Problem Set 2

API222: Big Data & Machine Learning

Brendan Hellweg

10/14/2021

Problem 1

LDA and Logistic regression both have linear decision boundaries and will perform similarly poorly when applied to nonlinear decision boundaries. QDA and KNN both have nonlinear decision boundaries. Depending on the type of nonlinearity of the Bayes Decision Boundary, either QDA or KNN May perform better. KNN will perform better if the assumption of normalcy does not hold, while QDA performs better when the assumption of normalcy holds.

Problem 2

Problem 2A See the table below.

```
##   X1 X2 X3      Y distance
## 1  0  3  2   Blue     3.61
## 2  0  2  3   Blue     3.61
## 3  0  4  1 Orange     4.12
## 4  1  3  1  Green     3.32
## 5  1  5  0 Orange     5.10
## 6 -1  4  2 Orange     4.58
## 7  0  4  2 Orange     4.47
```

Problem 2B The prediction with `K = 1` is `Green`. The decision criteria for KNN is to select the `K` data points closest to the test point and assign a value based on the observed value. In this case, the one nearest neighbor (where the Euclidian distance is minimized) has property `Y = Green`.

Problem 2C The prediction with `K = 3` is `Blue`. For the three nearest neighbors (where the Euclidian distance is minimized), two have the property `Y = Blue` and one has property `Y = Green`. Different KNN tests will have different decision criteria for inconsistent values, but typically the value with the most or majority observations in the set is what's selected.

Problem 2D If the Bayes decision boundary is highly nonlinear, we'd want a small `K` so we can be sensitive to nonlinearities.

Problem 3

When $\lambda = 0$, the Lasso model becomes equivalent to the residual sum of squares, therefore becoming identical to the OLS model. When λ approaches infinity, all coefficients approach zero and the model becomes a constant function.

Problem 4

While Lasso uses $||\beta||1$ with the sums of absolute values of βj, Ridge uses $||\beta||2$ with the sums of squares of βj. Using $||\beta||1$ forces some of the coefficient estimates to reach exactly zero, meaning that Lasso produces a sparce and therefore more interpretable model. Ridge, on the other hand, produces a dense and therefore harder to interpret model.

If all I care about is the predictive ability and I don't care at all about the level of interpretability, I would use Cross Validation to compare the two potential models (after also using it to identify the optimal values of λ to design each model). Cross Validation works by estimating the test error rate by holding aside subsets of the training observations in the fitting process and then applying the model to those saved subsets. The metric to compare is the test MSE.

Problem 5

We cannot draw this conclusion, as we do not know where along the ROC curve we want to minimize the number of false positives. Model A could start steeper and level off sooner than Model B, meaning that for early parts of the ROC curve it outperforms Model B. Conversely, Model B could dominate throughout the ROC curve. Therefore, we cannot make a conclusion from the information provided. It's usually a good idea to use AUC to determine which model works better, but there are edge cases such as the example described above.

Data Questions

Problem 1

```
c("The dimensions are ",
  nrow(prostate), " by ", ncol(prostate),".") %>%
str_c(.,collapse = '')
```

```
## [1] "The dimensions are 97 by 10."
```

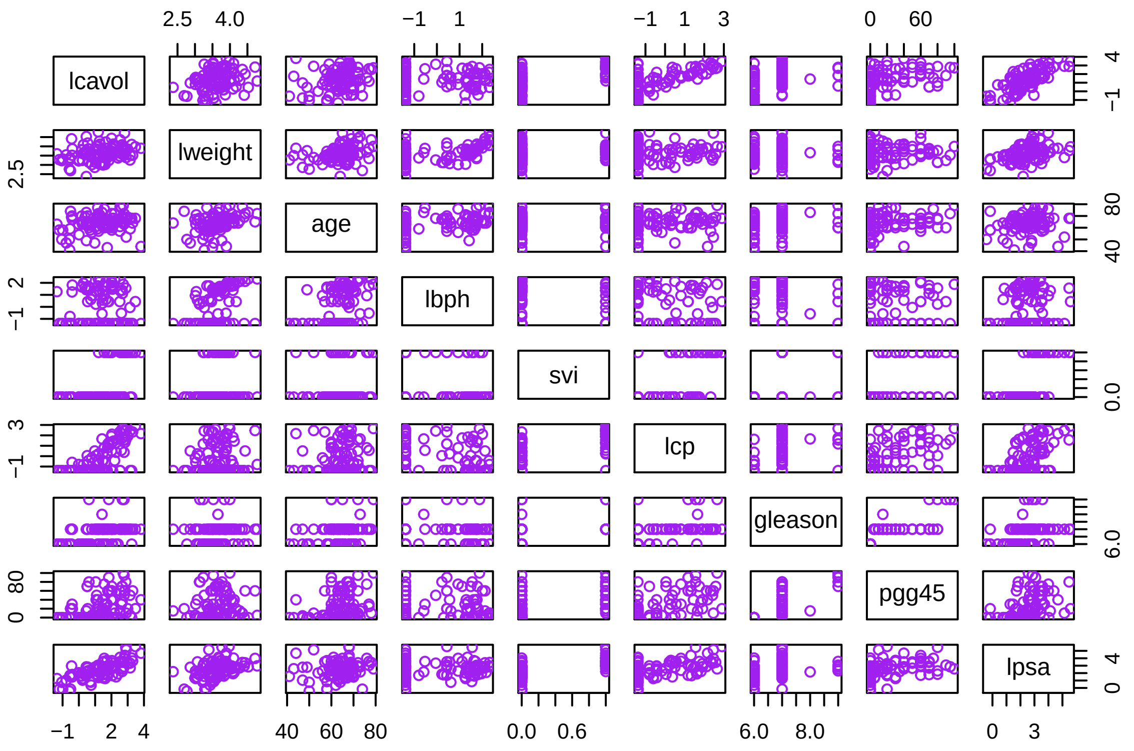
The final column represents whether the entry belongs in the test or the treatment set.

Problem 2

```
round(cor(prostate[1:9]),2)
```

```
##         lcavol lweight  age  lbph   svi   lcp gleason pgg45 lpsa
## lcavol    1.00    0.28 0.22  0.03  0.54  0.68    0.43  0.43 0.73
## lweight   0.28    1.00 0.35  0.44  0.16  0.16    0.06  0.11 0.43
## age       0.22    0.35 1.00  0.35  0.12  0.13    0.27  0.28 0.17
## lbph      0.03    0.44 0.35  1.00 -0.09 -0.01    0.08  0.08 0.18
## svi       0.54    0.16 0.12 -0.09  1.00  0.67    0.32  0.46 0.57
## lcp       0.68    0.16 0.13 -0.01  0.67  1.00    0.51  0.63 0.55
## gleason   0.43    0.06 0.27  0.08  0.32  0.51    1.00  0.75 0.37
## pgg45     0.43    0.11 0.28  0.08  0.46  0.63    0.75  1.00 0.42
## lpsa      0.73    0.43 0.17  0.18  0.57  0.55    0.37  0.42 1.00
```

```
pairs((prostate[1:9]), col = "purple", row1attop = T)
```



The Gleason variable has a small number of possible values in this dataset, so the results will be constrained to a small number of bands, which are vertical because the other variable has a strong degree of variation along a given value of gleason.

Problem 3

```
seatbelts2 <- drop_na(USSeatBelts)

nrow(seatbelts)-nrow(seatbelts2)
```

```
## [1] 209
```

Problem 4

```
## [1] "state"    "year"     "speed65"  "speed70"  "drinkage" "alcohol"  "enforce"
```

```
## [1] "state: 51"
## [1] "year: 15"
## [1] "speed65: 2"
## [1] "speed70: 2"
## [1] "drinkage: 2"
## [1] "alcohol: 2"
## [1] "enforce: 3"
```

Problem 5

```
seatbelts2 %>%
  group_by(state) %>%
  summarise(mean_fatalities = signif(mean(fatalities),3)) %>%
  slice_max(.,order_by = mean_fatalities,n = 1)
```

```
## # A tibble: 1 x 2
##   state mean_fatalities
##   <fct>           <dbl>
## 1 MS             0.0287
```

```
seatbelts2 %>%
  group_by(state) %>%
  summarise(mean_fatalities = signif(mean(fatalities),3)) %>%
  slice_min(.,order_by = mean_fatalities,n = 1)
```

```
## # A tibble: 1 x 2
##   state mean_fatalities
##   <fct>           <dbl>
## 1 RI             0.0105
```

Problem 6

```
seatbelts3 <- seatbelts2 %>%
  select(-c(1:2)) %>%
  dummy_cols(.,select_columns = c('speed65',
                                  'speed70',
                                  'drinkage',
                                  'alcohol',
                                  'enforce')) %>%
  select(-c(4:7,10)) %>%
  select(c(2,1,3:16))

seatbelt_lm <- lm(fatalities~.,seatbelts3)
summary(seatbelt_lm)
```

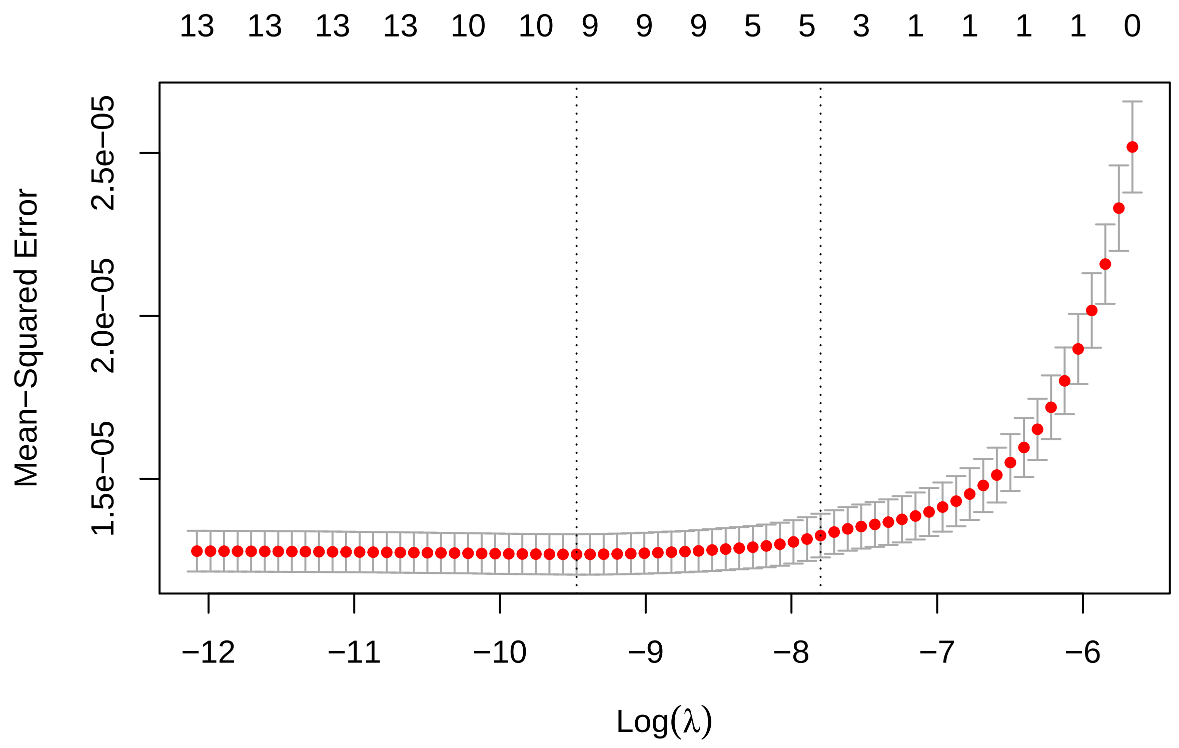
```
##
## Call:
## lm(formula = fatalities ~ ., data = seatbelts3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0097759 -0.0022532 -0.0002876  0.0020290  0.0140265
##
## Coefficients: (5 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.828e-02  3.889e-03   9.841  < 2e-16 ***
## miles             -1.491e-09  3.353e-09  -0.445 0.656810
## seatbelt          -7.528e-04  1.696e-03  -0.444 0.657267
## income            -7.959e-07  4.869e-08 -16.345  < 2e-16 ***
## age               -5.546e-05  1.136e-04  -0.488 0.625587
## speed65_no        -1.593e-04  4.623e-04  -0.345 0.730528
## speed65_yes               NA         NA      NA       NA
## speed70_no        -2.232e-03  5.352e-04  -4.171 3.54e-05 ***
## speed70_yes               NA         NA      NA       NA
## drinkage_no        1.174e-03  9.092e-04   1.291 0.197160
## drinkage_yes              NA         NA      NA       NA
## alcohol_no         1.824e-03  4.803e-04   3.798 0.000162 ***
## alcohol_yes               NA         NA      NA       NA
## enforce_no        -9.388e-04  5.194e-04  -1.807 0.071252 .
## enforce_primary    9.109e-04  5.273e-04   1.727 0.084680 .
## enforce_secondary         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.003532 on 545 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5067
## F-statistic: 58.01 on 10 and 545 DF,  p-value: < 2.2e-16
```

```
str_c("Adjusted R^2: ",round(100*seatbelt_lm[["residuals"]][["98"]],2))
```

```
## [1] "Adjusted R^2: 0.51"
```

Problem 7

```
set.seed(222)
cvfit <- cv.glmnet(data.matrix(seatbelts3[2:16]),data.matrix(seatbelts3[1]))
plot(cvfit)
```



```
## [1] "Lamda with lowest mean cross-validation error: 7.68e-05"

## [1] "Cross-validation error: 1.27e-05"

## [1] "Standard error for this Lamda value: 6.21e-07"

## [1] "Largest lamda value within 1se: 4.1000e-04"
```

Problem 8

```
#Randomize the data
seatbelts3<-seatbelts3[sample(nrow(seatbelts3)),]
#Create 5 equally size folds
folds <- cut(seq(1,nrow(seatbelts3)),breaks=5,labels=FALSE)
dataframe <- as.data.frame(c(1:100))
#Perform 5 fold cross validation
for(i in 1:5){
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- seatbelts3[testIndexes, ]
    trainData <- seatbelts3[-testIndexes, ]
    columnframe = as.data.frame(c())
for(k in 1:100){
#Perform KNN for every value of 0<k<101
```

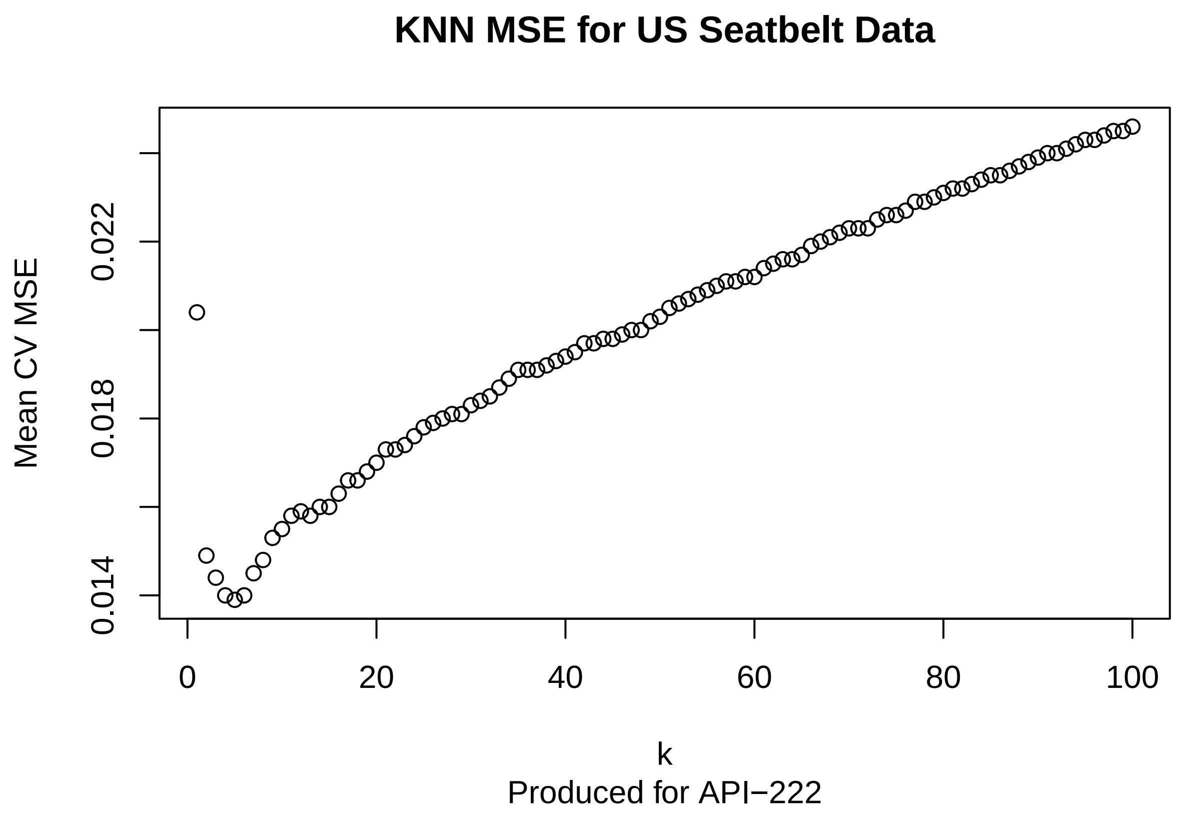
```r
    pr <- knn.reg(test = testData,train = trainData,k = k,y = trainData$seatbelt)
    tb <- as.data.frame(pr$pred) %>%
    mutate(pr = pr$pred, scores = testData$seatbelt) %>% mutate(sqerror = (.$pr-.$scores)^2)
    columnframe <- rbind(columnframe, mean(as.numeric(as.character(tb$sqerror))))}
    dataframe <- cbind(dataframe, (columnframe))}
kfold <- as.data.frame(dataframe) %>%
   mutate(mean = signif(rowMeans(.[2:5]),3),k=.$`c(1:100)`) %>% select(-c(1:6))
slice_min(kfold,order_by = mean,n = 1)
```

```
##     mean k
## 1 0.0139 5
```

Mean squared error is minimized at the above `k` value. Note that this will be randomized so the result will change each time it is run.

Problem 9

```r
plot(x = kfold$k,y = kfold$mean,
     main = "KNN MSE for US Seatbelt Data",
     sub = "Produced for API-222",
     xlab = 'k',ylab = "Mean CV MSE")
```


KNN MSE for US Seatbelt Data
Mean CV MSE
k
Produced for API−222