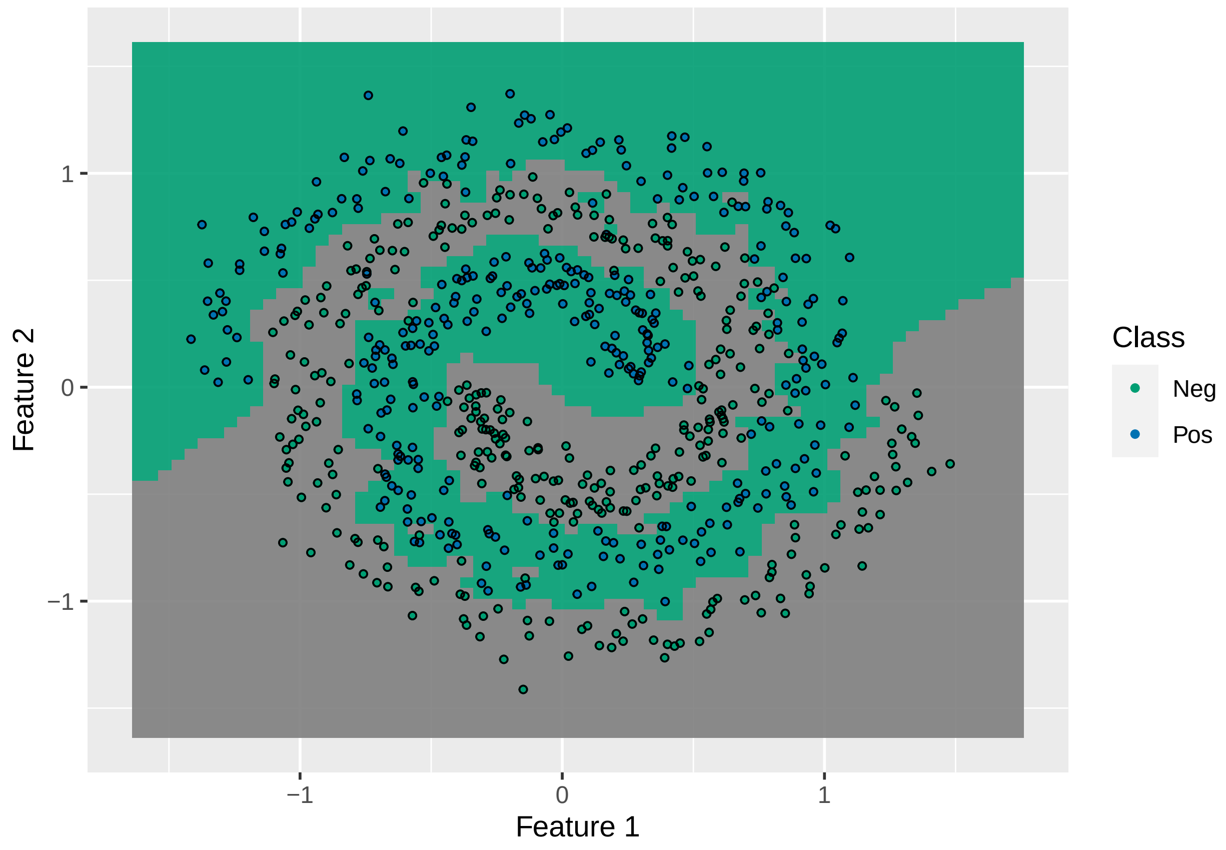


Try other K

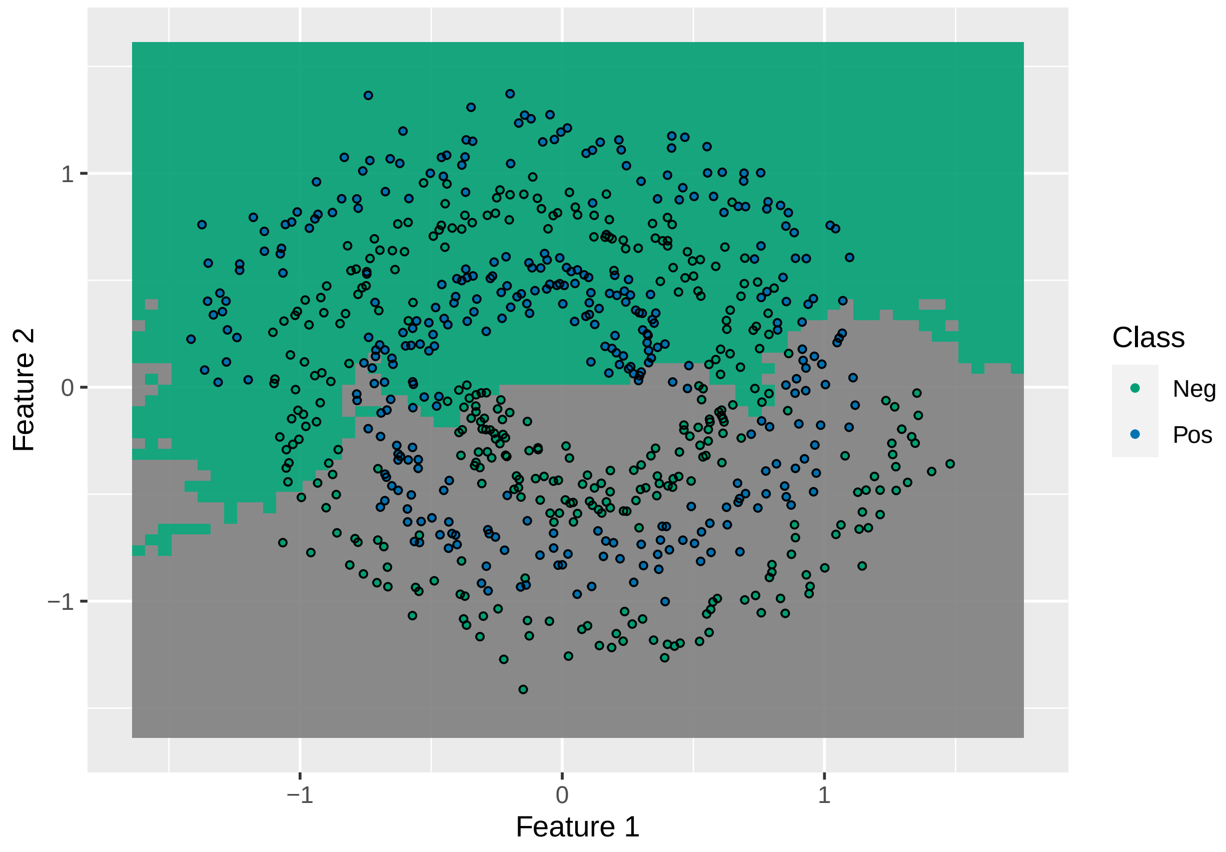
```
plot(new_data)
```



```
pred_grid_best_k <- as.numeric(knn(train = train_x, test = grid, cl = train_y, k = 1))
plot_decision_boundary(train_x, train_y, pred_grid_best_k, grid)
```



```r
pred_grid_best_k <- as.numeric(knn(train = train_x, test = grid, cl = train_y, k = 100))
plot_decision_boundary(train_x, train_y, pred_grid_best_k, grid)
```



Precision, Recall, and F1-score

```r
# Precision, Recall, and F1-score
precision <- function(predictions, actual) {
  tp <- sum(predictions == 1 & actual == 1)
  fp <- sum(predictions == 1 & actual == 2)
  return(tp / (tp + fp))
}

recall <- function(predictions, actual) {
  tp <- sum(predictions == 1 & actual == 1)
  fn <- sum(predictions == 2 & actual == 1)
  return(tp / (tp + fn))
}

f1_score <- function(predictions, actual) {
  prec <- precision(predictions, actual)
  rec <- recall(predictions, actual)
  return(2 * (prec * rec) / (prec + rec))
}

# Calculations for the test set
best_k_predictions_test <- knn(train = train_x, test = test_x, cl = train_y, k = best_k)
best_k_precision_test <- precision(best_k_predictions_test, test_y)
```

```r
best_k_recall_test <- recall(best_k_predictions_test, test_y)
best_k_f1_test <- f1_score(best_k_predictions_test, test_y)

list(
  Precision = best_k_precision_test,
  Recall = best_k_recall_test,
  F1 = best_k_f1_test
)
```

```
## $Precision
## [1] 0.9225806
##
## $Recall
## [1] 0.9597315
##
## $F1
## [1] 0.9407895
```