HW1

Napon Jatusripitak

1/29/2018

1 Tidy Verse!

1.1 Compare glm and glmnet

glm

```
## (Intercept)         cyl        disp          hp        drat          wt
## 12.30337416 -0.11144048  0.01333524 -0.02148212  0.78711097 -3.71530393
##        qsec          vs          am        gear        carb
##  0.82104075  0.31776281  2.52022689  0.65541302 -0.19941925
```

glmnet

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 35.3137765
## cyl         -0.8714512
## disp         .
## hp          -0.0101174
## drat         .
## wt          -2.5944368
## qsec         .
## vs           .
## am           .
## gear         .
## carb         .
```

1.2 Compare glm and glmnet with tidy()

glm

```
##           term    estimate   std.error  statistic    p.value
## 1  (Intercept) 12.30337416 18.71788443  0.6573058 0.51812440
## 2          cyl -0.11144048  1.04502336 -0.1066392 0.91608738
## 3         disp  0.01333524  0.01785750  0.7467585 0.46348865
## 4           hp -0.02148212  0.02176858 -0.9868407 0.33495531
## 5         drat  0.78711097  1.63537307  0.4813036 0.63527790
## 6           wt -3.71530393  1.89441430 -1.9611887 0.06325215
## 7         qsec  0.82104075  0.73084480  1.1234133 0.27394127
## 8           vs  0.31776281  2.10450861  0.1509915 0.88142347
## 9           am  2.52022689  2.05665055  1.2254035 0.23398971
## 10        gear  0.65541302  1.49325996  0.4389142 0.66520643
## 11        carb -0.19941925  0.82875250 -0.2406258 0.81217871
```

glmnet

```
##          term step   estimate lambda dev.ratio
## 1 (Intercept)    1 35.3137765      1 0.8087808
```

```
## 2         cyl    1 -0.8714512      1 0.8087808
## 3          hp    1 -0.0101174      1 0.8087808
## 4          wt    1 -2.5944368      1 0.8087808
```

2 Regression

2.1 OLS

2.1.1 Estimate y ~ x1 + x2 + u

```
ols.reg <- lm(as.numeric(result)-1 ~ temp + bp, data = sick_data)
ols.reg %>% tidy() %>% knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-5.2134563	0.5141439	-10.14007	0
temp	0.0628185	0.0050579	12.41987	0
bp	-0.0082865	0.0004702	-17.62194	0

2.1.2 Get predicted values. y_hat > 0.5 = positive for disease. y_hat < 0.5 = negative for disease

```
y_hat <- fitted(ols.reg)
ols.reg.predict <- sick_data$result
ols.reg.predict[y_hat >= 0.5]= "Positive"
ols.reg.predict[y_hat < 0.5]= "Negative"
```

We can construct a confusion matrix relating the predicted values to actual values

```
knitr:: kable(table(ols.reg.predict, sick_data$result))
```

	Negative	Positive
Negative	950	36
Positive	0	14

```
mean(ols.reg.predict == sick_data$result)
```

```
## [1] 0.964
```

Which tells us that OLS incorrectly predicted the outcomes of 36 out of 1000 people.

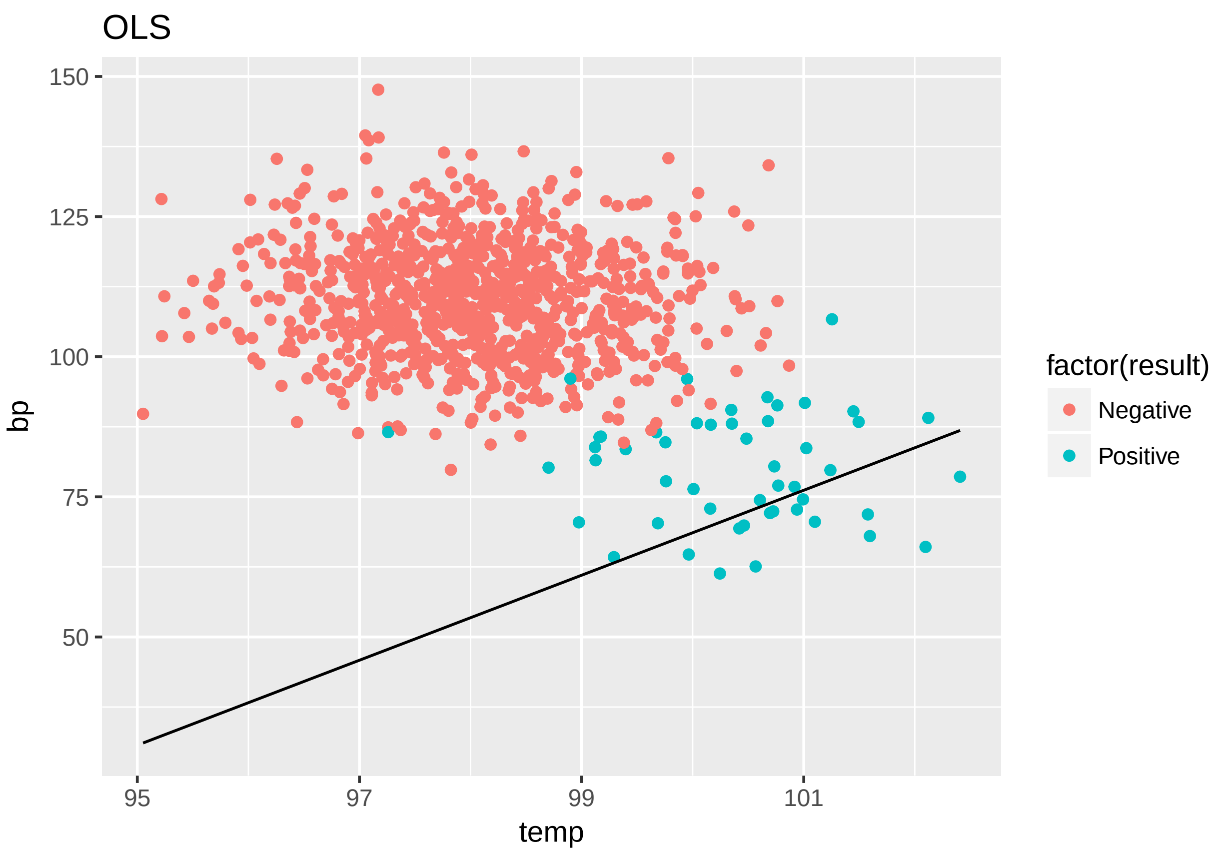
2.1.3 Generate the equation of the line where y_hat = 0.5 as a function of blood pressure and temperature

Equation is given by 0.5 = B0 + B1X1 + B2X2

```
intercept <- (0.5 - coef(ols.reg)[1]) / coef(ols.reg)[3]
slope <- -1 * coef(ols.reg)[2] / coef(ols.reg)[3]
line_0.5 <- function(x) {
```

```
  intercept + slope*x
}
```

2.1.4 Display the blood pressure and temperature data on a single scatterplot, using either color or shape to distinguish between positive and negative results. Add the line you calculated from the previous step.



2.2 Logit

2.2.1 Estimate y ~ x1 + x2 + u

```
logit.reg <- glm(result ~ temp + bp,data= sick_data ,family = "binomial")
logit.reg %>% tidy() %>% knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-199.3266800	46.8077535	-4.258412	2.06e-05
temp	2.3139723	0.4922912	4.700414	2.60e-06
bp	-0.3499531	0.0638023	-5.484962	0.00e+00

2.2.2 Get predicted values. y_hat > 0.5 = positive for disease. y_hat < 0.5 = negative for disease.

```r
y_hat <- fitted(logit.reg)
logit.reg.predict <- sick_data$result
logit.reg.predict[y_hat >= 0.5]= "Positive"
logit.reg.predict[y_hat < 0.5]= "Negative"
```

We can construct a confusion matrix relating the predicted values to the actual values.

```r
knitr::kable(table(logit.reg.predict, sick_data$result))
```

	Negative	Positive
Negative	946	4
Positive	4	46

```r
mean(logit.reg.predict == sick_data$result)
```

```
## [1] 0.992
```

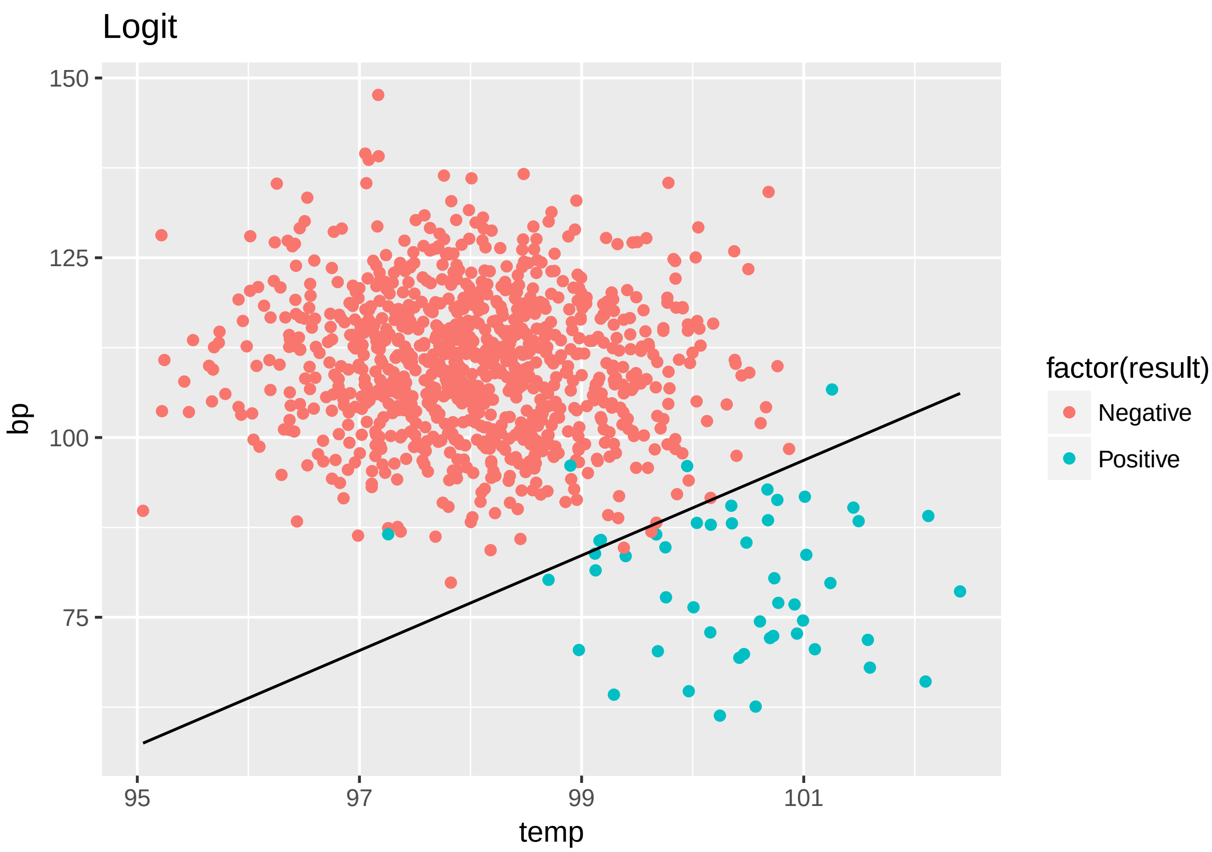
Which tells us that OLS incorrectly predicted the outcomes of 8 out of 1000 people.

2.2.3 Generate the equation of the line where y_hat = 0.5 as a function of blood pressure and temperature

Equation is given by 0.5 = B0 + B1X1 + B2X2

```r
intercept <- (0.5 - coef(logit.reg)[1]) / coef(logit.reg)[3]
slope <- -1 * coef(logit.reg)[2] / coef(logit.reg)[3]
line_0.5 <- function(x) {
  intercept + slope*x
}
```

2.2.4 Display the blood pressure and temperature data on a single scatterplot, using either color or shape to distinguish between positive and negative results. Add the line you calculated from the previous step.



2.3 Ridge

2.3.1 Estimate y ~ x1 + x2 + u

```
ridge.reg <- glmnet(x = as.matrix(sick_data[, -1]), y = sick_data$result, alpha = 0, lambda = 1, family
ridge.reg %>% tidy() %>% knitr::kable()
```

term	step	estimate	lambda	dev.ratio
(Intercept)	1	-10.1496223	1	0.1113747
temp	1	0.0837432	1	0.1113747
bp	1	-0.0094347	1	0.1113747

2.3.2 Get predicted values. y_hat > 0.5 = positive for disease. y_hat < 0.5 = negative for disease.

```
y_hat <- predict(ridge.reg, newx = as.matrix(sick_data[, -1]))
ridge.reg.predict <- sick_data$result
ridge.reg.predict[y_hat >= 0.5]= "Positive"
```

```r
ridge.reg.predict[y_hat < 0.5]= "Negative"
```

We can construct a confusion matrix relating the predicted values and the actual values.

```r
knitr:: kable(table(ridge.reg.predict, sick_data$result))
```

	Negative	Positive
Negative	950	50
Positive	0	0

```r
mean(ridge.reg.predict == sick_data$result)
```

```
## [1] 0.95
```

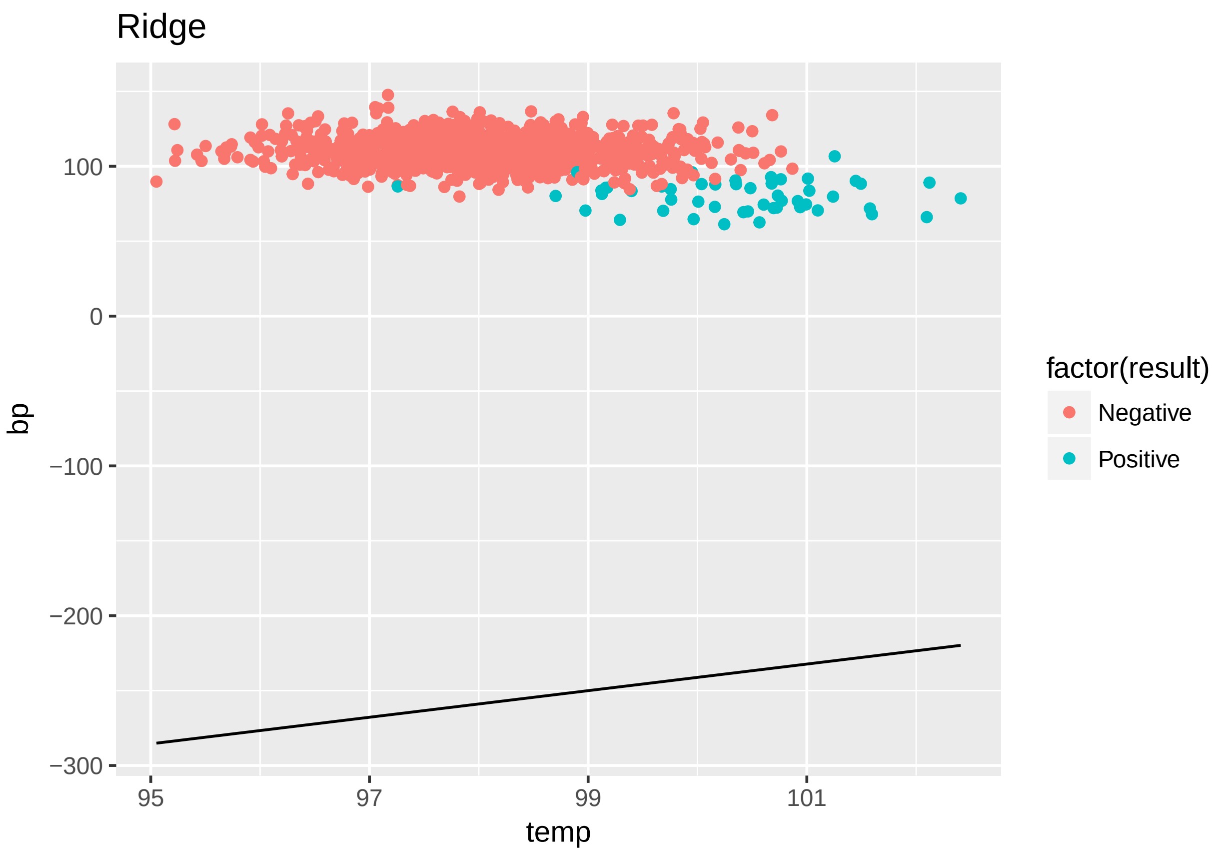
Which tells us that OLS incorrectly predicted the outcomes of 50 out of 1000 people.

2.3.3 Generate the equation of the line where y_hat = 0.5 as a function of blood pressure and temperature

Equation is given by 0.5 = B0 + B1X1 + B2X2

```r
intercept <- (0.5 - coef(ridge.reg)[1]) / coef(ridge.reg)[3]
slope <- -1 * coef(ridge.reg)[2] / coef(ridge.reg)[3]
line_0.5 <- function(x) {
  intercept + slope*x
}
```

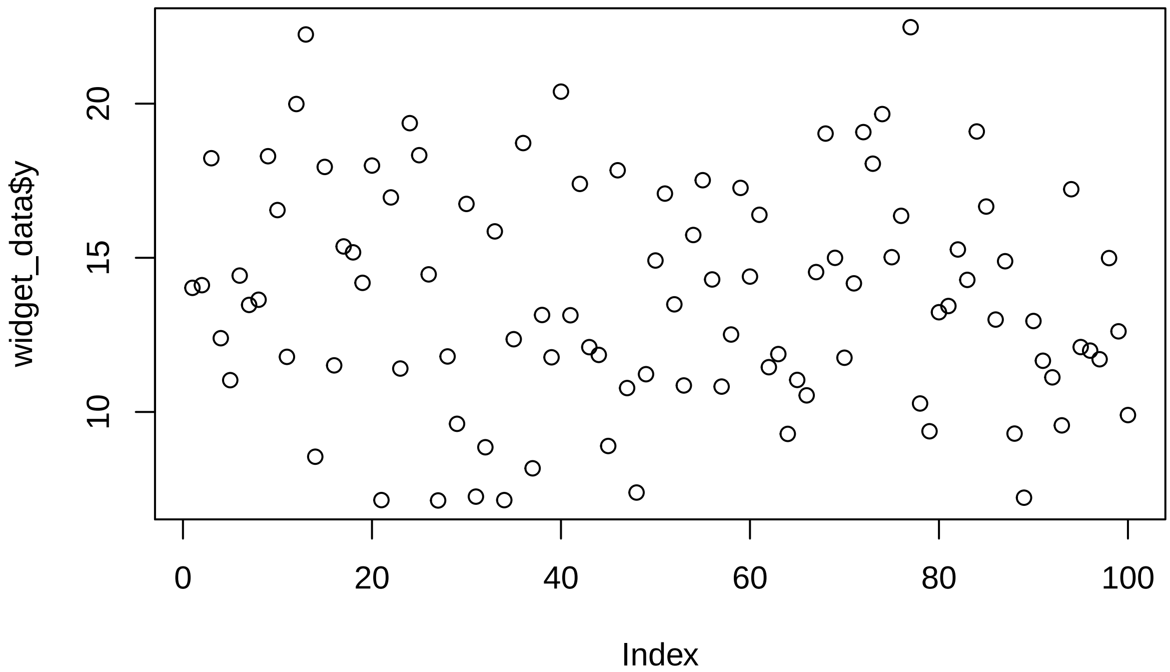
2.3.4 Display the blood pressure and temperature data on a single scatterplot, using either color or shape to distinguish between positive and negative results. Add the line you calculated from the previous step.



3 Regularization/Selection

3.1 Load the data, and plot the dependent variable y.

```
# Load the data, and plot the dependent variable y.
widget_data <- read.csv("widget_data.csv")
plot(widget_data$y)
```



3.2 Ridge

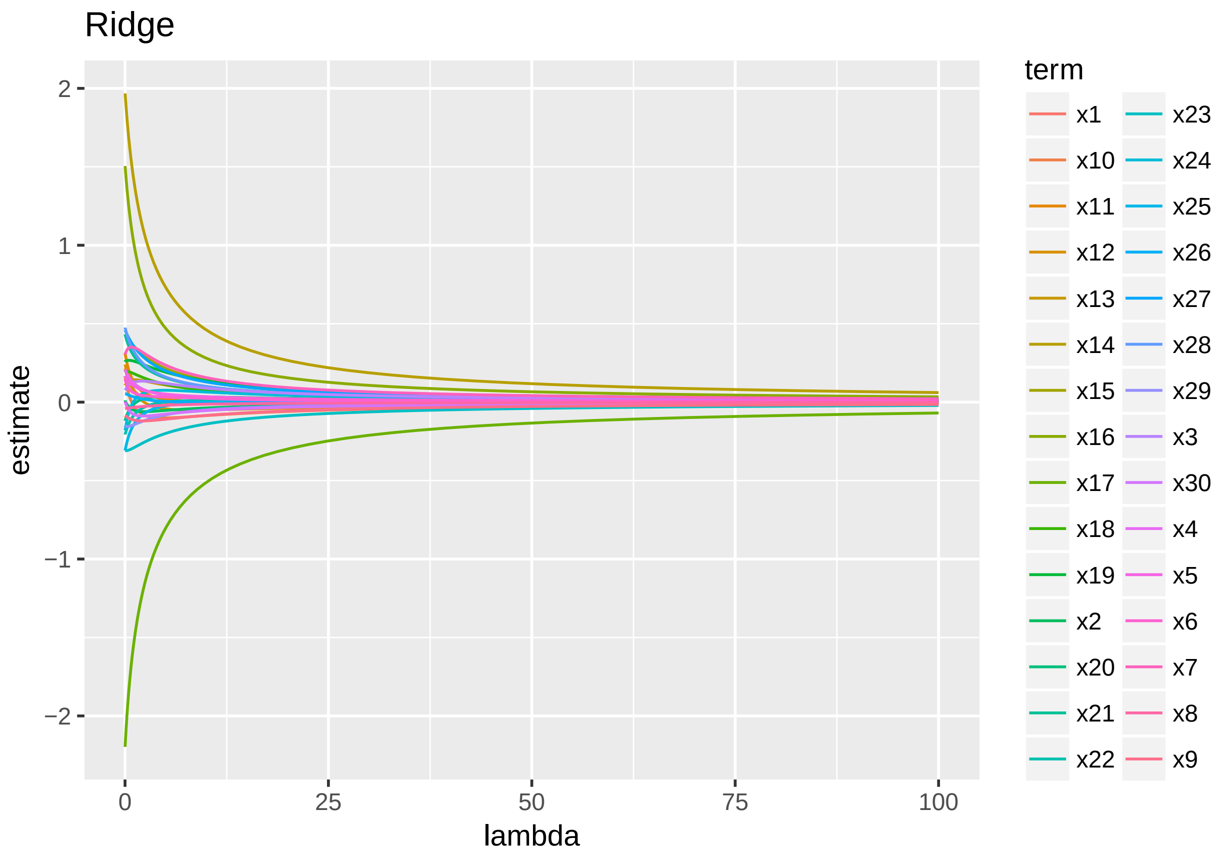
3.2.1 Use glmnet() from the glmnet package to estimate a ridge regression with a sequence of labda from 0.01 to 100

```
lambdas <- 10^seq(2, -2, length =100)
ridge.reg.widget <- glmnet(x = as.matrix(widget_data[, -1]), y = widget_data$y, alpha = 0, lambda = lam
```

3.2.2 Use tidy from the broom package to extract the data from the regression into a useable format and use ggplot2 to plot the coefficient estimates as lambda changes.

Drop intercepts

```
ridge.plot <- filter(ridge.reg.widget %>% tidy() , term != "(Intercept)")
```



Plot

3.2.3 Use cross validation with cv.glmnet to pick the value of lambda that will minimize mean squared error, and give the coefficients you get when using that lambda

```
ridge.five.fold <- cv.glmnet(x = as.matrix(widget_data[,-1]), y = widget_data$y, alpha = 0, lambda = lar
optimal.lambda <- ridge.five.fold$lambda.min
```

The lambda that minimizes the mean squared error is

```
optimal.lambda
```

```
## [1] 0.5994843
```

The coefficients that are obtained using this lambda is

```
ridge.five.fold %>% coef()
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  6.53974886
## x1           0.04687432
## x2          -0.05866287
## x3           0.13271906
## x4           0.09735664
## x5           0.08676961
## x6           0.04747757
## x7           0.33391005
## x8          -0.02994267
## x9          -0.11855448
```

```
## x10         -0.06100574
## x11          0.08792688
## x12          0.03005903
## x13          0.14093433
## x14          1.23337802
## x15          0.32651230
## x16          0.86109456
## x17         -1.33764461
## x18          0.16832553
## x19          0.25097807
## x20          0.00799584
## x21          0.25687193
## x22         -0.05382336
## x23         -0.27427951
## x24          0.03918495
## x25         -0.10217817
## x26          0.02503140
## x27          0.31831485
## x28          0.27791927
## x29         -0.13167963
## x30         -0.08673519
```

3.3 Lasso

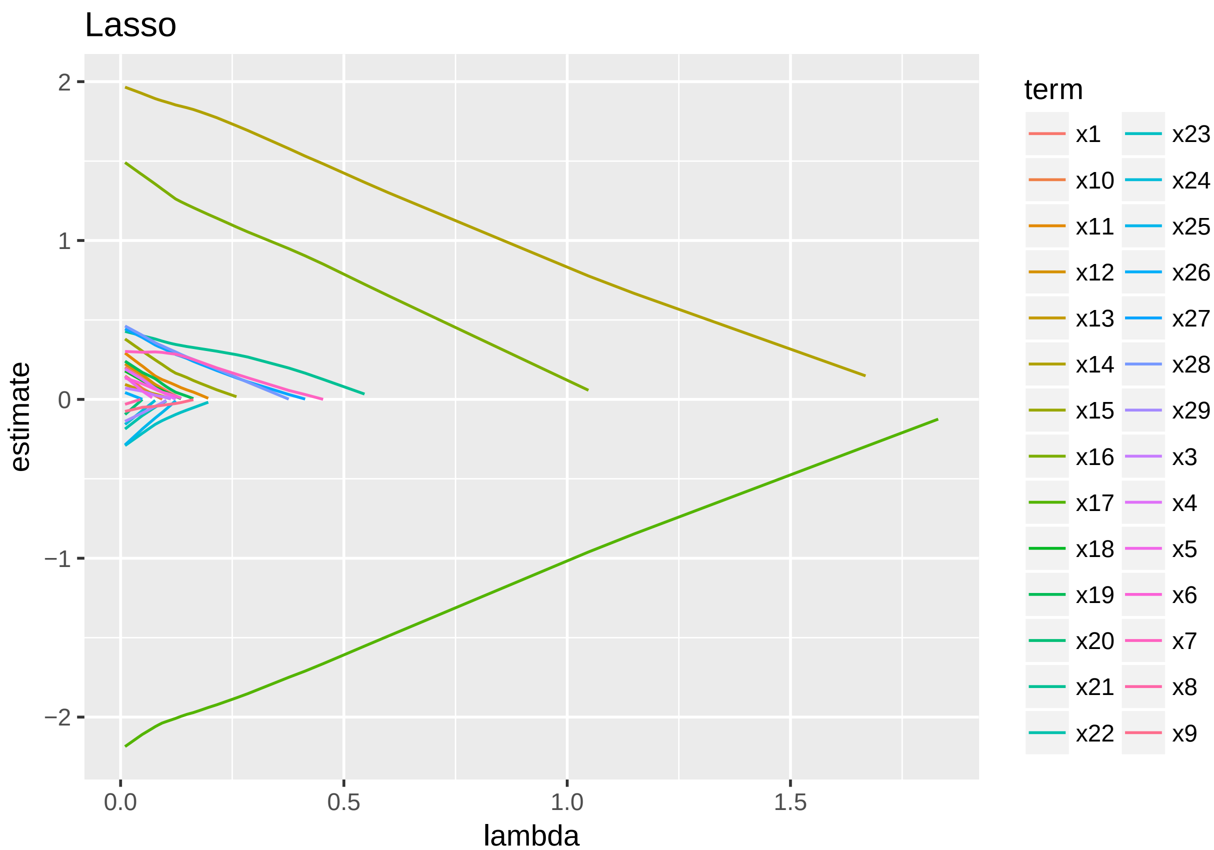
3.3.1 Use glmnet() from the glmnet package to estimate a lasso regression with a sequence of labda from 0.01 to 100

```
lambdas <- 10^seq(2, -2, length =100)
lasso.reg.widget <- glmnet(x = as.matrix(widget_data[, -1]), y = widget_data$y, alpha = 1, lambda = lamb
```

3.3.2 Use tidy from the broom package to extract the data from the regression into a useable format and use ggplot2 to plot the coefficient estimates as lambda changes.

Drop intercepts

```
lasso.plot <- filter(lasso.reg.widget %>% tidy() , term != "(Intercept)")
```



Plot

3.3.3 Use cross validation with cv.glmnet to pick the value of lambda that will minimize mean squared error, and give the coefficients you get when using that lambda

```
lasso.five.fold <- cv.glmnet(x = as.matrix(widget_data[,-1]), y = widget_data$y, alpha = 1, lambda = lar
optimal.lambda <- lasso.five.fold$lambda.min
```

The lambda that minimizes the mean squared error is

```
optimal.lambda
```

```
## [1] 0.2154435
```

The coefficients that are obtained using this lambda is

```
lasso.five.fold %>% coef()
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  6.11395143
## x1           .
## x2           .
## x3           .
## x4           .
## x5           .
## x6           .
## x7           .
## x8           .
## x9           .
```

```
## x10          .
## x11          .
## x12          .
## x13          .
## x14          1.42734604
## x15          .
## x16          0.79110149
## x17         -1.61115385
## x18          .
## x19          .
## x20          .
## x21          0.08182361
## x22          .
## x23          .
## x24          .
## x25          .
## x26          .
## x27          .
## x28          .
## x29          .
## x30          .
```

3.4 Ridge vs Lasso

Both ridge and lasso models shrink the coefficient estimates, biasing them toward 0 as lambda increases. However, while the ridge regression does not shrink the estimates to an absolute zero, the lasso regression does this, effectively removing less relevant predictors out of the model altogether. At the optimal lambda, the ridge regression retains all predictors, whereas the lasso regression retains only 7 predictors.

4 Classification

4.1 Split the data into 2/3 training and 1/3 test data.

```
smp <- sample(300, 200)
train <- pol_data[smp, ]
test <- pol_data[-smp, ]
```

4.2 Naive Bayes

4.2.1 Estimate each model using the training data only.

```
NB <- naiveBayes(group ~ ., data = train)
summary(NB)
```

```
##         Length Class  Mode
## apriori 2      table  numeric
## tables  3      -none- list
## levels  2      -none- character
## call    4      -none- call
```

4.2.2 Use the model to predict the outcome in the test data.

```
ypredict <- predict(NB, test)
ypredict
```

```
##   [1] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##   [6] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [11] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [16] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [21] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [26] Socialcrat   Socialcrat   Politicalist Socialcrat   Socialcrat
##  [31] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [36] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [41] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [46] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [51] Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##  [56] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [61] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [66] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [71] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [76] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [81] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [86] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [91] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [96] Socialcrat   Politicalist Politicalist Politicalist Politicalist
## Levels: Politicalist Socialcrat
```

4.2.3 Create a table of the predicted classes against the real classes.

We can construct a confusion matrix

```
knitr::kable(table(ypredict, test$group))
```

	Politicalist	Socialcrat
Politicalist	44	1
Socialcrat	1	54

```
mean(ypredict == test$group)
```

```
## [1] 0.98
```

Which tells us that the model incorrectly predicts the outcomes of 2 out of 100 individuals.

4.3 SVM

4.3.1 Estimate each model using the training data only.

```
tune.out <- tune(svm, group ~ ., data = train, kernel="linear", ranges = list(cost=c(0.001, 0.01, 0.1, 
tune.out$best.model
```

```
##
## Call:
```

```
## best.tune(method = svm, train.x = group ~ ., data = train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.3333333
##
## Number of Support Vectors:  30
```

4.3.2 Use the model to predict the outcome in the test data.

```
ypredict <- predict(tune.out$best.model, test)
ypredict
```

```
##            3            4            5            6            8
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##            9           10           11           13           15
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##           18           20           21           22           26
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##           33           34           41           42           44
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##           45           46           48           53           59
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##           62           63           65           76           77
##   Socialcrat   Socialcrat Politicalist   Socialcrat   Socialcrat
##           80           81           88           89           96
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##           98           99          104          105          110
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##          112          113          115          116          119
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##          121          123          125          126          130
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##          131          139          144          146          149
##   Socialcrat   Socialcrat   Socialcrat   Socialcrat   Socialcrat
##          153          157          158          164          166
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          167          170          171          177          180
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          192          198          203          208          209
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          210          215          218          219          220
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          224          226          227          229          241
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          242          245          246          251          254
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          256          265          266          270          273
## Politicalist Politicalist Politicalist Politicalist Politicalist
```

```
##          274          275          277          281          282
## Politicalist Politicalist Politicalist Politicalist Politicalist
##          284          287          291          294          295
##   Socialcrat Politicalist Politicalist Politicalist Politicalist
## Levels: Politicalist Socialcrat
```

4.3.3 Create a table of the predicted classes against the real classes.

We can construct a confusion matrix

```r
knitr::kable(table(ypredict, test$group))
```

	Politicalist	Socialcrat
Politicalist	44	1
Socialcrat	1	54

```r
mean(ypredict == test$group)
```

```
## [1] 0.98
```

Which tells us that the model incorrectly predicts the outcomes of 2 out of 100 individuals. The result appears to be the same with the Naive Bayes model. However, after multiple tries with different set.seed(), the Naive Bayes model seems to outperform the SVM model.