Homework 05

STAT 430, Fall 2017

Due: Friday, October 13, 11:59 PM

Please see the homework instructions document for detailed instructions and some grading notes. Failure to follow instructions will result in point reductions.

Exercise 1 (Detecting Cancer with KNN)

[7 points] For this exercise 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. Use KNN with all available predictors. For simplicity, do not scale the data. (In practice, scaling would slightly increase performance on this dataset.) Consider $k = 1, 3, 5, 7, \ldots, 51$. Plot train and test error vs k on a single plot.

Use the seed value provided below for this exercise.

```
set.seed(314)
library(FNN)
```

```
trn_data = read.csv('wisc-trn.csv')
tst_data = read.csv('wisc-tst.csv')
```

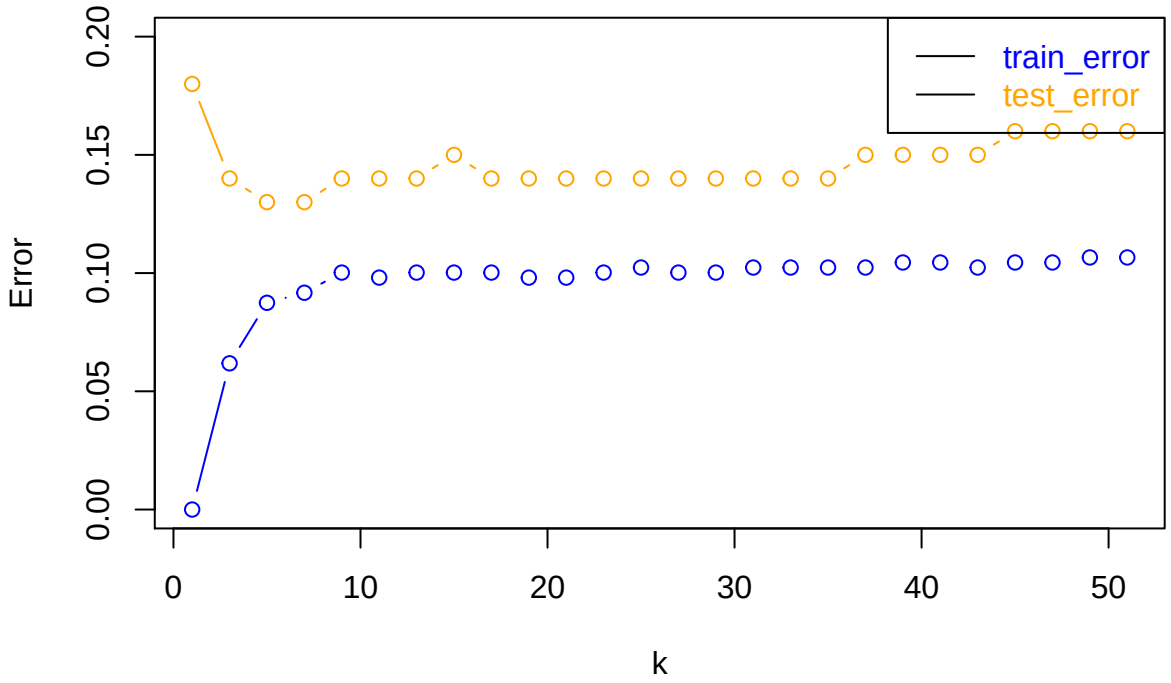
```
x_trn = trn_data[, -1]
x_tst = tst_data[, -1]
y_trn = trn_data$class
y_tst = tst_data$class
class(y_trn)
```

```
## [1] "factor"
```

```
get_knn_error = function(x_test, y_test, k) {
  pred = knn(
  train = x_trn,
  test = x_test,
  cl = y_trn,
  k = k
  )
  error = mean(pred != y_test)
}
```

```
k = seq(1, 51, by = 2)
train_error = sapply(k, get_knn_error, x_test = x_trn, y_test = y_trn)
test_error = sapply(k, get_knn_error, x_test = x_tst, y_test = y_tst)
```

```
plot(k,train_error, col = 'blue', type = 'b', ylim = c(0,0.2), ylab = 'Error')
lines(k, test_error, col = 'orange', type = 'b')
legend('topright',legend = c('train_error', 'test_error'),text.col = c('blue','orange'),
       cex = 1, lwd = 1)
```



Exercise 2 (Logistic Regression Decision Boundary)

[**5 points**] Continue with the cancer data from Exercise 1. Now consider an additive logistic regression that considers only two predictors, `radius` and `symmetry`. Plot the test data with `radius` as the x axis, and `symmetry` as the y axis, with the points colored according to their tumor status. Add a line which represents the decision boundary for a classifier using 0.5 as a cutoff for predicted probability.

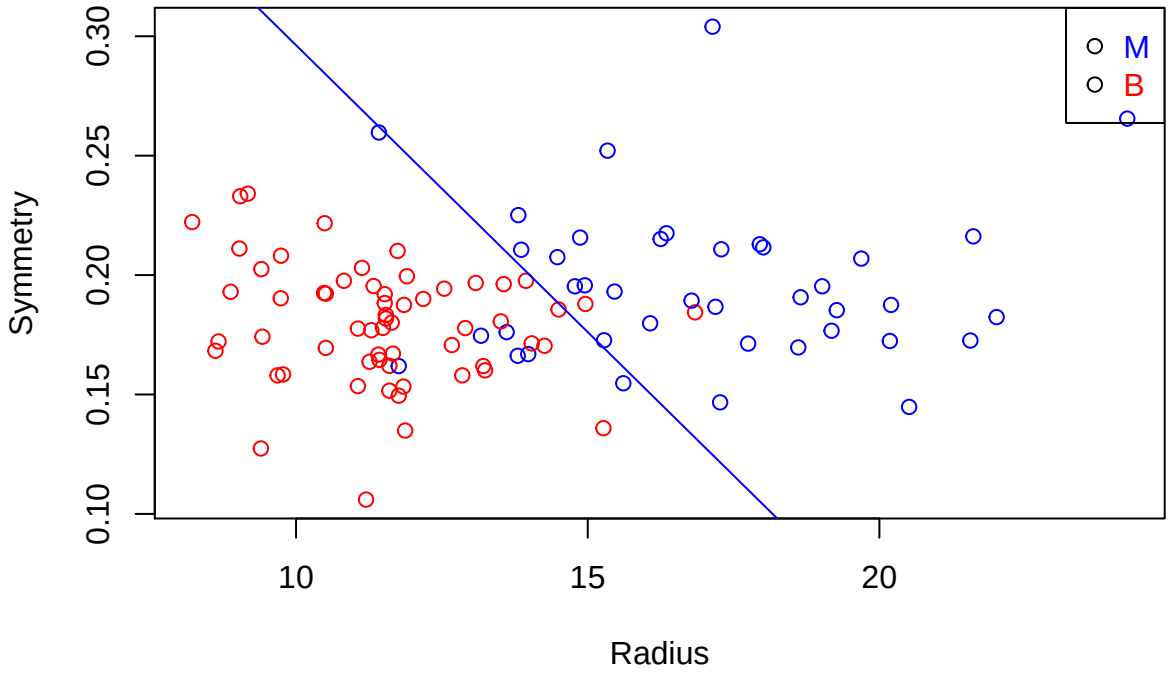
```
trn_new = data.frame(radius = trn_data$radius,symmetry = trn_data$symmetry, tumor = y_trn)
```

```
cut = 0.5
fit = glm(tumor ~ ., family = 'binomial', data = trn_new)
```

Since cutoff = 0.5, Thus

$$log(\frac{p(x)}{1-p(x)}) = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 = 0$$

```
slope = -coef(fit)[2]/coef(fit)[3]
intercept = -coef(fit)[1]/coef(fit)[3]
plot(tst_data$radius,tst_data$symmetry,type = 'p', col = c('red','blue')[tst_data$class],
     xlab = "Radius", ylab = "Symmetry")
abline(intercept, slope, col = "blue")
legend("topright",legend = c("M","B"),cex = 1, pch = 1, text.col = c('blue','red'))
```



Exercise 3 (Sensitivity and Specificity of Cancer Detection)

[5 points] Continue with the cancer data from Exercise 1. Again consider an additive logistic regression that considers only two predictors, `radius` and `symmetry`. Report test sensitivity, test specificity, and test accuracy for three classifiers, each using a different cutoff for predicted probability:

- $c = 0.1$
- $c = 0.5$
- $c = 0.9$

Consider `M` to be the "positive" class when calculating sensitivity and specificity. Summarize these results using a single well-formatted table.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
c = c(0.1,0.5,0.9)
model = glm(class ~ radius + symmetry, family = 'binomial', data = trn_data)
#pred = predict(model, newdata = tst_data, type = "response")
```

```
calc = function(m, type, c, above = 'M', below = 'B'){
  #model = glm(class ~ radius + symmetry, family = 'binomial', data = trn_data)
  pred = predict(m, newdata = tst_data, type = "response")
  pred = ifelse(pred > c, above, below)
  tst_tab = table(predicted = pred, actual = tst_data$class)
  if (type == 'sen'){
    (tst_tab[4]/(tst_tab[3] + tst_tab[4]))
  }
  else if (type == 'spe'){
```

```
    tst_tab[1]/(tst_tab[1] + tst_tab[2])
  }
  else {#if (type == "acc"){
    (tst_tab[1] + tst_tab[4]) / (tst_tab[1] + tst_tab[2] + tst_tab[3] + tst_tab[4])
  }
}
```

```
Sensitivity = sapply(c, calc, m = model, type = 'sen')
Specificity = sapply(c, calc, m = model, type = 'spe')
Accuracy = sapply(c, calc, m = model, type = "acc")
```

```
df = data.frame(cutoff = c, Sensitivity, Specificity, Accuracy)
knitr::kable(df)
```

cutoff	Sensitivity	Specificity	Accuracy
0.1	0.950	0.8000000	0.86
0.5	0.825	0.9666667	0.91
0.9	0.575	0.9833333	0.82

Exercise 4 (Comparing Classifiers)

[7 points] Use the data found in hw05-trn.csv and hw05-tst.csv which contain train and test data respectively. Use `y` as the response. Coerce `y` to be a factor after importing the data if it is not already.

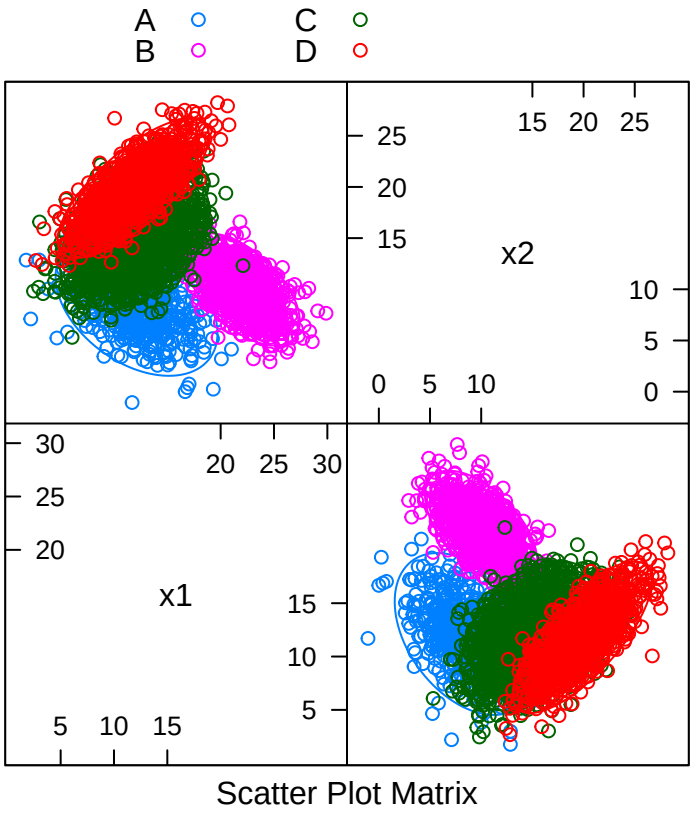
Create pairs plot with ellipses for the training data, then train the following models using both available predictors:

- Additive Logistic Regression
- LDA (with Priors estimated from data)
- LDA with Flat Prior
- QDA (with Priors estimated from data)
- QDA with Flat Prior
- Naive Bayes (with Priors estimated from data)

Calculate test and train error rates for each model. Summarize these results using a single well-formatted table.

```
train = read.csv("hw05-trn.csv")
test = read.csv("hw05-tst.csv")
```

```
x = train[,c("x1","x2")]
y = train$y
featurePlot(x=x, y=y, plot="ellipse",auto.key = list(columns = 3))
```



Scatter Plot Matrix

```
library(MASS)
library(e1071)
library(nnet)
model_multi = multinom(y ~ ., data = train, trace = FALSE)
trn_lda = lda(y ~ ., data = train)
lda_flat = lda(y ~., data = train, prior = c(1,1,1,1)/4)
trn_qda = qda(y ~., data = train)
qda_flat = qda(y ~., data = train, prior = c(1,1,1,1)/4)
trn_nb = naiveBayes(y ~., data = train)
```

```
calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}
calc_err = function(model, actual, data){
  pred = predict(model, newdata = data)$class
  calc_class_err(actual = actual, predicted = pred)
}
```

```
e0 = calc_class_err(actual = train$y, predicted = predict(model_multi, newdata = train))
e1 = calc_err(trn_lda, train$y, data = train)
e2 = calc_err(lda_flat, train$y, data = train)
e3 = calc_err(trn_qda, train$y, data = train)
e4 = calc_err(qda_flat, train$y, data = train)
e5 = calc_class_err(actual = train$y, predicted = predict(trn_nb, newdata = train,type = "class"))
train_err = c(e0, e1, e2, e3, e4, e5)
```

```
te0 = calc_class_err(actual = test$y, predicted = predict(model_multi, newdata =test,))
te1 = calc_err(trn_lda, test$y, data = test)
te2 = calc_err(lda_flat, test$y, data = test)
```

```
te3 = calc_err(trn_qda, test$y, data = test)
te4 = calc_err(qda_flat, test$y, data = test)
te5 = calc_class_err(actual = test$y, predicted = predict(trn_nb, test, type = "class"))
test_err = c(te0, te1, te2, te3, te4, te5)
```

```
df1 = data.frame(models = c("logistic model","lda", "lda_flat", "qda", "qda_flat", "navie bayes"),
                 train_error = round(train_err,5), test_error = test_err)
knitr::kable(df1)
```

models	train_error	test_error
logistic model	0.14822	0.17425
lda	0.16200	0.19825
lda_flat	0.19067	0.16875
qda	0.14778	0.16925
qda_flat	0.17911	0.14000
navie bayes	0.17333	0.20000

Exercise 5 (Concept Checks)

[**1 point each**] Answer the following questions based on your results from the three exercises.

(a) Which k performs best in Exercise 1?

```
k[test_error == test_error[which.min(test_error)]]
```

```
## [1] 5 7
```

Since k = 5 and k = 7 both have the lowest test error, k = 7 perform best with more complexity and less variate

(b) In Exercise 4, which model performs best?

qda with flat prior model performs best

(c) In Exercise 4, why does Naive Bayes perform poorly?

Because there is relation between x1 and x2 and Naive Bayes does not consider that

(d) In Exercise 4, which performs better, LDA or QDA? Why?

QDA

Because QDA considers different variance between each class, while LDA shares the same variance

(e) In Exercise 4, which prior performs better? Estimating from data, or using a flat prior? Why?

QDA with flat prior

```
table(test$y) / length(test$y)
```

```
##
##    A    B    C    D
## 0.25 0.25 0.25 0.25
```

because flat prior gives each class the same possibility in the dataset as the test dataset does.

(f) In Exercise 4, of the four classes, which is the easiest to classify?

B, Becuase B has the smallest overlap area with the other 3 classes.

(g) [**Not Graded**] In Exercise 3, which classifier would be the best to use in practice?

0.1

It is better to reduce the occurence of False Positive