HW1

James Zhao

April 17, 2019

Q1 - Regression

OLS

a)

```
setwd("~/GitHub/MMSS_311_2")
sick <- read.csv("sick_data.csv")
sick$RESULT.DUMMY <- ifelse (sick$result == "Positive", 1, 0)
OLS <- lm(RESULT.DUMMY~temp+bp, data = sick)
summary(OLS)
```

```
##
## Call:
## lm(formula = RESULT.DUMMY ~ temp + bp, data = sick)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32785 -0.09918 -0.02229  0.05700  0.82096
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.2134563  0.5141439  -10.14   <2e-16 ***
## temp         0.0628185  0.0050579   12.42   <2e-16 ***
## bp          -0.0082865  0.0004702  -17.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1695 on 997 degrees of freedom
## Multiple R-squared:  0.3966, Adjusted R-squared:  0.3954
## F-statistic: 327.7 on 2 and 997 DF,  p-value: < 2.2e-16
```

b)

```
sick$PREDICTED.VALUE <- fitted(OLS)
sick$PREDICTED.OUTCOME <- ifelse(sick$PREDICTED.VALUE >= 0.5, "Positive", "Negative")
sick$PREDICTED.ACCURACY <- ifelse(sick$PREDICTED.OUTCOME == sick$result, 1, 0)
accuracy.ols <- mean(sick$PREDICTED.ACCURACY)
accuracy.ols
```

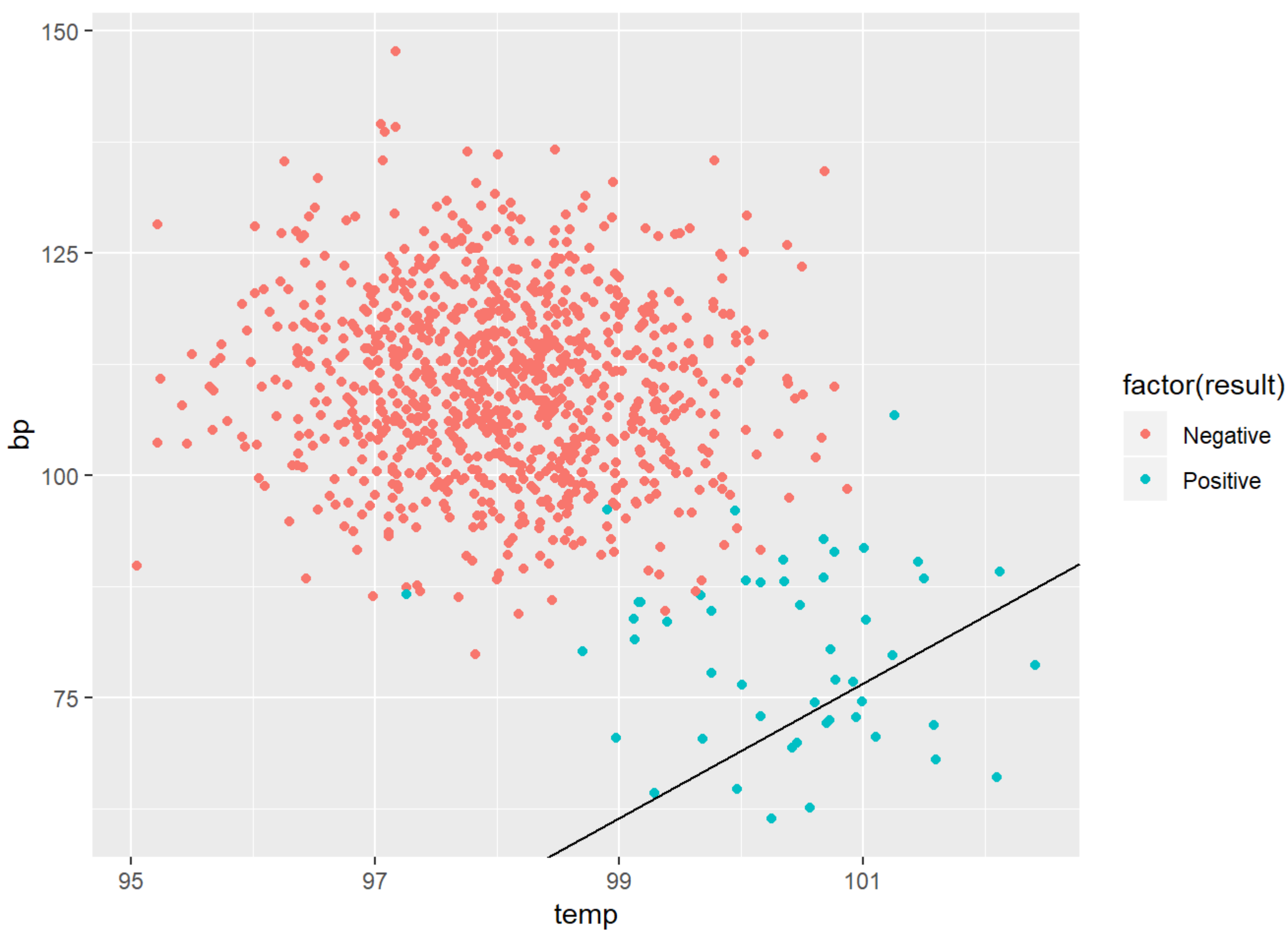
```
## [1] 0.964
```

The OLS regression correctly predicts the results 96.4% of the time.

c) The equation of the line is -5.7134563 + 0.00628185temp - 0.0082865bp = 0

d)

```
library(ggplot2)
ggplot(sick, aes(temp, bp)) +
  geom_point()+
  geom_point(aes(colour = factor(result)))+
  geom_abline(intercept = -689.1506, slope = 7.580824232)
```



Logit

a)

```
logit <- glm(RESULT.DUMMY ~ temp+bp, data = sick, family = binomial)
summary(logit)
```

```
##
## Call:
## glm(formula = RESULT.DUMMY ~ temp + bp, family = binomial, data = sick)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.62332  -0.02253  -0.00462  -0.00093   3.02311
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -199.3267    46.8077  -4.258 2.06e-05 ***
## temp           2.3140     0.4923   4.700 2.60e-06 ***
## bp            -0.3499     0.0638  -5.485 4.14e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 397.030  on 999  degrees of freedom
## Residual deviance:  53.837  on 997  degrees of freedom
## AIC: 59.837
##
## Number of Fisher Scoring iterations: 10
```

b)

```
sick$PREDICTED.VALUE.LOGIT <- fitted(logit)
sick$PREDICTED.OUTCOME.LOGIT <- ifelse(sick$PREDICTED.VALUE.LOGIT >= 0.5, "Positive", "Negative"
)
sick$PREDICTED.ACCURACY.LOGIT <- ifelse(sick$PREDICTED.OUTCOME.LOGIT == sick$result, 1, 0)
accuracy.logit <- mean(sick$PREDICTED.ACCURACY.LOGIT)
accuracy.logit
```

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

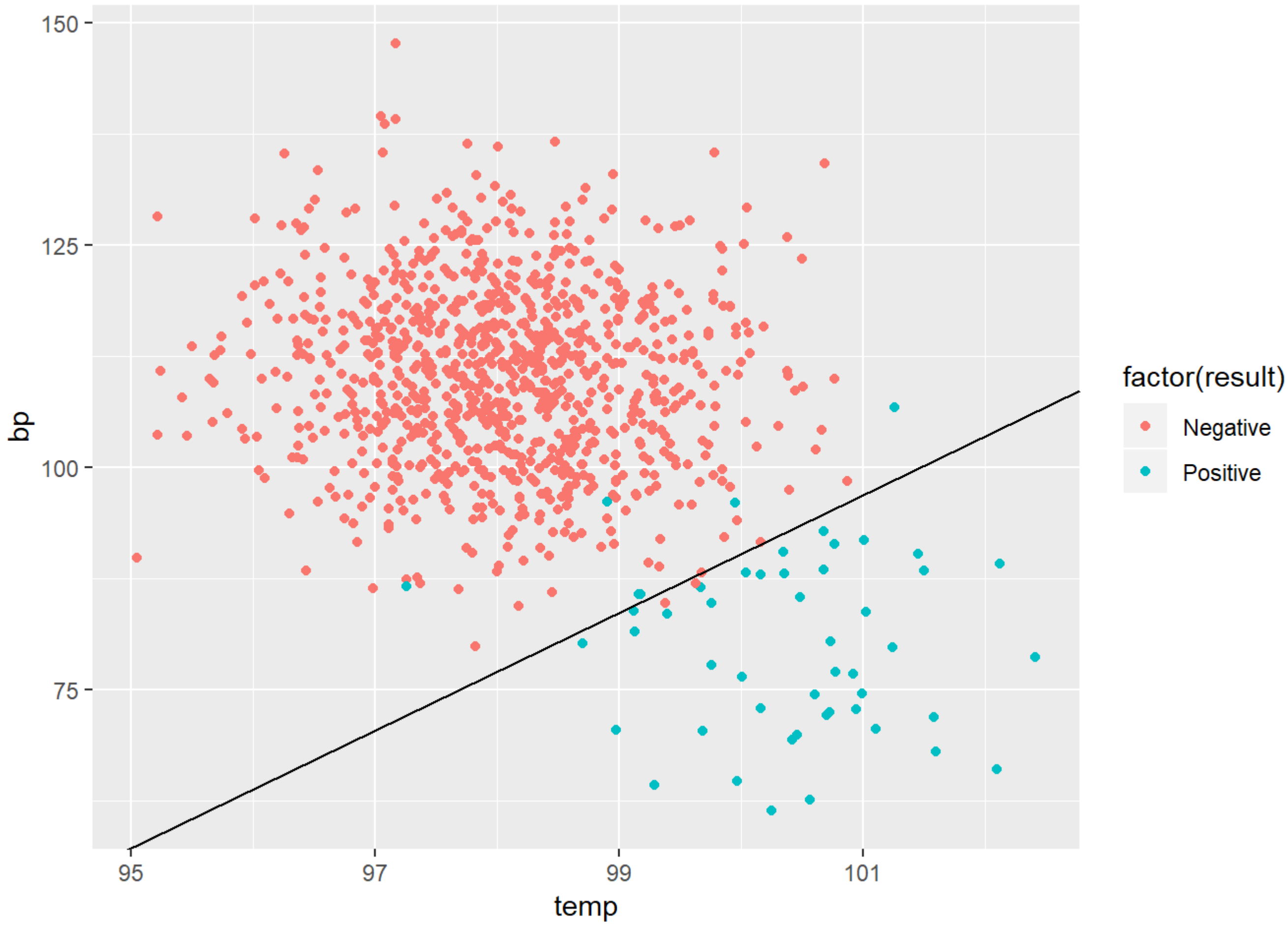
The Logit regression correctly predicts the results 99.2% of the time.

c)

The equation of the line is bp = 6.612235temp - 571.0099.

d)

```
library(ggplot2)
ggplot(sick, aes(temp, bp)) +
  geom_point()+
  geom_point(aes(colour = factor(result)))+
  geom_abline(intercept = -571.0099, slope = 6.612235)
```



Q2 Regularization/Selection

a)

```
setwd("~/GitHub/MMSS_311_2")
widget <- read.csv("widget_data.csv")
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------- tidyverse 1.2.1
--
```

```
## v tibble  2.1.1       v purrr   0.3.2
## v tidyr   0.8.3       v dplyr   0.8.0.1
## v readr   1.3.1       v stringr 1.4.0
## v tibble  2.1.1       v forcats 0.4.0
```

```
## -- Conflicts ---------------------------------------------------------- tidyverse_conflicts()
--
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(broom)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

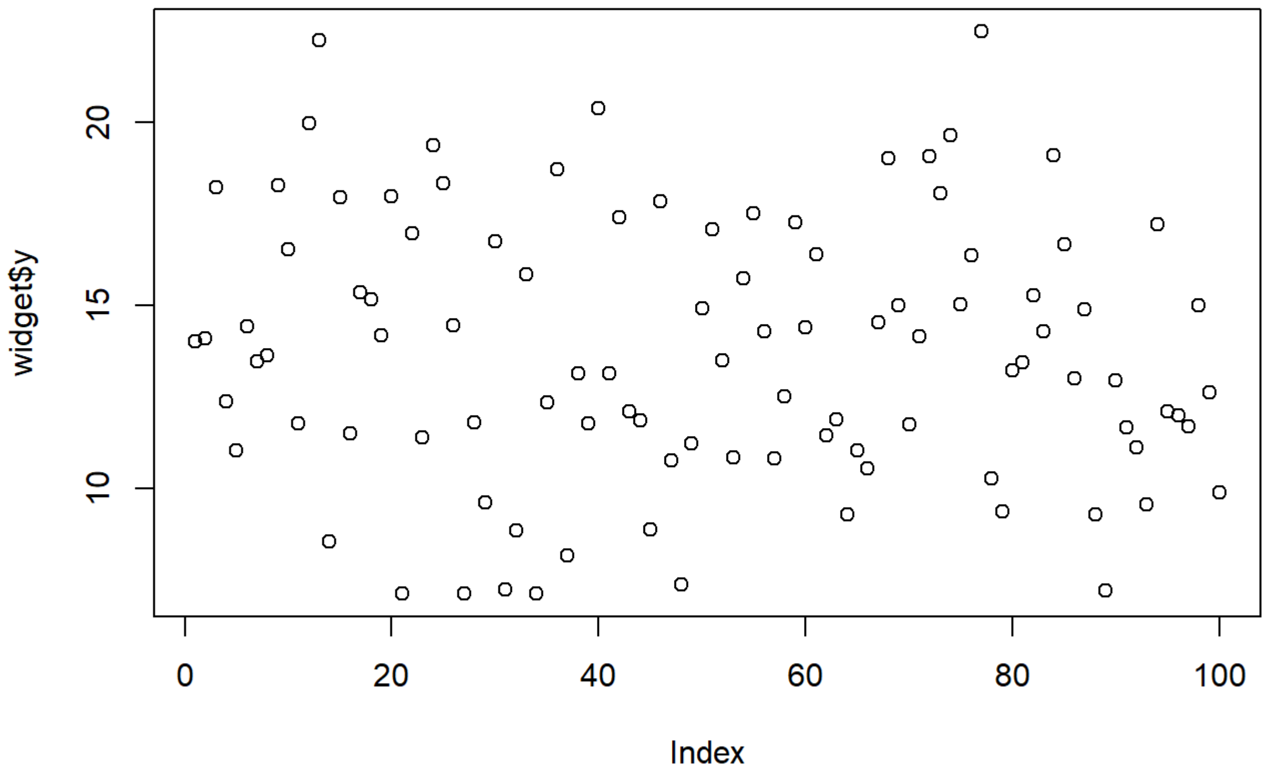
```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-16
```

```r
plot (widget$y)
```



Ridge

b)

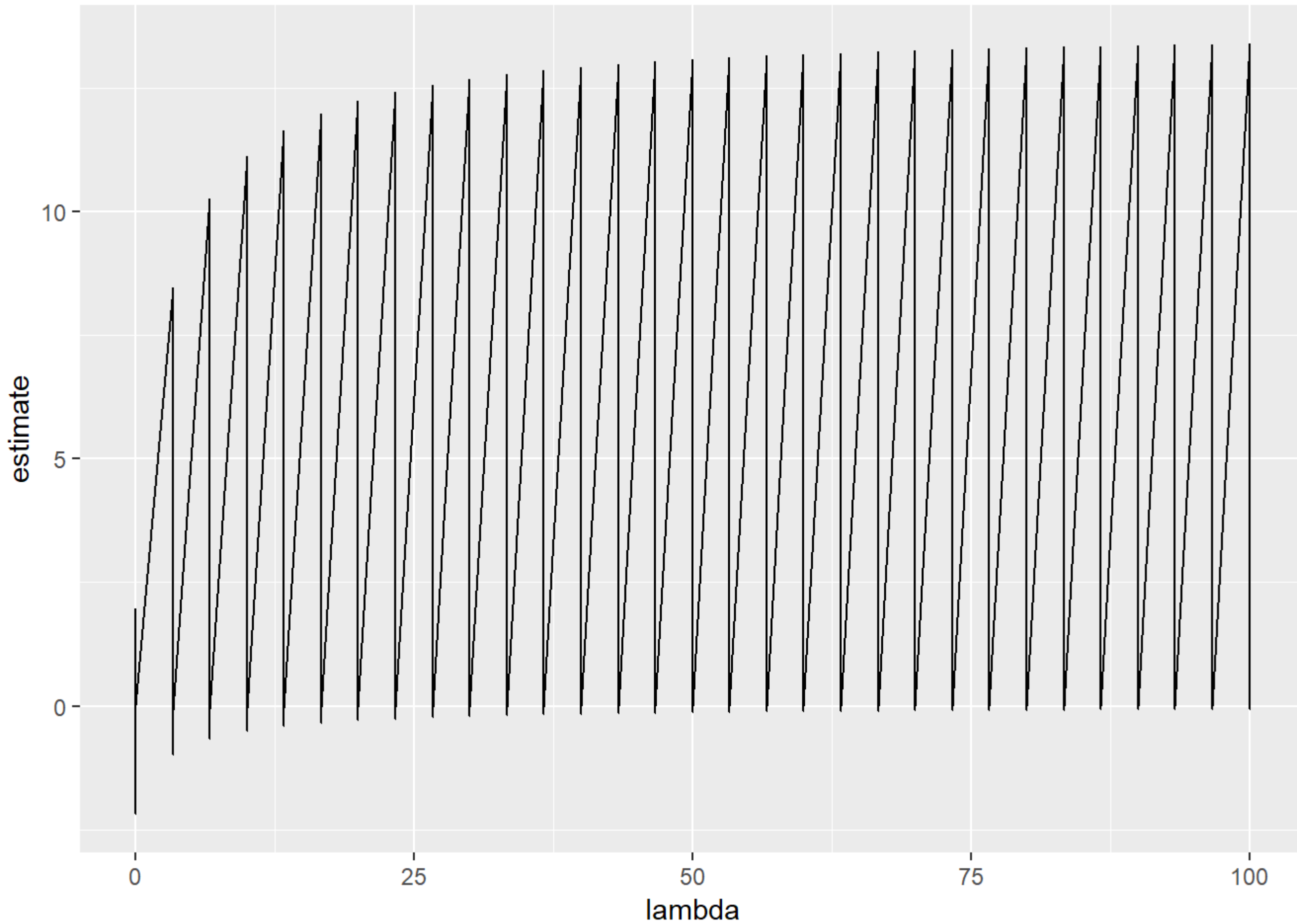
```
x <- model.matrix(y~., widget)[,-1]

grid = seq(1/100, 100, length = 31)
ridge_mod = glmnet(x, widget$y, alpha = 0, lambda = grid)
ridge_mod
```

```
##
## Call:  glmnet(x = x, y = widget$y, alpha = 0, lambda = grid)
##
##       Df    %Dev  Lambda
##  [1,] 30 0.05904 100.000
##  [2,] 30 0.06094  96.670
##  [3,] 30 0.06296  93.330
##  [4,] 30 0.06513  90.000
##  [5,] 30 0.06745  86.670
##  [6,] 30 0.06994  83.340
##  [7,] 30 0.07262  80.000
##  [8,] 30 0.07552  76.670
##  [9,] 30 0.07865  73.340
## [10,] 30 0.08206  70.000
## [11,] 30 0.08577  66.670
## [12,] 30 0.08984  63.340
## [13,] 30 0.09432  60.000
## [14,] 30 0.09926  56.670
## [15,] 30 0.10470  53.340
## [16,] 30 0.11090  50.000
## [17,] 30 0.11780  46.670
## [18,] 30 0.12560  43.340
## [19,] 30 0.13450  40.010
## [20,] 30 0.14480  36.670
## [21,] 30 0.15670  33.340
## [22,] 30 0.17090  30.010
## [23,] 30 0.18780  26.670
## [24,] 30 0.20850  23.340
## [25,] 30 0.23420  20.010
## [26,] 30 0.26720  16.680
## [27,] 30 0.31090  13.340
## [28,] 30 0.37160  10.010
## [29,] 30 0.46130   6.676
## [30,] 30 0.60540   3.343
## [31,] 30 0.81160   0.010
```

c)

```
useful_ridge_mod <- tidy (ridge_mod)
ggplot(useful_ridge_mod, aes(lambda, estimate)) + geom_line()
```



d)

```
cv_ridge_mod <- cv.glmnet(x, widget$y, alpha = 0)$lambda.min
cv_ridge_mod
```

```
## [1] 0.4507848
```

```
lambda_min_ridge_mod = glmnet(x, widget$y, alpha = 0, lambda = cv_ridge_mod)
summary (lambda_min_ridge_mod)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      30     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

The coefficients are printed in the summary when using the value of lambda that minimizes the mean squared error.

Lasso

b)

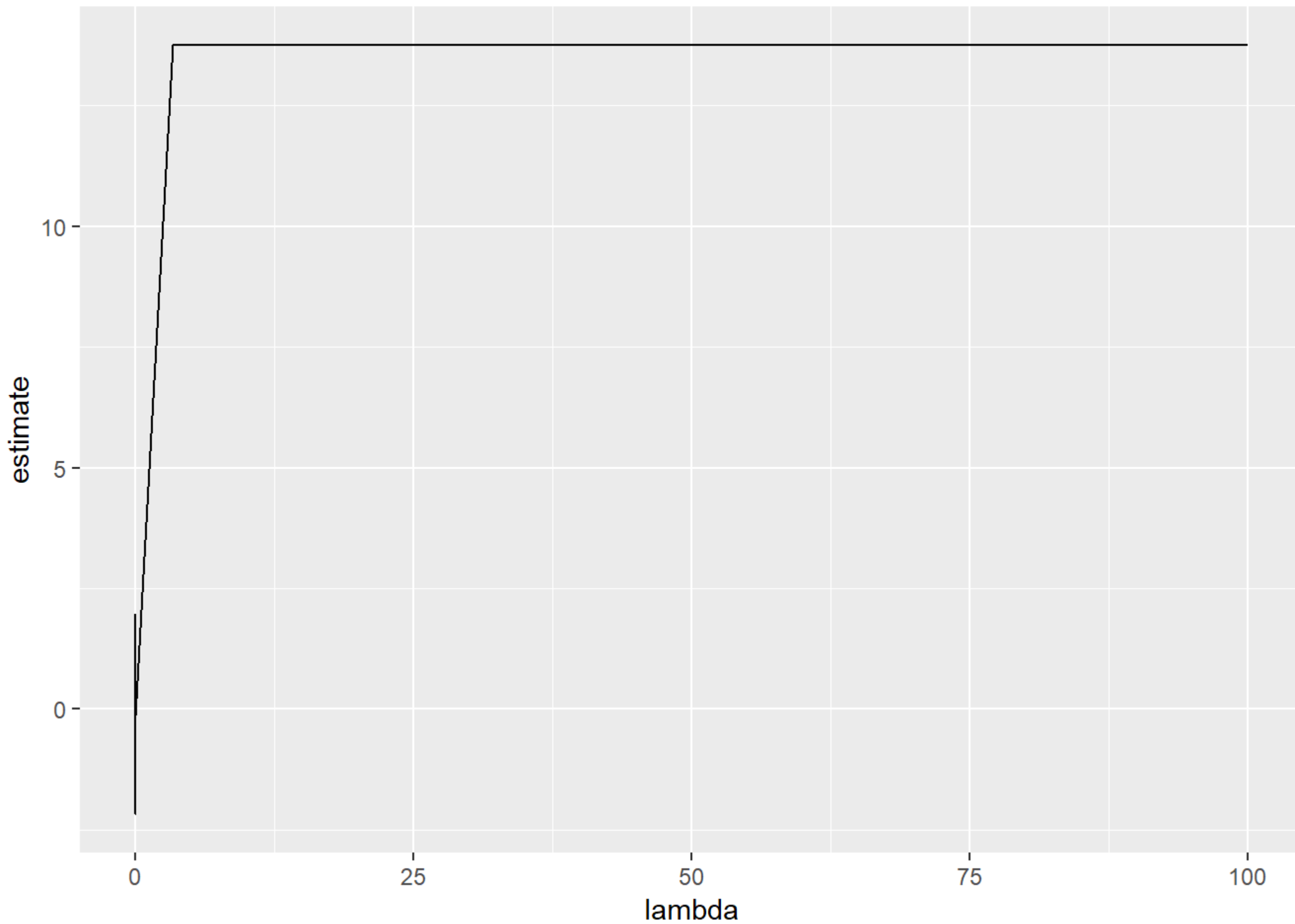
```
x <- model.matrix(y~., widget)[,-1]

grid = seq(1/100, 100, length = 31)
lasso_mod = glmnet(x, widget$y, alpha = 1, lambda = grid)
lasso_mod
```

```
##
## Call:  glmnet(x = x, y = widget$y, alpha = 1, lambda = grid)
##
##       Df   %Dev  Lambda
##  [1,]  0 0.0000 100.000
##  [2,]  0 0.0000  96.670
##  [3,]  0 0.0000  93.330
##  [4,]  0 0.0000  90.000
##  [5,]  0 0.0000  86.670
##  [6,]  0 0.0000  83.340
##  [7,]  0 0.0000  80.000
##  [8,]  0 0.0000  76.670
##  [9,]  0 0.0000  73.340
## [10,]  0 0.0000  70.000
## [11,]  0 0.0000  66.670
## [12,]  0 0.0000  63.340
## [13,]  0 0.0000  60.000
## [14,]  0 0.0000  56.670
## [15,]  0 0.0000  53.340
## [16,]  0 0.0000  50.000
## [17,]  0 0.0000  46.670
## [18,]  0 0.0000  43.340
## [19,]  0 0.0000  40.010
## [20,]  0 0.0000  36.670
## [21,]  0 0.0000  33.340
## [22,]  0 0.0000  30.010
## [23,]  0 0.0000  26.670
## [24,]  0 0.0000  23.340
## [25,]  0 0.0000  20.010
## [26,]  0 0.0000  16.680
## [27,]  0 0.0000  13.340
## [28,]  0 0.0000  10.010
## [29,]  0 0.0000   6.676
## [30,]  0 0.0000   3.343
## [31,] 28 0.8113   0.010
```

c)

```
useful_lasso_mod <- tidy (lasso_mod)
ggplot(useful_lasso_mod, aes(lambda, estimate)) + geom_line()
```



d)

```
cv_lasso_mod <- cv.glmnet(x, widget$y, alpha = 1)$lambda.min
cv_lasso_mod
```

```
## [1] 0.1737112
```

```
lambda_min_lasso_mod = glmnet(x, widget$y, alpha = 1, lambda = cv_lasso_mod)
summary(lambda_min_lasso_mod)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      30     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

The coefficients are printed in the summary when using the value of lambda that minimizes the mean squared error.

f)

As can be seen from the 2 plots, the variation in estimates when using different values of lambda are significantly higher for the ridge regression as compared to the lasso regression. As such, the ridge regression is likely to be less useful than the lasso regression because of this high varition in estimated values depending on lambda.

Q3 Classification

a)

```
pol <- read.csv("pol_data.csv")
library("caret")
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
library("kernlab")
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
set.seed(1)
split=2/3
trainIndex <- createDataPartition(pol$group, p=split, list=FALSE)
train <- pol[ trainIndex,]
test <- pol[-trainIndex,]
```

SVM

b)

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(1)

svm_Linear <- train(group ~., data = train, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)
svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 200 samples
##   3 predictor
##   2 classes: 'Politicalist', 'Socialcrat'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9633333  0.9266667
##
## Tuning parameter 'C' was held constant at a value of 1
```

c)

```
test_pred_svm <- predict(svm_Linear, newdata = test)
print(test_pred_svm)
```

```
##   [1] Socialcrat   Socialcrat   Socialcrat   Politicalist 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   Socialcrat   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] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [56] Politicalist Politicalist Socialcrat   Politicalist Politicalist
##  [61] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [66] Politicalist Politicalist Politicalist Socialcrat   Politicalist
##  [71] Politicalist Politicalist Socialcrat   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] Politicalist Politicalist Politicalist Politicalist Politicalist
## Levels: Politicalist Socialcrat
```

d)

```
confusionMatrix(test_pred_svm, test$group)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Politicalist Socialcrat
##   Politicalist           47          1
##   Socialcrat              3         49
##
##                Accuracy : 0.96
##                  95% CI : (0.9007, 0.989)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.92
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9400
##             Specificity : 0.9800
##          Pos Pred Value : 0.9792
##          Neg Pred Value : 0.9423
##              Prevalence : 0.5000
##          Detection Rate : 0.4700
##    Detection Prevalence : 0.4800
##       Balanced Accuracy : 0.9600
##
##        'Positive' Class : Politicalist
##
```

```
table(test_pred_svm, test$group)
```

```
##
## test_pred_svm  Politicalist Socialcrat
##   Politicalist           47          1
##   Socialcrat              3         49
```

Naive Bayes

b)

```
NBclassifier=naiveBayes(group~., data=train)
print(NBclassifier)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
## Politicalist   Socialcrat
##          0.5          0.5
##
## Conditional probabilities:
##               pol_margin
## Y                   [,1]      [,2]
##   Politicalist 0.6450832 0.1819798
##   Socialcrat   0.3448000 0.2034524
##
##               col_degree
## Y                   [,1]      [,2]
##   Politicalist 0.3121395 0.1708398
##   Socialcrat   0.5850000 0.2246097
##
##               house_income
## Y                  [,1]     [,2]
##   Politicalist 78947.11 9517.112
##   Socialcrat   49657.32 9921.416
```

c)

```
pred_NB <- predict(NBclassifier, test)
print(pred_NB)
```

```
##   [1] Socialcrat   Socialcrat   Socialcrat   Politicalist 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   Socialcrat   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] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [56] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [61] Politicalist Politicalist Politicalist Politicalist Politicalist
##  [66] Politicalist Politicalist Politicalist Socialcrat   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] Politicalist Politicalist Politicalist Politicalist Politicalist
## Levels: Politicalist Socialcrat
```

d)

```
confusionMatrix(pred_NB, test$group)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Politicalist Socialcrat
##   Politicalist           49          1
##   Socialcrat              1         49
##
##                Accuracy : 0.98
##                  95% CI : (0.9296, 0.9976)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.96
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.98
##             Specificity : 0.98
##          Pos Pred Value : 0.98
##          Neg Pred Value : 0.98
##              Prevalence : 0.50
##          Detection Rate : 0.49
##    Detection Prevalence : 0.50
##       Balanced Accuracy : 0.98
##
##        'Positive' Class : Politicalist
##
```

```
table(pred_NB, test$group)
```

```
##
## pred_NB        Politicalist Socialcrat
##   Politicalist           49          1
##   Socialcrat              1         49
```