Homework 5

STAT 430, Spring 2017

Due: Friday, March 10 by 11:59 PM

Exercise 1

[15 points] For this homework we will use data found in `wisc-train.csv` and `wisc-test.csv` which contain train and test data respectively. `wisc.csv` is provided but not used. This is a modification of the Breast Cancer Wisconsin (Diagnostic) dataset from the UCI Machine Learning Repository. Only the first 10 feature variables have been provided. (And these are all you should use.)

- UCI Page
- Data Detail

You should consider coercing the response to be a factor variable. Do not use cross-validation for this exercise.

Use KNN. Consider $k = 1, 2, \ldots, 50$. Find the best k using both scaled and unscaled predictors. For both, plot train and test accuracy vs k on a single plot, report the best k, and report the associated test accuracy.

So, your answer will be two plots (both with two lines), two values of k, and two test accuracies. Was the scaling helpful?

Use the seed value provided below for this exercise.

Solution:

Note that some code, for plotting and summarizing, is hidden. See the .Rmd file for code.

```
set.seed(314)
```

```
# import data
wisc_train = read.csv("wisc-train.csv")
wisc_test = read.csv("wisc-test.csv")
```

```
# coerce to factor
wisc_train$class = as.factor(wisc_train$class)
wisc_test$class = as.factor(wisc_test$class)
```

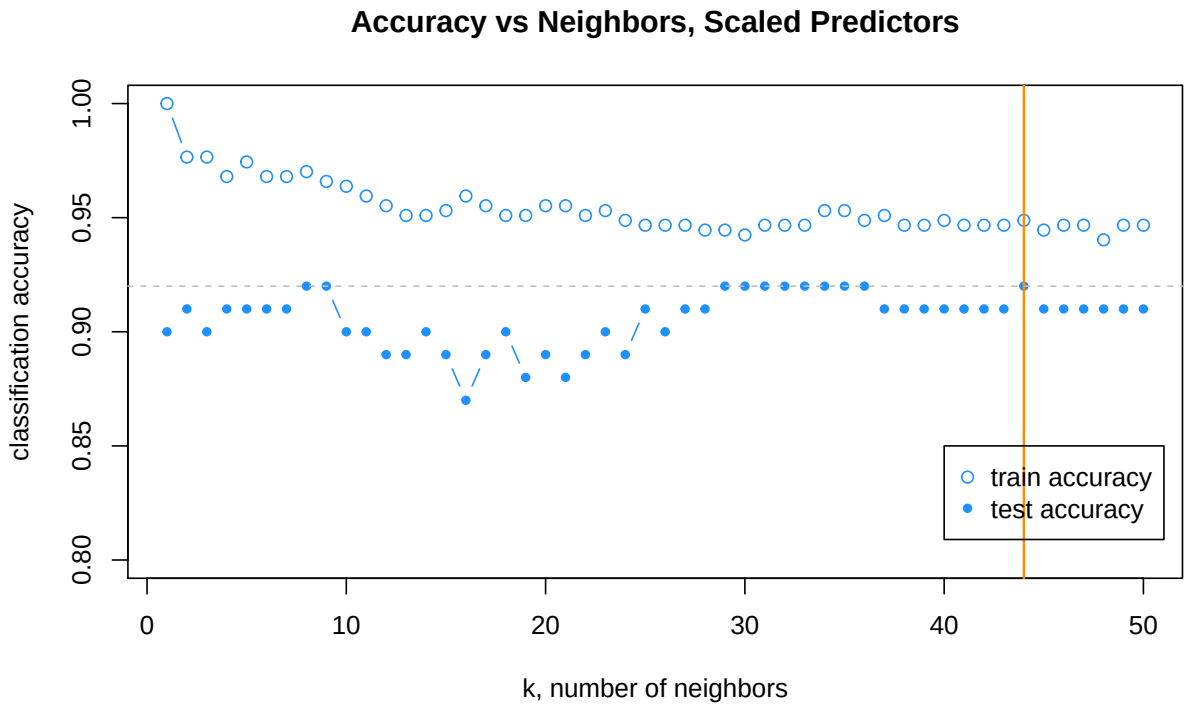
```
# training data
X_wisc_train = wisc_train[, -1]
y_wisc_train = wisc_train$class

# testing data
X_wisc_test = wisc_test[, -1]
y_wisc_test = wisc_test$class
```

```
library(class)
```

```
accuracy = function(actual, predicted) {
  mean(actual == predicted)
}
```

```
# setup for scaled results
k_to_try = 1:50
te_acc_k = rep(x = 0, times = length(k_to_try))
tr_acc_k = rep(x = 0, times = length(k_to_try))
```

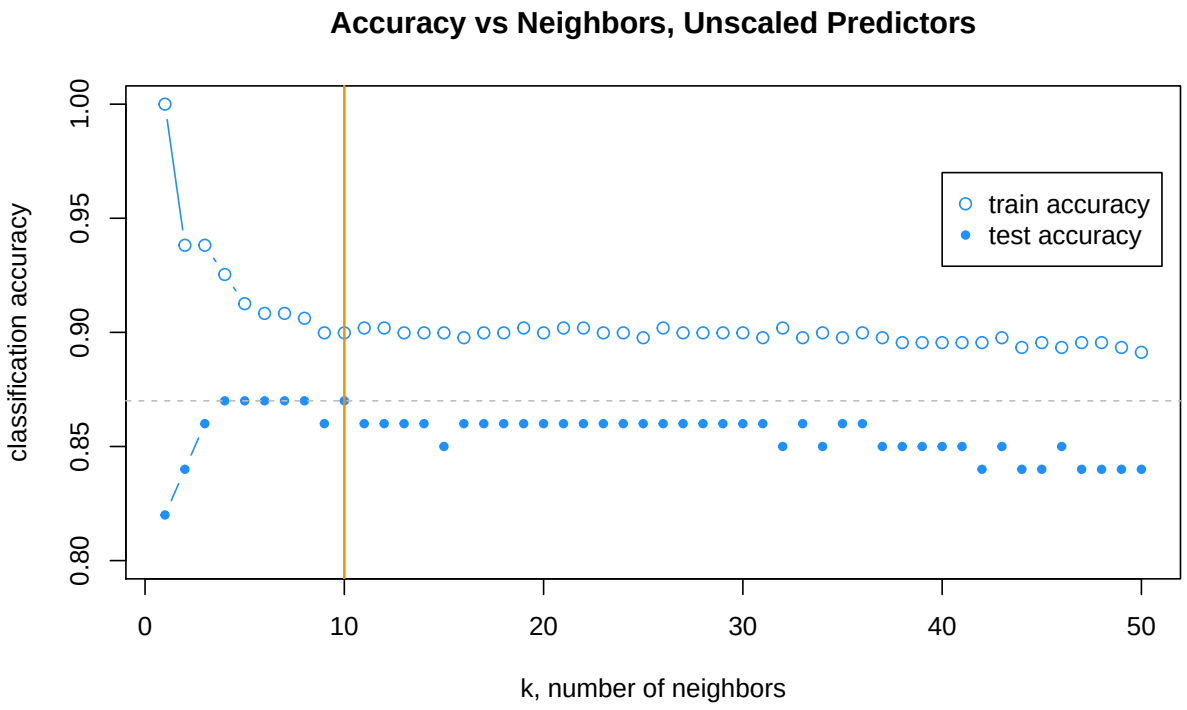
```r
# train scaled
for(i in seq_along(k_to_try)) {

  te_pred = knn(train = scale(X_wisc_train),
                test = scale(X_wisc_test),
                cl = y_wisc_train,
                k = k_to_try[i])

  tr_pred = knn(train = scale(X_wisc_train),
                test = scale(X_wisc_train),
                cl = y_wisc_train,
                k = k_to_try[i])

  te_acc_k[i] = accuracy(y_wisc_test, te_pred)
  tr_acc_k[i] = accuracy(y_wisc_train, tr_pred)

}
```




```r
# setup for unscaled results
set.seed(42)
k_to_try = 1:50
te_acc_k = rep(x = 0, times = length(k_to_try))
tr_acc_k = rep(x = 0, times = length(k_to_try))
```

```r
# train unscaled
for(i in seq_along(k_to_try)) {

  te_pred = knn(train = (X_wisc_train),
                test = (X_wisc_test),
```

```
                cl = y_wisc_train,
                k = k_to_try[i])

  tr_pred = knn(train = (X_wisc_train),
                test = (X_wisc_train),
                cl = y_wisc_train,
                k = k_to_try[i])

  te_acc_k[i] = accuracy(y_wisc_test, te_pred)
  tr_acc_k[i] = accuracy(y_wisc_train, tr_pred)

}
```



Predictor Scaling	k	Test Accuracy
Yes	44	0.92
No	10	0.87

We see better performance with the predictor scaling.

Exercise 2

[**15 points**] Calculate *train*, *test*, and *5-fold cross-validated* **accuracy** for both an **additive logistic regression** and **LDA**. You may use the `createFolds()` function from `caret`, but you may not use the `train()` function from `caret`.

Use your UIN in place of `uin`.

```r
uin = 123456789
set.seed(uin)
```

Solution:

```r
library(MASS)
```

```r
num_folds = 5
wisc_folds = caret::createFolds(wisc_train$class, k = num_folds)
```

```r
glm_acc = rep(0, times = num_folds)
lda_acc = rep(0, times = num_folds)
```

```r
for(i in seq_along(wisc_folds)) {

  # split for fold i
  train = wisc_train[-wisc_folds[[i]], ]
  valid = wisc_train[wisc_folds[[i]], ]

  # logistic regression
  glm_fit = glm(class ~ ., data = train, family = "binomial")
  glm_prob = predict(glm_fit, valid)
  glm_pred = ifelse(glm_prob > 0.5, "M", "B")
  glm_acc[i] = accuracy(actual = valid$class, predicted = glm_pred)

  # lda
  lda_fit = lda(class ~ ., data = train)
  lda_pred = predict(lda_fit, valid)$class
  lda_acc[i] = accuracy(actual = valid$class, predicted = lda_pred)

}
```

```r
# cv results
glm_cv = mean(glm_acc)
lda_cv = mean(lda_acc)
```

```r
# logistic
glm_fit = glm(class ~ ., data = wisc_train, family = "binomial")

# train acc
glm_train_prob = predict(glm_fit, wisc_train)
glm_train_pred = ifelse(glm_train_prob > 0.5, "M", "B")
glm_train_acc = accuracy(actual = wisc_train$class, predicted = glm_train_pred)

# test acc
glm_test_prob = predict(glm_fit, wisc_test)
glm_test_pred = ifelse(glm_test_prob > 0.5, "M", "B")
glm_test_acc = accuracy(actual = wisc_test$class, predicted = glm_test_pred)
```

```r
# lda
lda_fit = lda(class ~ ., data = wisc_train)

# train acc
lda_train_pred = predict(lda_fit, wisc_train)$class
lda_train_acc = accuracy(actual = wisc_train$class, predicted = lda_train_pred)
```

```r
# test acc
lda_test_pred = predict(lda_fit, wisc_test)$class
lda_test_acc = accuracy(actual = wisc_test$class, predicted = lda_test_pred)
```

Method	Train Accuracy	5-Fold CV Accuracy	Test Accuracy
Logistic	0.9594883	0.9466002	0.90
LDA	0.9402985	0.9381348	0.92