

# Homework 06

STAT 430, Fall 2017

Due: Friday, October 27, 11:59 PM

For this homework we will use data found in [wisc-trn.csv](#) and [wisc-tst.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.

You should use the `caret` package and training pipeline to complete this homework. Any time you use the `train()` function, first run `set.seed(1337)`.

---

## Exercise 1 (Tuning KNN with caret)

[6 points] Train a KNN model using all available predictors, **no data preprocessing**, 5-fold cross-validation, and a well chosen value of the tuning parameter. Consider  $k = 1, 3, 5, 7, \dots, 101$ . Store the tuned model fit to the training data for later use. Plot the cross-validated accuracies as a function of the tuning parameter.

**Solution:**

```
# import data
wisc_trn = read.csv("wisc-trn.csv")
wisc_tst = read.csv("wisc-tst.csv")

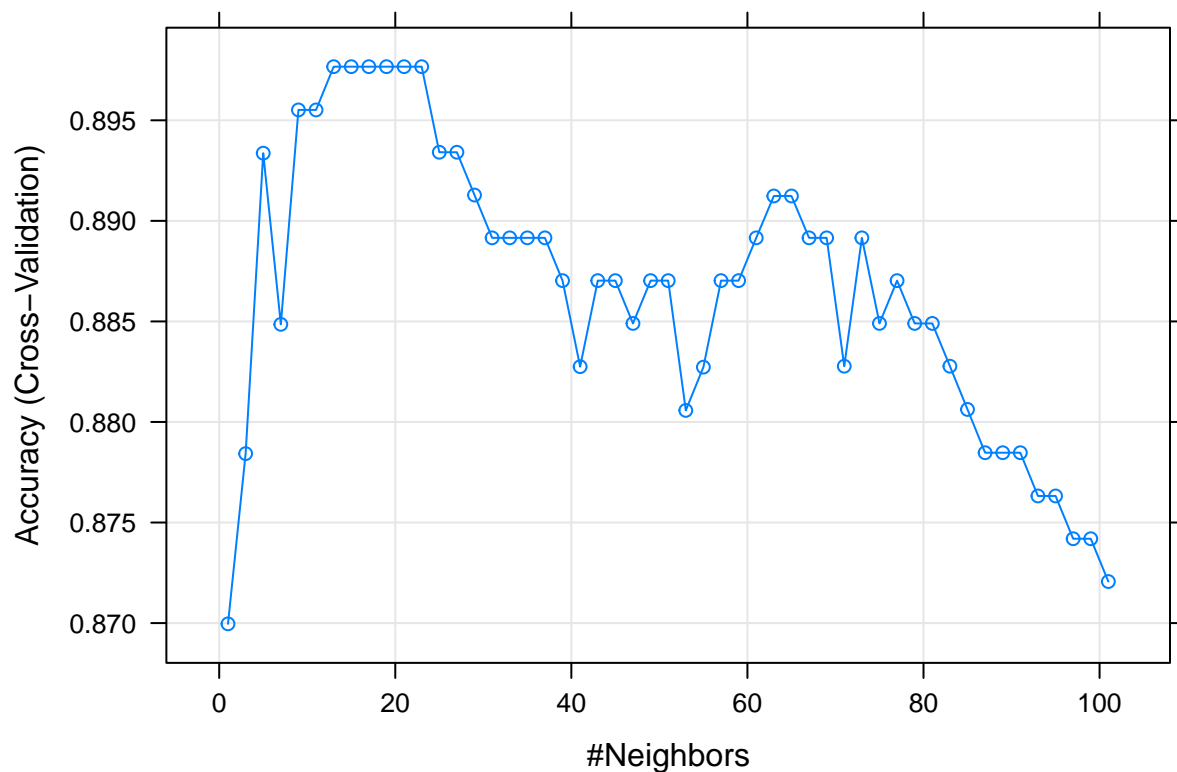
# coerce response to factor
wisc_trn$class = as.factor(wisc_trn$class)
wisc_tst$class = as.factor(wisc_tst$class)

# load all needed packages
library(caret)
library(randomForest)

# setup knn tuning
knn_control = trainControl(method = "cv", number = 5)
knn_tuning = expand.grid(k = seq(1, 101, by = 2))

# training and tune knn model
set.seed(1337)
wisc_knn = train(class ~ ., data = wisc_trn, method = "knn",
                  trControl = knn_control,
                  tuneGrid = knn_tuning)

plot(wisc_knn)
```



## Exercise 2 (More Tuning KNN with caret)

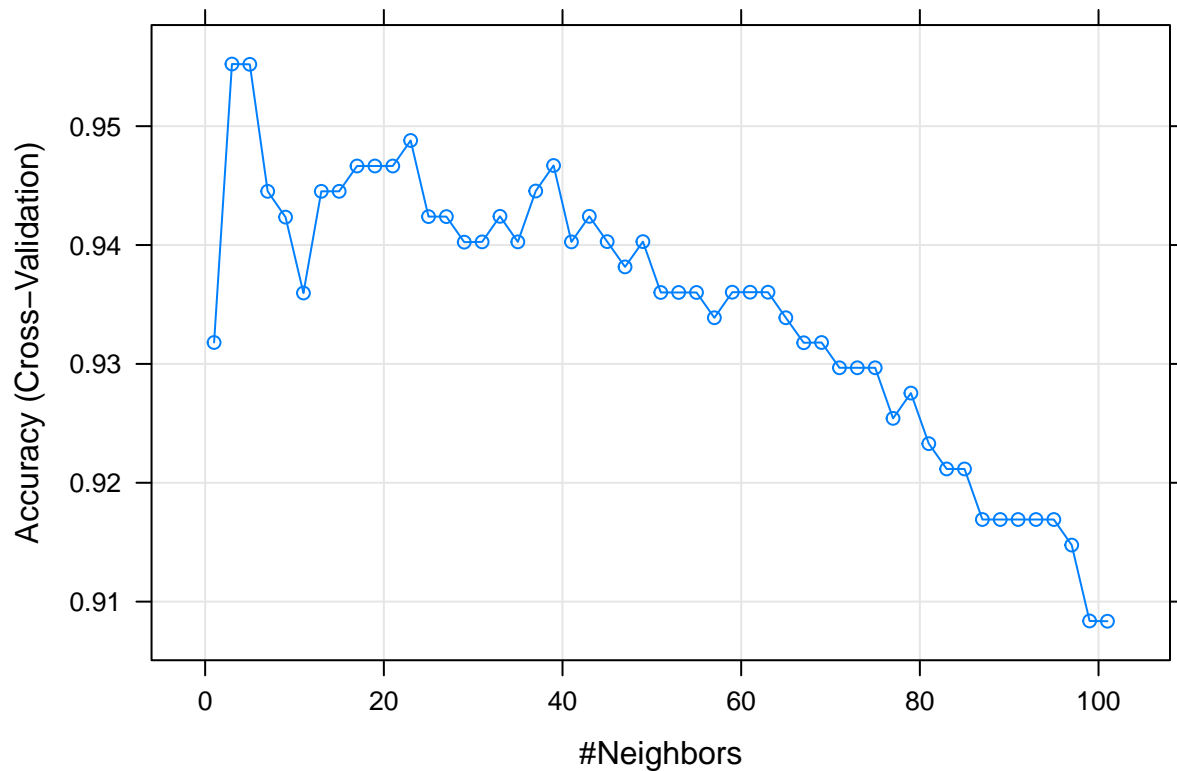
[6 points] Train a KNN model using all available predictors, predictors scaled to have mean 0 and variance 1, 5-fold cross-validation, and a well chosen value of the tuning parameter. Consider  $k = 1, 3, 5, 7, \dots, 101$ . Store the tuned model fit to the training data for later use. Plot the cross-validated accuracies as a function of the tuning parameter.

**Solution:**

```
# setup knn tuning
knn_control = trainControl(method = "cv", number = 5)
knn_preprocess = c("center", "scale")
knn_tuning = expand.grid(k = seq(1, 101, by = 2))

# training and tune knn model
set.seed(1337)
wisc_knn_scaled = train(class ~ ., data = wisc_trn, method = "knn",
                        trControl = knn_control,
                        preProcess = knn_preprocess,
                        tuneGrid = knn_tuning)
```

```
plot(wisc_knn_scaled)
```



### Exercise 3 (Random Forest?)

[6 points] Now that we've introduced `caret`, it becomes extremely easy to try different statistical learning methods. Train a random forest using all available predictors, **no data preprocessing**, 5-fold cross-validation, and well a chosen value of the tuning parameter. Using `caret` to perform the tuning, there is only a single tuning parameter, `mtry`. Consider `mtry` values between 1 and 10. Store the tuned model fit to the training data for later use. Report the cross-validated accuracies as a function of the tuning parameter using a well formatted table.

**Solution:**

```
# setup rf tuning
rf_control = trainControl(method = "cv", number = 5)
rf_tuning = expand.grid(mtry = c(1:10))

# training and tune knn model
set.seed(1337)
wisc_rf = train(class ~ ., data = wisc_trn, method = "rf",
               trControl = rf_control,
               tuneGrid = rf_tuning)

knitr::kable(wisc_rf$results, digits = 3)
```

mtry	Accuracy	Kappa	AccuracySD	KappaSD
1	0.940	0.871	0.021	0.046
2	0.947	0.885	0.026	0.058
3	0.940	0.872	0.025	0.054
4	0.949	0.890	0.023	0.051
5	0.940	0.872	0.024	0.052
6	0.940	0.872	0.024	0.051
7	0.938	0.867	0.022	0.046
8	0.942	0.876	0.028	0.058
9	0.940	0.872	0.036	0.075
10	0.938	0.867	0.033	0.069

## Exercise 4 (Concept Checks)

[1 point each] Answer the following questions based on your results from the three exercises. Format your answer to this exercise as a table with one column indicating the part, and the other column for your answer. See the `rmarkdown` source for a template of this table.

- What value of  $k$  is chosen for KNN without predictor scaling?
- What is the cross-validated accuracy for KNN without predictor scaling?
- What is the test accuracy for KNN without predictor scaling?
- What value of  $k$  is chosen for KNN **with** predictor scaling?
- What is the cross-validated accuracy for KNN **with** predictor scaling?
- What is the test accuracy for KNN **with** predictor scaling?
- Do you think that KNN is performing better with or without predictor scaling?
- What value of `mtry` is chosen for the random forest?
- Using the random forest, what is the (estimated) probability that the 10th observation of the test data is a cancerous tumor?
- Using the random forest, what is the (test) sensitivity?
- Using the random forest, what is the (test) specificity?
- Based on these results, is the random forest or KNN model performing better?

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

```
get_best_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best, ]
  rownames(best_result) = NULL
  best_result
}
```

```
a = wisc_knn$bestTune$k
b = get_best_result(wisc_knn)$Accuracy
```

```

c = calc_acc(actual = wisc_tst$class, predicted = predict(wisc_knn, wisc_tst))
d = wisc_knn_scaled$bestTune$k
e = get_best_result(wisc_knn_scaled)$Accuracy
f = calc_acc(actual = wisc_tst$class, predicted = predict(wisc_knn_scaled, wisc_tst))
g = "With scaling."
h = wisc_rf$bestTune$mtry
i = predict(wisc_rf, wisc_tst[10,], type = "prob")$M

wisc_rf_conmat = confusionMatrix(table(actual = wisc_tst$class,
                                       predicted = predict(wisc_rf, wisc_tst)),
                                positive = "M")

j = wisc_rf_conmat$byClass["Sensitivity"]
k = wisc_rf_conmat$byClass["Specificity"]
l = ifelse(calc_acc(actual = wisc_tst$class,
                    predicted = predict(wisc_rf, wisc_tst)) >
          calc_acc(actual = wisc_tst$class,
                    predicted = predict(wisc_knn_scaled, wisc_tst)),
          yes = "Random Forest",
          no = "KNN")

results = data.frame(
  part = LETTERS[1:12],
  answer = c(a,b,c,d,e,f,g,h,i,j,k,l)
)

knitr::kable(results)

```

part	answer
A	23
B	0.897666437886067
C	0.86
D	3
E	0.955227636696408
F	0.88
G	With scaling.
H	4
I	0.04
J	0.945945945945946
K	0.920634920634921
L	Random Forest