BIOST 546 - HW3

John Schoof

2/23/2021

Question 1: Lasso Regression

```r
## generate data
  set.seed(1)

  x <- rnorm(30)

  noise <- rnorm(30)
```

```r
## generate response vector
  y <- 3 - 2*x + 3*x^2 + noise
```

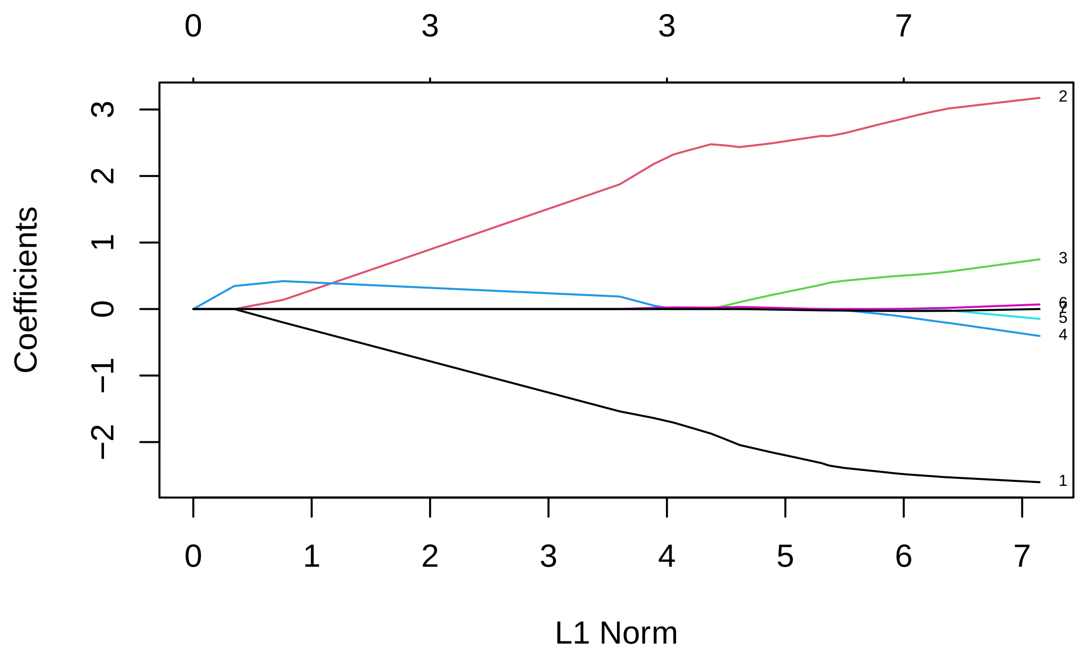
1a-b. The above two code chunks create a set of random data, x and y, with values between zero and one.

```r
## generate matrix of data
  datx <- bind_cols(x, x^2, x^3, x^4, x^5, x^6, x^7)
```

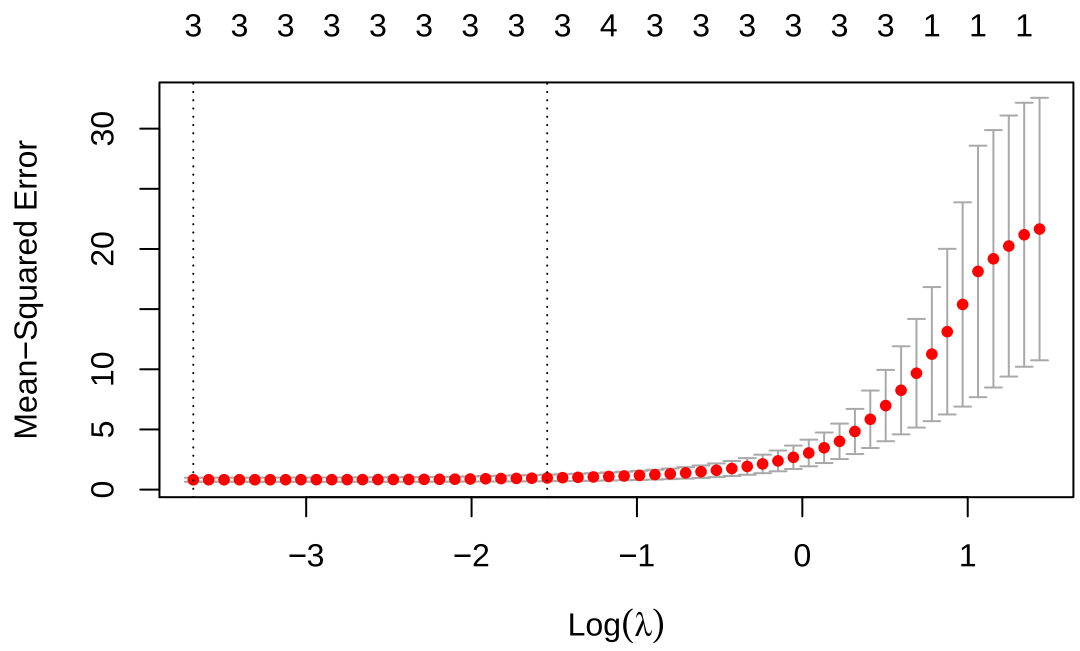
```
## New names:
## * NA -> ...1
## * NA -> ...2
## * NA -> ...3
## * NA -> ...4
## * NA -> ...5
## * ...
```

```r
  colnames(datx) <- c("x", "x2", "x3", "x4", "x5", "x6", "x7")
  datx <- as.matrix(datx)

## create grid
  grid = 10^seq(8,-5,length=100)

## Lass model
  mod1 = glmnet(datx, y, alpha=1, lambda = grid)

## plot
  plot(mod1,
       label = T)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



```
## select tuning parameter
  mod1.cv = cv.glmnet(datx, y, alpha=1)

## plot
  plot(mod1.cv)
```



```
## minimum lambda value
  (bestlam <- mod1.cv$lambda.min)
```

```
## [1] 0.02516
```

```
## fit model with min lambda
  mod1.final <- glmnet(datx, y, alpha=1, lambda = bestlam)
  (mod1.pred <- predict(mod1.final, type = "coefficients", s = bestlam))
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept)  3.396823
## x           -1.876747
## x2           2.470690
## x3           0.003475
## x4           .
## x5           .
## x6           0.022497
## x7           .
```

```
  mod1.pred[mod1.pred!=0]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1]  3.396823 -1.876747  2.470690  0.003475  0.022497
```

1c. The first plot displays the values of the seven coefficients on different levels of lambda. The second plot displays the model MSE on the log(lambda). In the cross validation model, the value of lambda that minimizes the model MSE is 0.0252. I then fit the model on the full dataset using the optimal lambda value. Below is the equation for that model. As one can see, the variables x^3, x^4, x^5,and x^7 fall out of the model.

$$[\mathrm{P}(Y = y)|x, x^2, x^6] = 3.40 - 1.87 * x + 2.47 * x^2 + 0.02 * x^6$$

```
## predictor vector
  x <- rnorm(1000)
  noise <- rnorm(100)

## generate response vector
  y <- 3 - 2*x + 3*x^2 + noise

## generate matrix of data
  datx.d <- bind_cols(x, x^2, x^3, x^4, x^5, x^6, x^7)
```

```
## New names:
## * NA -> ...1
## * NA -> ...2
## * NA -> ...3
## * NA -> ...4
## * NA -> ...5
## * ...
```

```
  colnames(datx.d) <- c("x", "x2", "x3", "x4", "x5", "x6", "x7")
  datx.d <- as.matrix(datx.d)

# Apply Fitted Model
  mod1.d <- predict(mod1.final, newx = datx.d, s = bestlam)

# Calculate the MSE
  lasso.mse <- mean((mod1.d-y)^2)
```

1d. The model MSE on the dataset with 1000 observations is 5.8171.

Question 2: Simple Logistic Regression

```
## load data
  dat2 <- fread("wdbc.data")
  dat2 <- dat2[ , -1]
  table(dat2$V2)
```

```
##
##   B   M
## 357 212
```

```r
  which(is.na(dat2)) ## no missing data
```

```
## integer(0)
```

```r
## kx1 = V3,eep and rename variables
  dat2 <- dat2 %>% dplyr::select(1, 2:31) %>% `colnames<-`(c("diagnosis", paste0("X", 1:30)))
  dat2 <- dat2 %>% mutate(diagnosis = as.factor(diagnosis))
  glimpse(dat2)
```

```
## Rows: 569
## Columns: 31
## $ diagnosis <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, ...
## $ X1        <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450, 18.250, 1...
## $ X2        <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.98, 20.83, 2...
## $ X3        <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, 119.60, 90....
## $ X4        <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, 1040.0, 577...
## $ X5        <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0.12780, 0.0...
## $ X6        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0.17000, 0.1...
## $ X7        <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0.15780, 0.1...
## $ X8        <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0.08089, 0.0...
## $ X9        <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087, 0.1794, 0...
## $ X10       <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0.07613, 0.0...
## $ X11       <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345, 0.4467, 0...
## $ X12       <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902, 0.7732, 1...
## $ X13       <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.180, 3.856, 2...
## $ X14       <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.91, 50.96, ...
## $ X15       <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.011490, 0.00751...
## $ X16       <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.024610, 0.03345...
## $ X17       <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0.03672, 0.0...
## $ X18       <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.018850, 0.01137...
## $ X19       <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0.02165, 0.0...
## $ X20       <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.005115, 0.00508...
## $ X21       <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.88, 17.06, 1...
## $ X22       <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.66, 28.14, 3...
## $ X23       <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40, 153.20, 11...
## $ X24       <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, 1606.0, 897...
## $ X25       <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791, 0.1442, 0...
## $ X26       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249, 0.2576, 0...
## $ X27       <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0.53550, 0.3...
## $ X28       <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0.17410, 0.1...
## $ X29       <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985, 0.3063, 0...
## $ X30       <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0.12440, 0.0...
```

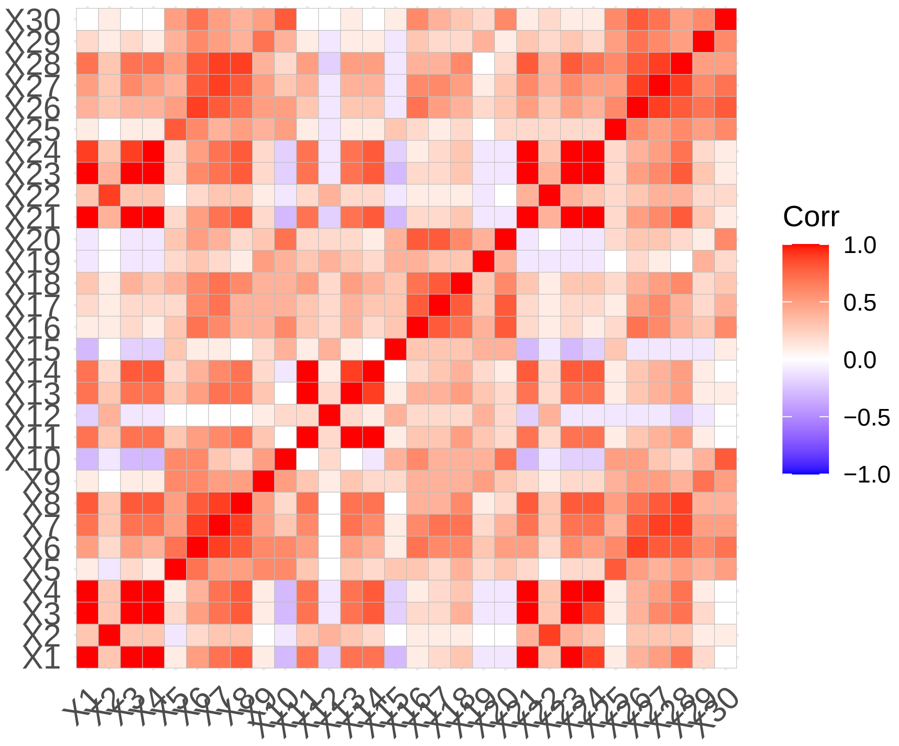
2a. The Breast Cancer Wisconsin (Diagnostic) data has a sample size of 569, 30 predictor variables, and the outcome variable is defined by two classes, benign and malignant. The benign class has 357 observations and the malignant class has 212 observations.

```r
## split sample
  set.seed(2)
  train_id = sample(nrow(dat2), 400)
  train = dat2[train_id,]
  test  = dat2[-train_id,]
```

```r
## Normalize predictors in training set
  train_norm <- apply(train[, -1], 2, scale)

## Normalize predictors in test set
  test_norm <- apply(test[, -1], 2, scale)
```

2b-c. We first split our sample into a training and testing set. We then normalize the predicto variables around a mean of zero. We normalize the predictor variables on the testing and training set separately in the code above so that we avoid overfitting the data. The test set must be a dataset that the model has not seen before.

```r
## correlation matrix for training predictors
  corr <- round(cor(train_norm),1)

## plot correlation matrix
  ggcorrplot(corr)
```



2d. From the above correlation matrix, we can see that X1-4 are highly correlated with X21-24.

```r
  train_norm <- as.data.frame(train_norm)
  train_norm <- cbind(train$diagnosis, train_norm)
  train_norm$diagnosis <- train_norm$`train$diagnosis`
  train_norm <- train_norm %>% dplyr::select(diagnosis, everything())
  train_norm <- train_norm %>% dplyr::select(-`train$diagnosis`)

  test_norm <- as.data.frame(test_norm)
  test_norm <- cbind(test$diagnosis, test_norm)
  test_norm$diagnosis <- test_norm$`test$diagnosis`
  test_norm <- test_norm %>% dplyr::select(diagnosis, everything())
  test_norm <- test_norm %>% dplyr::select(-`test$diagnosis`)

## Fit Simple Logisitic Regression Model
  mod2 <- glm(train$diagnosis ~ ., family = binomial(link = "logit"), data = train_norm)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
  summary(mod2)
```

```
##
## Call:
## glm(formula = train$diagnosis ~ ., family = binomial(link = "logit"),
##     data = train_norm)
##
## Deviance Residuals:
##       Min          1Q      Median          3Q         Max
## -0.000273   0.000000   0.000000   0.000000   0.000320
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    92.46   28281.57    0.00     1.00
## X1           -863.00  462270.50    0.00     1.00
## X2             -3.80   13416.45    0.00     1.00
## X3            989.13  544553.11    0.00     1.00
## X4           -189.77  100538.36    0.00     1.00
## X5            117.78   15574.74    0.01     0.99
## X6           -302.08   41791.18   -0.01     0.99
## X7             -8.09   39076.94    0.00     1.00
## X8            265.44   49259.17    0.01     1.00
## X9           -122.56   22339.35   -0.01     1.00
## X10            96.25   28237.70    0.00     1.00
## X11           559.97   61288.54    0.01     0.99
## X12             3.35    9628.51    0.00     1.00
## X13          -110.48   70063.53    0.00     1.00
## X14          -540.24  107218.22   -0.01     1.00
## X15            31.71   12943.54    0.00     1.00
## X16            73.75   13649.61    0.01     1.00
## X17            52.34   57193.23    0.00     1.00
## X18            90.04   20354.30    0.00     1.00
## X19           -93.14   24992.86    0.00     1.00
## X20          -262.42   65894.49    0.00     1.00
## X21         -1030.88  195279.90   -0.01     1.00
## X22           105.94   15834.64    0.01     0.99
## X23          -502.59  175318.35    0.00     1.00
## X24          2156.15  279421.09    0.01     0.99
## X25           -97.37   20531.90    0.00     1.00
## X26             5.47   30882.30    0.00     1.00
## X27           106.24   52714.25    0.00     1.00
## X28           162.49   25575.21    0.01     0.99
## X29            241.66   25846.59    0.01     0.99
## X30            -39.36   62867.79    0.00     1.00
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.2387e+02  on 399  degrees of freedom
## Residual deviance: 8.5036e-07  on 369  degrees of freedom
## AIC: 62
##
## Number of Fisher Scoring iterations: 25
```

```
## Correlation between variables X1 and X3
  corr <- cor(train_norm$X1, train_norm$X3)
```

2e. The correlation between X1 and X3 is 0.9978. This is almost perfect correlation. Because of this we will not know which of the two variables actually has a biological connection to the outcome.

```
## Training set performances
  glm.prob.train <- predict(mod2, type = "response") # my model's predictions

  ## Binary classification based on probability threshold
  glm.label.train <- rep("B", nrow(train_norm))
  glm.label.train[glm.prob.train > .5] <- "M"

  ## confusion matrix (true positive rate and false positive rate)
  (tt.glm.train <- table(glm.label.train, train_norm$diagnosis))
```

```
##
## glm.label.train   B   M
##               B 255   0
##               M   0 145
```

```
  ## misclassification error (proportion of all 569 that I predicted correctly)
  accuracy.train <- mean(glm.label.train == train_norm$diagnosis)

# Test set performances
  glm.prob.test <- predict(mod2, type = "response", newdata = test_norm)

  ## Binary classification based on probability threshold
  glm.label.test <- rep("B", nrow(test_norm))
  glm.label.test[glm.prob.test > .5] <- "M"
  ## confusion matrix (true positive rate and false positive rate)
  (tt.glm.test <- table(glm.label.test, test_norm$diagnosis))
```

```
##
## glm.label.test  B  M
##              B 99  7
##              M  3 60
```

```
  ## misclassification error (proportion of test set that I predicted correctly)
  (glm.accuracy <- mean(glm.label.test == test_norm$diagnosis))
```

```
## [1] 0.9408
```

2f. Above are the confusion tables for the training and test set. The training set has a perfect prediction accuracy. Given that the test prediction accuracy is lower, 0.9408, it appears as though there is overfitting present.

Question 3

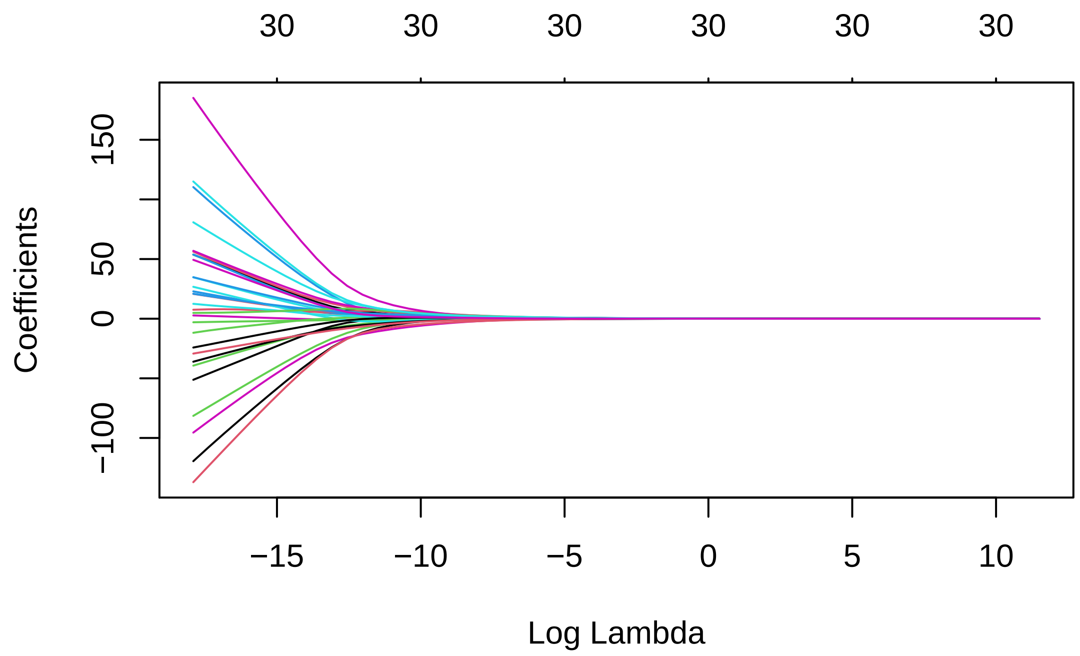
```
# Creating a matrix
  x.train <- model.matrix(diagnosis~. - 1, data = train_norm)
  x.test <- model.matrix(diagnosis~. - 1, data = test_norm)

# Creating a vector y
  y.train <- train_norm$diagnosis
  y.test <- test_norm$diagnosis
```

```
# Create grid
  grid <- 10^seq(5,-18,length =100)

# Ridge regression model
  mod3 <- glmnet(x = x.train, y = y.train, alpha=0 , family = "binomial"(link = "logit"), lambda = grid
```

```
## Warning: Convergence for 57th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
```

3a-b. The above two code chunks construct data matrices with our data because the glmnet function only accepts matrices as inputs. We then create a grid of lambda values and then fit a ridge logistic regression model on the training set.

```
plot(mod3, xvar = "lambda")
```

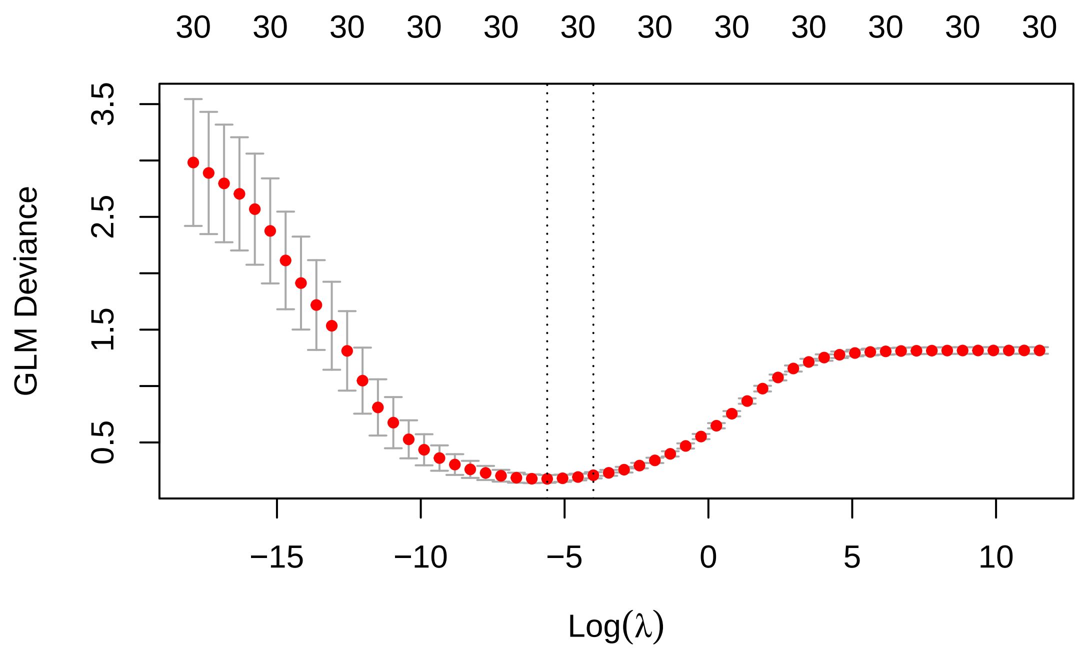


3c. Above is the plot of the 30 predictor coefficients on the log of lambda values. By the time log(lambda) = 0, all coefficients are very close to zero. As can be seen across the top of the plot, the coefficients never equal zero and all 30 predictors stay in the model.

```
# ridge model with cv
  mod3.cv <- cv.glmnet(x = x.train, y = y.train, alpha = 0, family = binomial, lambda = grid, type.meas
```

```
## Warning: Convergence for 57th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning: Only deviance, mse, mae available as type.measure for GLM models;
## deviance used instead

## Warning: Convergence for 57th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning: Convergence for 58th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning: Convergence for 57th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning: Convergence for 58th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning: Convergence for 73th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

```
# lambda that minimizes misclassification error
  best.lam <- mod3.cv$lambda.min

# Plot of misclassification error
  plot(mod3.cv)
```



3d. Above is the plot of test set MSE with error bars on the log of lambda values. The optimal lambda is defined by the left vertical line.

```r
# variables that are different from 0 for best lambda
  (ridge.coef <- predict(mod3.cv, type = "coefficients", s= mod3.cv$lambda.min))
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                    1
## (Intercept) -0.61121
## X1           0.27443
## X2           0.50034
## X3           0.26301
## X4           0.35893
## X5           0.10844
## X6          -0.35471
## X7           0.64297
## X8           0.70122
## X9          -0.06458
## X10         -0.26475
## X11          1.07486
## X12         -0.23200
## X13          0.61030
## X14          0.86823
## X15          0.19936
## X16         -0.61694
## X17          0.10210
## X18          0.39863
## X19         -0.30273
## X20         -0.67843
## X21          0.71825
## X22          0.98491
## X23          0.55546
## X24          0.76080
## X25          0.70414
## X26         -0.03787
## X27          0.93454
## X28          0.84647
## X29          0.88362
## X30          0.12808
```

```r
  length(ridge.coef[ridge.coef !=0] - 1)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] 31
```

3e. As we see above, none of the 30 predictor variables equal zero. This is expected because by definition the ridge regression model does not do feature selection.

```r
  ## Training set performances
  ridge.prob.train <- predict(mod3.cv, type = "response", s = mod3.cv$lambda.min, newx = x.train) # my

  ## Binary classification based on probability threshold
  ridge.label.train <- rep("B", nrow(train_norm))
  ridge.label.train[ridge.prob.train > .5] <- "M"
```

```
## confusion matrix (true positive rate and false positive rate)
(tt.ridge.train <- table(ridge.label.train, train_norm$diagnosis))
```

```
##
## ridge.label.train   B   M
##                 B 254   3
##                 M   1 142
```

```
## misclassification error (proportion of all 569 that I predicted correctly)
accuracy.train <- mean(ridge.label.train == train_norm$diagnosis)

## Test set performances
ridge.prob.test <- predict(mod3.cv, type = "response", , s = mod3.cv$lambda.min, newx = x.test)

## Binary classification based on probability threshold
ridge.label.test <- rep("B", nrow(test_norm))
ridge.label.test[ridge.prob.test > .5] <- "M"
## confusion matrix (true positive rate and false positive rate)
(tt.ridge.test <- table(ridge.label.test, test_norm$diagnosis))
```

```
##
## ridge.label.test   B   M
##                B 101   2
##                M   1  65
```

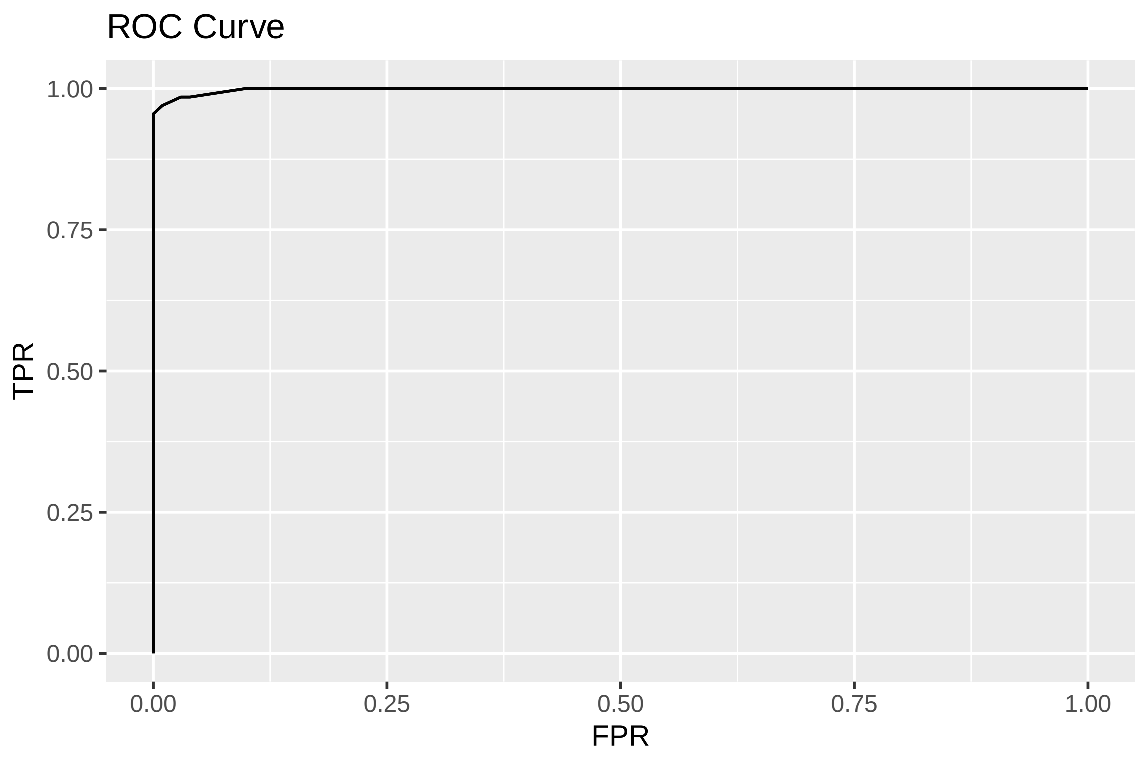
```
## misclassification error (proportion of test set that I predicted correctly)
(ridge.accuracy <- mean(ridge.label.test == test_norm$diagnosis))
```

```
## [1] 0.9822
```

3f. Above are the confusion tables for the training and test set. The test prediction accuracy is 0.9822. This is a very high accuracy and it is similar to the prediction error in the training set, so we don't have evidence for overfitting.

```
## roc curve
  n_segm = 21
  TPR = replicate(n_segm, 0)
  FPR = replicate(n_segm, 0)
  p_th = seq(0,1,length.out = n_segm)

  for (i in 1:n_segm)
  {
    ridge.label.test = rep("B", nrow(test))
    ridge.label.test[ridge.prob.test > p_th[i]] = "M"

    tt.ridge.test = table(ridge.label.test, test_norm$diagnosis)
    TPR[i] = mean(ridge.label.test[test$diagnosis == "M"] == test_norm$diagnosis[test_norm$diagnosis ==
    FPR[i] = mean(ridge.label.test[test$diagnosis == "B"] != test_norm$diagnosis[test_norm$diagnosis ==
  }

# plot(x = FPR, y = TPR, 'l')
```

```
  ggplot() +
    geom_path(aes(x = FPR, y = TPR)) + geom_path(aes(x = FPR, y = TPR))+
    labs(title = "ROC Curve")
```



```
## area under the curve
  auc.vec <- c()
    for (i in 1:20){
      auc.vec[i] <- ((FPR[i]-FPR[(i+1)]) * ((TPR[i]+TPR[(i+1)])/2))
    }
  (auc <- sum(auc.vec))
```

```
## [1] 0.9986
```

3g-h. Above is the ROC curve for the ridge model. As you can see, there is very little space in the upper left hand corner that is not under the curve. This is reflected in the AUC of 0.9986.

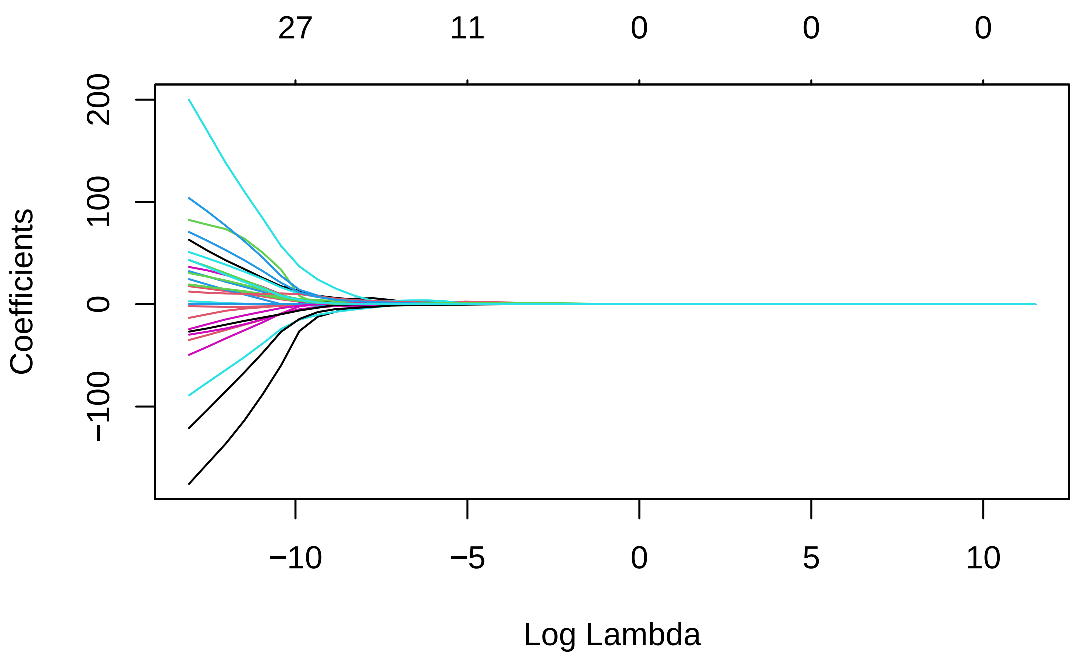
Question 4: Lasso Logistic Regression

```
# Create grid
  grid <- 10^seq(5,-18,length =100)

# Ridge regression model
  mod4 <- glmnet(x = x.train, y = y.train, alpha=1 , family = "binomial"(link = "logit"), lambda = grid
```

```
## Warning: Convergence for 48th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
```

4b. The above code chunk creates a grid of lambda values and then fits a lasso logistic regression model on the training set.

```
plot(mod4, xvar = "lambda")
```



4c. Above is the plot of the 30 predictor coefficients on the log of lambda values. By the time log(lambda) = 0, all coefficients become equal to zero and drop out of the model. The number of coefficients in the model can be seen across the top of the plot.

```
# ridge model with cv
  mod4.cv <- cv.glmnet(x = x.train, y = y.train, alpha = 1, family = binomial, lambda = grid, type.meas
```

```
## Warning: Convergence for 48th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
```

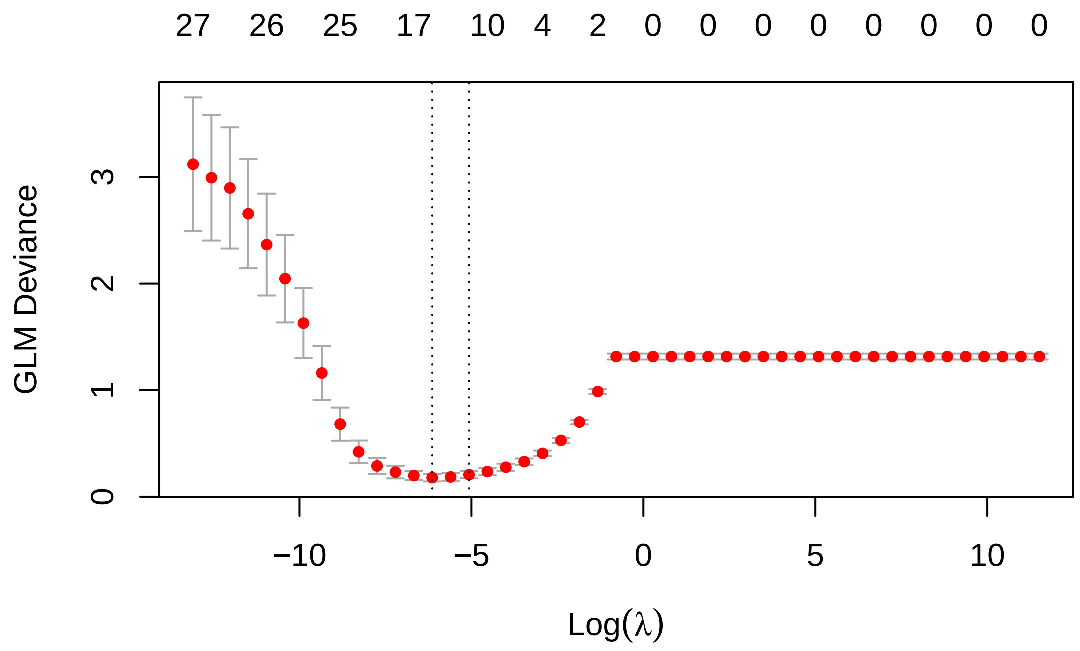
```
## Warning: Only deviance, mse, mae available as type.measure for GLM models;
## deviance used instead
```

```
## Warning: Convergence for 49th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
```

```
## Warning: Convergence for 55th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
```

```
## Warning: Convergence for 56th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned
## Warning: Convergence for 55th lambda value not reached after maxit=1e+05
## iterations; solutions for larger lambdas returned

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

```
# lambda that minimizes misclassification error
  best.lam <- mod4.cv$lambda.min

# Plot of misclassification error
  plot(mod4.cv)
```



4d. Above is the plot of test set MSE with error bars on the log of lambda values. The optimal lambda is defined by the left vertical line. The vertical line on the right is a model with a similar MSE but fewer predictor variables.

```r
# variables that are different from 0 for best lambda
  (log.lasso.coef <- predict(mod4.cv, type = "coefficients", s= mod4.cv$lambda.min))
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                    1
## (Intercept) -0.17016
## X1            .
## X2           0.07997
## X3            .
## X4            .
## X5            .
## X6            .
## X7            .
## X8           0.80063
## X9            .
## X10         -0.07528
## X11          2.77014
## X12         -0.40804
## X13           .
## X14           .
## X15          0.45521
## X16         -0.87327
## X17           .
## X18          0.19219
## X19           .
## X20         -0.59587
## X21           .
## X22          1.54869
## X23           .
## X24          3.83946
## X25          0.42926
## X26           .
## X27          1.60494
## X28          1.19541
## X29          0.58290
## X30           .
```

```r
  length(log.lasso.coef[log.lasso.coef !=0] - 1)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient

## [1] 16
```

4e. As we see above, 16 of the predictor coefficients have nonzero values. This is expected because the lasso regression model does do feature selection. Variables that are not predictive fall out of the model.

```r
  ## Training set performances
  log.lasso.prob.train <- predict(mod4.cv, type = "response", s = mod4.cv$lambda.min, newx = x.train) #

  ## Binary classification based on probability threshold
  log.lasso.label.train <- rep("B", nrow(train_norm))
  log.lasso.label.train[log.lasso.prob.train > .5] <- "M"
  ## confusion matrix (true positive rate and false positive rate)
  (tt.log.lasso.train <- table(log.lasso.label.train, train_norm$diagnosis))
```

```
##
## log.lasso.label.train   B   M
##                     B 254   3
##                     M   1 142
```

```
  ## misclassification error (proportion of all 569 that I predicted correctly)
  accuracy.train <- mean(log.lasso.label.train == train_norm$diagnosis)

  ## Test set performances
  log.lasso.prob.test <- predict(mod4.cv, type = "response", , s = mod4.cv$lambda.min, newx = x.test)

  ## Binary classification based on probability threshold
  log.lasso.label.test <- rep("B", nrow(test_norm))
  log.lasso.label.test[log.lasso.prob.test > .5] <- "M"
  ## confusion matrix (true positive rate and false positive rate)
  (tt.log.lasso.test <- table(log.lasso.label.test, test_norm$diagnosis))
```

```
##
## log.lasso.label.test   B   M
##                    B 101   2
##                    M   1  65
```

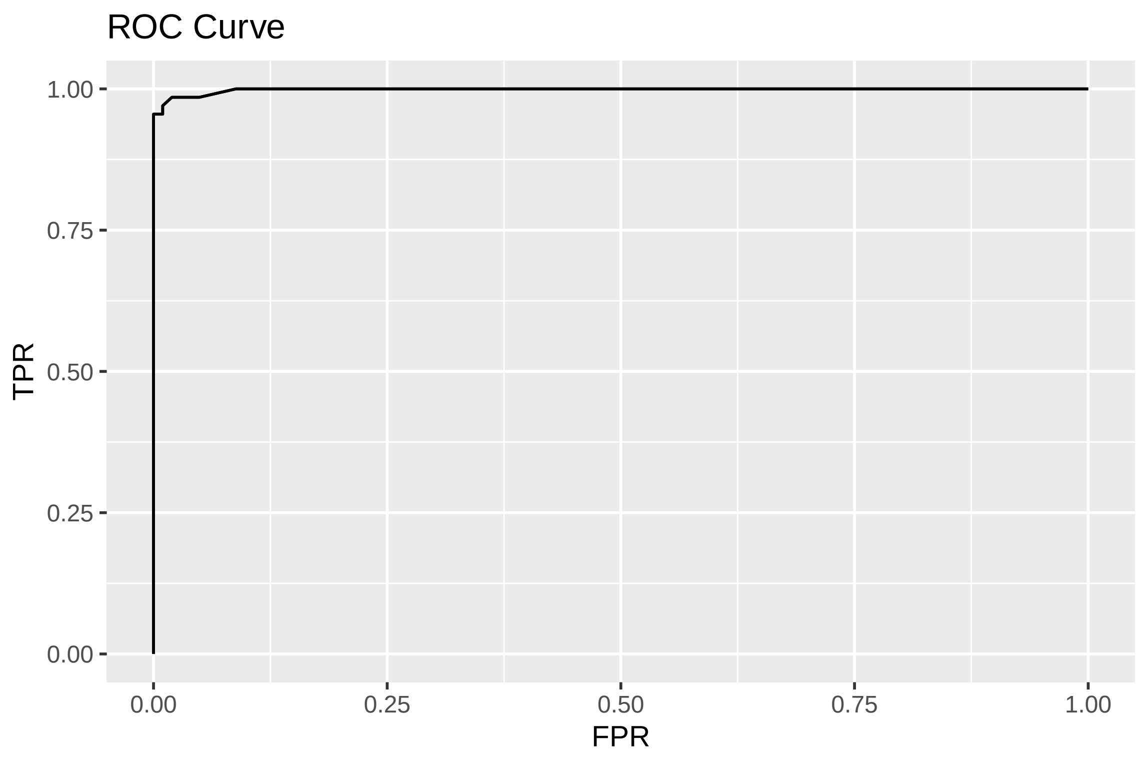
```
  ## misclassification error (proportion of test set that I predicted correctly)
  (log.lasso.accuracy <- mean(log.lasso.label.test == test_norm$diagnosis))
```

```
## [1] 0.9822
```

4f. Above are the confusion tables for the training and test set. The test prediction accuracy is 0.9822. This is a very high accuracy and it is similar to the prediction error in the training set, so we don't have evidence for overfitting.

```
## roc curve
  n_segm = 21
  TPR = replicate(n_segm, 0)
  FPR = replicate(n_segm, 0)
  p_th = seq(0,1,length.out = n_segm)

  for (i in 1:n_segm)
  {
    log.lasso.label.test = rep("B", nrow(test))
    log.lasso.label.test[log.lasso.prob.test > p_th[i]] = "M"

    tt.log.lasso.test = table(log.lasso.label.test, test_norm$diagnosis)
    TPR[i] = mean(log.lasso.label.test[test$diagnosis == "M"] == test_norm$diagnosis[test_norm$diagnosi
    FPR[i] = mean(log.lasso.label.test[test$diagnosis == "B"] != test_norm$diagnosis[test_norm$diagnosi
  }

# plot(x = FPR, y = TPR, 'l')
```

```
ggplot() +
  geom_path(aes(x = FPR, y = TPR)) + geom_path(aes(x = FPR, y = TPR))+
  labs(title = "ROC Curve")
```



```
## area under the curve
  auc.vec <- c()
    for (i in 1:20){
      auc.vec[i] <- ((FPR[i]-FPR[(i+1)]) * ((TPR[i]+TPR[(i+1)])/2))
    }
  (auc <- sum(auc.vec))
```

```
## [1] 0.9986
```

4g-h. Above is the ROC curve for the ridge model. As you can see, there is very little space in the upper left hand corner that is not under the curve. This is reflected in the AUC of 0.9986.

Question 5: Discussion

The below table summarizes the test set prediction error for all three of the models that I fit on the Breast Cancer Wisconsin Data Set. The ridge logistic model and the lasso logistic model have the same prediction accuracy, 0.982, which is higher that the normal logistic regression model.

The ridge and lasso models are penalized models. Although they have the same prediction accuracy, I would choose the lasso model because it only includes 16 predictor variables compared to 30 in the other two. This makes interpretation slightly easier and it can help us drop correlated variables from the model.

Table 1: Model Evaluation

Model	Test Prediction Accuracy
GLM	0.941
Ridge Logistic	0.982
Lasso Logistic	0.982