Recruiter Message: I was tasked to perform normal assumptions, lasso regression, and cross-validation to test the model for problem 1. My partner focused on problem 2 of the final project. These problems were chosen by us and confirmed by our professor to proceed to do analysis on the datasets.

# STAT456 Final Project

Erick Guevara AND Costas Tzoumas

## STAT 456, Spring 2023

## Final Project, May 1 –

## 11, 2023

**Instructions:**

- The final is a group project from May 1 to 11. You can't use the "life happens pass" for this project.
- Please read this project instruction quickly now and then again more carefully later to understand what you need to manage this project. If you have any general questions, such as the description of the problems below, you can email me at least one day before the due date.
- You are allowed to use your textbook as well as other reference books you feel you might need. You should use the statistical software R/Rstudio to perform your analysis and use Rmarkdown to write the report. Please submit two files to Blackboard: PDF of your report and the .Rmd file.
- Every report should have the title (STAT 456 ~ Final Project), authors, and date on the first page.
- All tables and figures should be clearly labeled and numbered. For your pdf, all pages should be numbered, and set the font size of the main body of the report to be 12pt.
- Make sure you proofread your report before submitting it.
- You are under no circumstances allowed to consult with any person other than your professor. You are expected to comply with George Mason's policy on academic integrity. Unauthorized help will result in failing the exam.
- Please do not ask me or TA to debug your codes.

**Good luck!**

```
library(knitr)
knitr::opts_chunk$set(echo = TRUE) tinytex::install_tinytex(force = TRUE)

library(knitr) library(ggplot2)
library(caret)
```

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

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-7
```

```
library(class)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr           1.1.1      v readr
2.1.4 ## v forcats 1.0.0      v stringr
1.5.0 ## v lubridate 1.9.2   v tibble
3.2.1
## v purrr             1.0.1            v tidyr            1.3.0

## -- Conflicts ---------------------------------------- tidyverse_conflicts() -## x tidyr::expand()
masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()                  masks stats::lag()
## x purrr::lift()                  masks caret::lift()
## x tidyr::pack()                masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

## Problem 1

Humans have greatly associated themselves with Songs & Music. It can improve mood, decrease pain and anxiety, and facilitate opportunities for emotional expression. Research suggests that music can benefit our physical and mental health in numerous ways.

Lately, multiple studies have been carried out to understand songs and it's popularity based on certain factors. Such song samples are broken down & their parameters are recorded to tabulate. Predicting the Song Popularity is the main aim.

**Data**

The dataset is provided in song_data.csv. There are 18,835 observations. To judge the predictive performance of your selected model, please hold the last 3000 observations as the test set to estimate the prediction error $\sum_{i=15836}^{18835}(Y_i - \hat{Y}_i)^2$, where $Y_i$ is the popularity of the $i$th song. So only use the first 15,835 observations for estimating/training your model.

**Methods**

You might want to conduct some preliminary data exploration first using data visualization and statistical techniques to describe dataset characterizations, such as size, quantity, and relationship, in order to better understand the nature of the data. Also, you may need to run the following steps multiple times to finally select a model to best predict the Song Popularity.

*Step 1.* Model Building: identify your model (variables) for statistical analysis. [**Hint:** pairwise scatterplot, transformations of the predictors if necessary, stepwise selection with various criteria such as AIC, BIC, *k*-fold cross-validation (CV), and leave-one-out cross-validation (LOOCV), LASSO, ...]

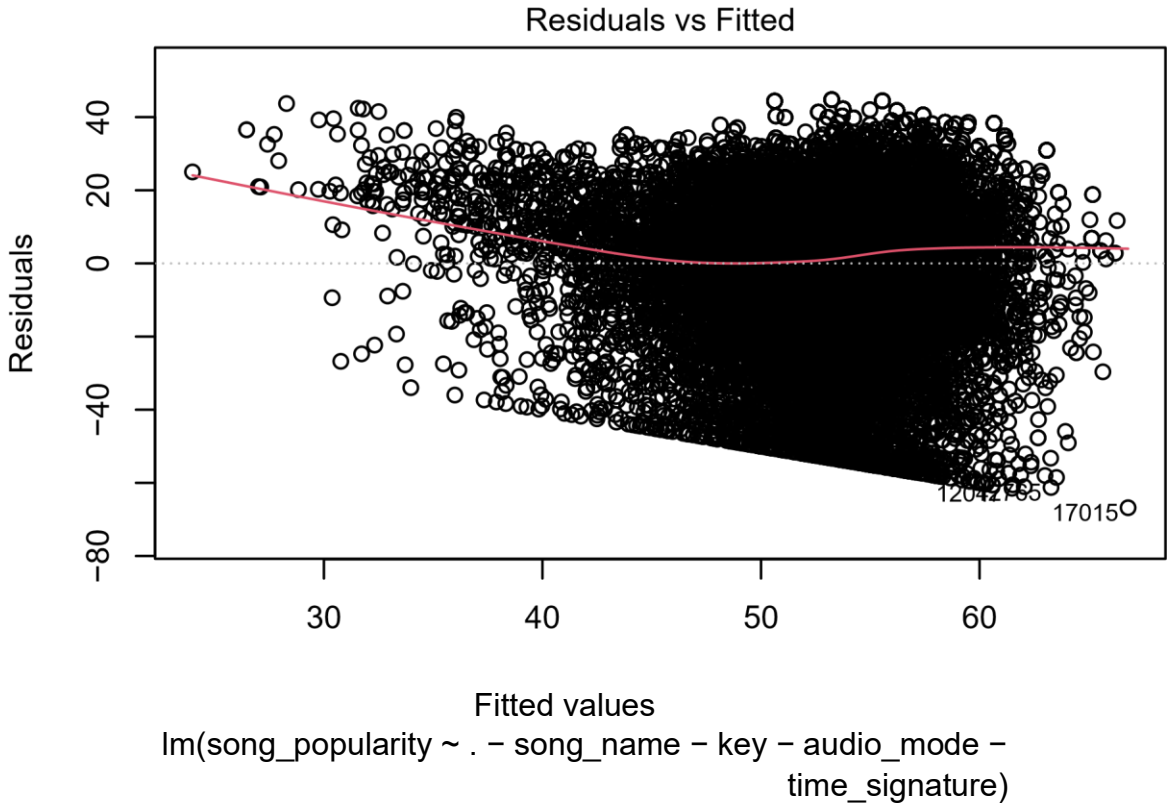
*Step 2.* Diagnostics of your identified model above to determine your final model for association and prediction analysis. [**Hint:** You need to state your model assumptions, and you may consider all diagnostics tools to check the model assumptions. If the assumptions are violated, how will you remedy the analysis? For example, if you find outliers in the data, what to do with the outliers; if the linear assumption does not hold, what to do? You may also consider the interaction among the predictors.]

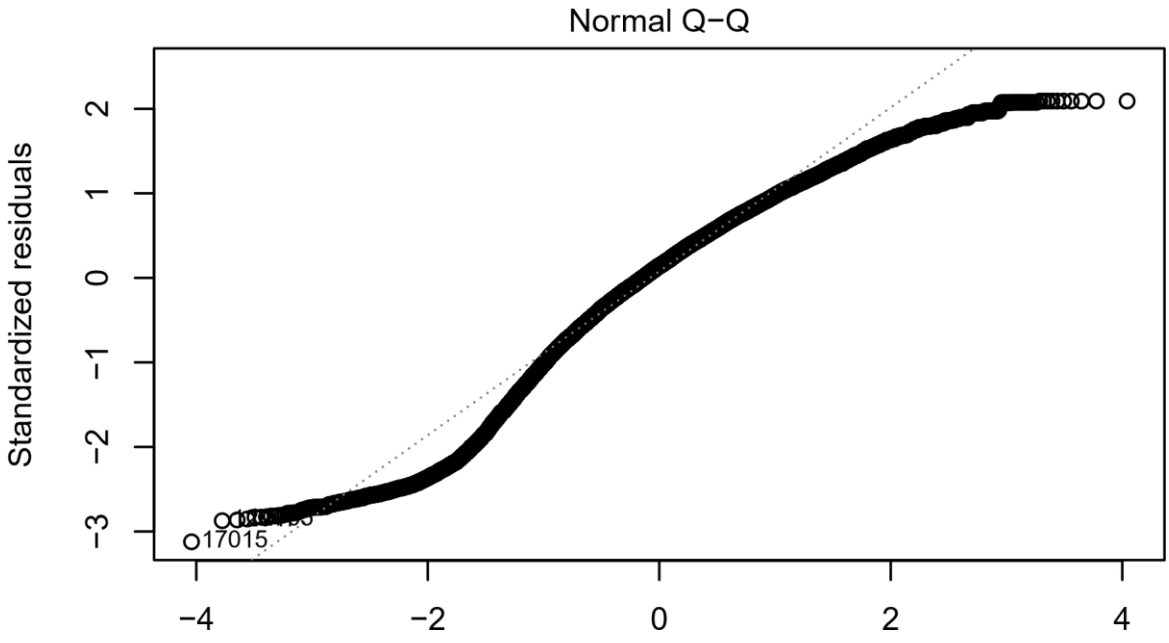
*Step 3.* Conduct statistical inference to claim significant association and determine a prediction model. Use the test set to calculate the prediction mean squared error.

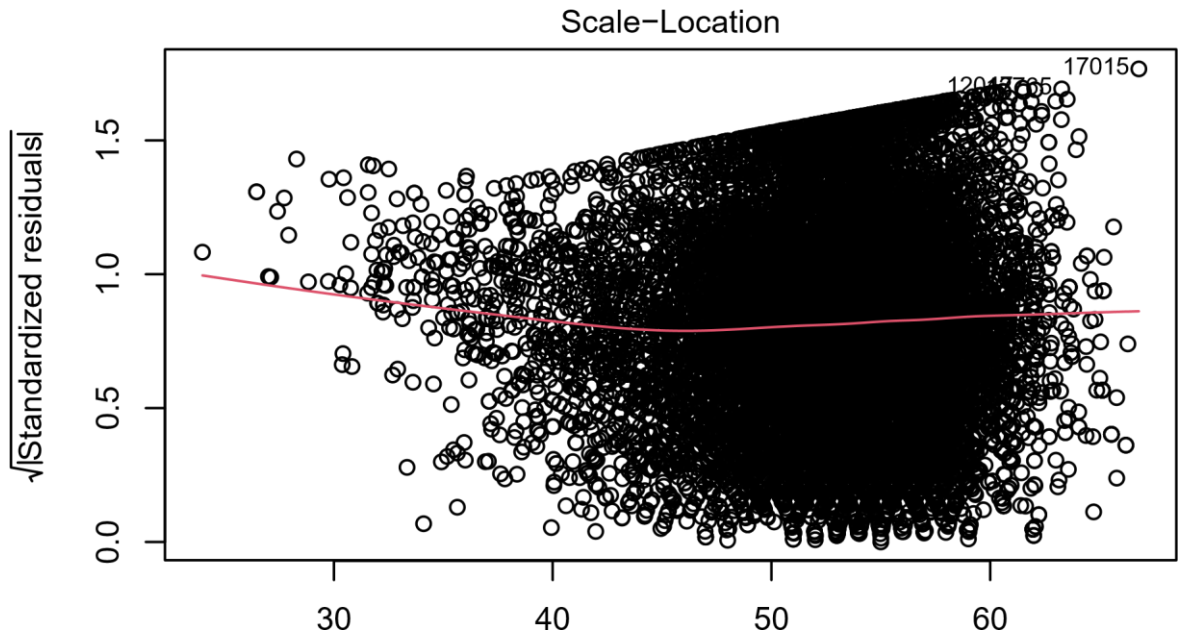
```
library(glmnet)

songs <- read.csv('C:/Users/justd/Desktop/song_data.csv', header=T)

songs_lm <- lm(song_popularity ~. -song_name -key -audio_mode -time_signature, data=songs)

#Checks if the assumptions/conditions are met summary(songs_lm)
```

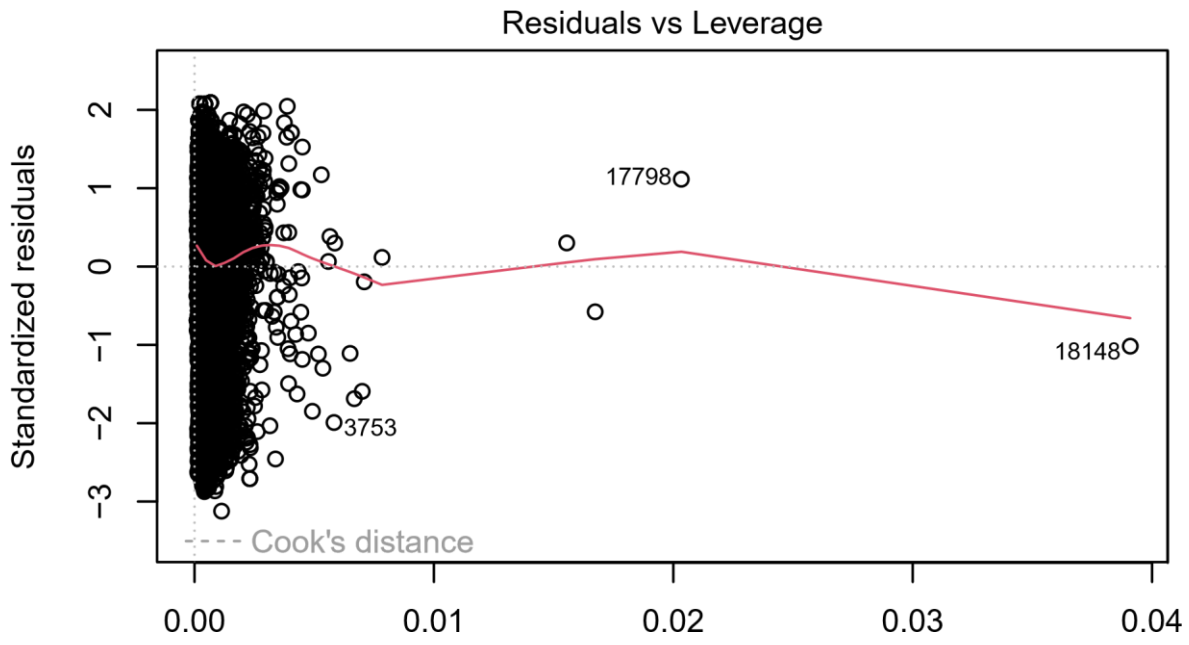
```
##
## Call:
## lm(formula = song_popularity ~ . - song_name - key - audio_mode ##
time_signature, data = songs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -66.796 -12.324   3.029  15.652  44.764
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.755e+01  1.882e+00  35.891  < 2e-16 *** ##
song_duration_ms -5.185e-06  2.660e-06  -1.949   0.0513 .
## acousticness    -4.248e+00  7.585e-01  -5.601  2.16e-08 *** ##
danceability         1.292e+01  1.181e+00  10.943  < 2e-16 *** ##
energy    -1.162e+01  1.404e+00  -8.281  < 2e-16 ***
## instrumentalness -1.037e+01  7.855e-01  -13.199  < 2e-16 ***
##  liveness   -4.429e+00   1.111e+00   -3.986   6.75e-05   ***   ##
loudness 6.989e-01  6.978e-02  10.015  < 2e-16 ***
## speechiness                  -2.294e+00  1.561e+00  -1.469            0.1418
## tempo                          -1.131e-02  5.603e-03  -2.018            0.0436 *
## audio_valence -8.718e+00  7.521e-01  -11.592  < 2e-16 *** ## -
--
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 21.41 on 18824 degrees of freedom
## Multiple R-squared: 0.04566,                    Adjusted R-squared: 0.04516
## F-statistic: 90.07 on 10 and 18824 DF, p-value: < 2.2e-16
plot(songs_lm)
```

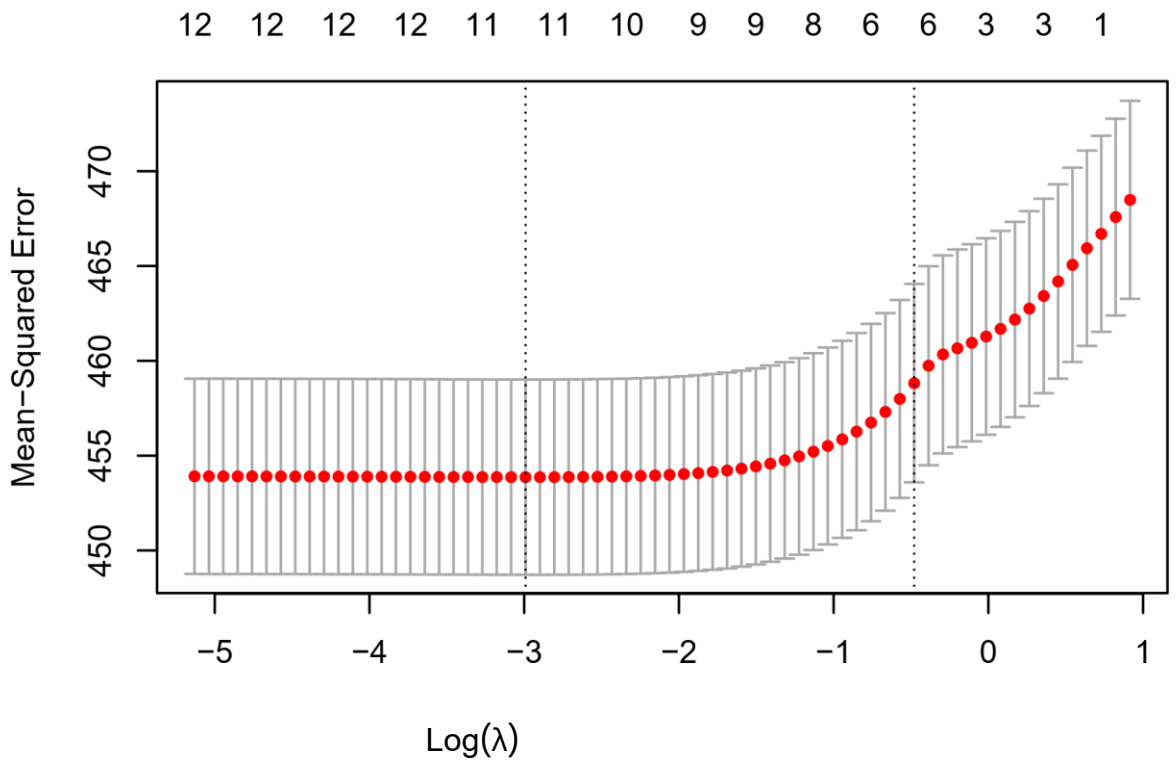



Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
17015
−4
−2
0
2
4
Theoretical Quantiles lm(song_popularity ~ . − song_name −
key − audio_mode − time_signature)


Scale−Location
17015
√|Standardized residuals|
0.0
0.5
1.0
1.5
30
40
50
60
Fitted values lm(song_popularity ~ . − song_name − key −
audio_mode − time_signature)




```
#Lasso Regression train <-
songs[1:15836, ] test<-
songs[15837:18835, ]

x_train <- as.matrix(train[, 3:14]) y_train <-
as.matrix(train[, 2])

x_test <- as.matrix(test[, 3:14]) y_test <-
as.matrix(test[, 2])

las_model <- cv.glmnet(x_train, y_train, alpha = 1) plot(las_model)
```




```
#10k Fold Cross Validation

cbind(lambda=round(las_model$lambda, 1), cvm=signif(las_model$cvm,7),
      cvsd=signif(las_model$cvsd,5))
```

```
##       lambda     cvm    cvsd
## [1,]     2.5 468.4877 5.2213
## [2,]     2.3 467.5799 5.1896
## [3,]     2.1 466.6981 5.1677
## [4,]     1.9 465.9377 5.1501
## [5,]     1.7 465.0615 5.1227
## [6,]     1.6 464.1853 5.1248
## [7,]     1.4 463.4194 5.1276
## [8,]     1.3 462.7553 5.1365
## [9,]     1.2 462.1766 5.1516
## [10,]    1.1 461.6856 5.1673
## [11,]    1.0 461.2815 5.1824
## [12,]    0.9 460.9512 5.1964
## [13,]    0.8 460.6607 5.2118
## [14,]    0.7 460.3364 5.2214
## [15,]    0.7 459.7422 5.2474
## [16,]    0.6 458.8211 5.2321
## [17,]    0.6 457.9901 5.2183
## [18,]    0.5 457.3068 5.2103
## [19,]    0.5 456.7437 5.2044

## [20,]    0.4 456.2677 5.1992
```

```
## [21,]         0.4 455.8586 5.1946
## [22,]         0.4 455.5100 5.1919
## [23,]         0.3 455.2107 5.1888
## [24,]         0.3 454.9583 5.1834
## [25,]         0.3 454.7505 5.1783
## [26,]         0.2 454.5772 5.1744
## [27,]         0.2 454.4330 5.1703
## [28,]         0.2 454.3134 5.1671
## [29,]         0.2 454.2162 5.1645
## [30,]         0.2 454.1418 5.1612
## [31,]         0.2 454.0800 5.1586
## [32,]         0.1 454.0287 5.1569
## [33,]         0.1 453.9870 5.1554
## [34,]         0.1 453.9535 5.1541
## [35,]         0.1 453.9293 5.1534
## [36,]         0.1 453.9103 5.1528
## [37,]         0.1 453.8956 5.1522
## [38,]         0.1 453.8844 5.1517
## [39,]         0.1 453.8763 5.1511
## [40,]         0.1 453.8710 5.1504
## [41,]         0.1 453.8682 5.1498
## [42,]         0.1 453.8669 5.1492
## [43,]         0.1 453.8666 5.1486
## [44,]         0.0 453.8670 5.1482
## [45,]         0.0 453.8683 5.1480
## [46,]         0.0 453.8712 5.1483
## [47,]         0.0 453.8737 5.1484
## [48,]         0.0 453.8766 5.1487
## [49,]         0.0 453.8792 5.1489
## [50,]         0.0 453.8818 5.1490
## [51,]         0.0 453.8851 5.1493
## [52,]         0.0 453.8874 5.1493
## [53,]         0.0 453.8895 5.1494
## [54,]         0.0 453.8922 5.1496
## [55,]         0.0 453.8946 5.1497
## [56,]         0.0 453.8968 5.1498
## [57,]         0.0 453.8988 5.1499
## [58,]         0.0 453.9006 5.1500
## [59,]         0.0 453.9023 5.1502
## [60,]         0.0 453.9040 5.1502
## [61,]         0.0 453.9055 5.1503
## [62,]         0.0 453.9069 5.1504
## [63,]         0.0 453.9082 5.1505
## [64,]         0.0 453.9094 5.1506
## [65,]         0.0 453.9105 5.1506
## [66,]         0.0 453.9112 5.1507
```

```
cvm <-
las_model$cvm
sub1 <-
which.min(cvm)

cvmTest <- cvm[sub1]+
las_model$cvsd[sub1]
```

```
sub2 <- which(cvm <
cvmTest)[1]
```

```
las_model$lambda[sub2]
```

```
## [1] 0.6190959
```

```
las_model$lambda.1se
```

```
## [1] 0.6190959
```

```
#What MSE min and 1SE values start on lambda.min <-
las_model$lambda.min lambda.1se <-
las_model$lambda.1se
```

```
lasso.predBest <- predict(las_model, s=lambda.min, newx=x_test)
lasso.pred1se <- predict(las_model, s=lambda.1se,
  newx=x_test)

pmse.best <- mean((lasso.predBest-y_test)^2) mean((lasso.pred1se-y_test)^2)
```

```
## [1] 518.1881
```

```
cat("This is the predicted mean square error value: ", pmse.best)
```

```
## This is the predicted mean square error value: 506.971
```

```
lasso.coef=predict(las_model,type="coefficients", s=lambda.min)
```

```
lasso.coef2=predict(las_model,type="coefficients",
  s=lambda.1se) lasso.coef
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                    s1 ## (Intercept)
6.494401e+01 ##
song_duration_ms -
1.438106e-06 ##
acousticness               -
6.239774e+00 ##
danceability
5.328451e+00 ## energy
-1.646977e+01
```

```
## instrumentalness -7.388927e+00
## key                          -5.250957e-02
##  liveness  -4.058390e+00
##  loudness  7.241664e-01
## audio_mode    .
## speechiness    -
3.130244e-01 ## tempo
-1.348920e-02
## time_signature               1.372066e+00
```

The R-square of 0.04566 shows a very low correlation score between the response and predictor variables. Residual vs fitted values plot show a spread across the zero line showing a stable variance.To check the normality of the data we use the normal qq plot which appears to be fairly linear. The Scale location plot appears to be randomly scattered across the horizontal band. The Residuals vs leverage plots appear to show a almost no outliers with 2 of them possibly being influential points.

The variables selected from the lasso regression mode is shown above as the lambda minimum choose most of the variables to be relevant to the model. The mean square error prediction plot shows which lambda values are best for least errors and which is the highest. The model output was similar in output as it kept almost all of the variables for it.

## Problem 2.

**Diabetes Data Set.** This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

From the data set in the diabetes.csv file, we can find several variables, some of which are independent (several medical predictor variables) and only one target dependent variable (Outcome).

Columns Description:

- Pregnancies: To express the number of pregnancies;
- Glucose: To express the Glucose level in blood;
- BloodPressure: To express the Blood pressure measurement
- SkinThickness: To express the thickness of the skin
- Insulin: To express the Insulin level in blood
- BMI: To express the Body mass index
- DiabetesPedigreeFunction: To express the Diabetes percentage
- Age: To express the age
- Outcome: To express the final result 1 is Yes and 0 is No.

Let's consider the classification of Outcome using the rest of the variables in the data. In the following, we split the data into two parts: the first half of the data set as the training set and the second half as test data to examine the performance of a classification method.

```
dia <- read.csv('C:/Users/justd/Desktop/diabetes.csv', header=T)
```

(a) Use the training data to build a logistic regression model and predict the probability that an individual in the test data is healthy or has heart disease, given the values of the predictors. Classify the individual as healthy or having heart disease based on whether the predicted probability is above or below 0.5.

```
dia$Outcome <- as.factor(dia$Outcome)

train_diabetes <- dia[1:384,] test_diabetes <-
dia[385:768,]


diabetes_mod <- glm(Outcome ~ ., data = train_diabetes, family = 'binomial') diabetes.probs <-
predict(diabetes_mod, type = "response") diabetes.pred <- rep(0, 768)
diabetes.pred[diabetes.probs > 0.5] <- "unhealthy" diabetes.pred[diabetes.probs < 0.5] <- "Healthy"
```

(b) You can now choose up to 10 different sets of predictors and repeat part (a). For each selection, generate a confusion matrix to evaluate the correct and incorrect classifications of observations in the test set. Define the **test error** as the proportion of misclassified observations within the test set. Report the model that provides the smallest test error.

Model 1

```
library(caret)

diabetes_mod1 <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
                       Insulin, data = train_diabetes, family = 'binomial')
diabetes1.probs <- predict(diabetes_mod1, new_data = test_diabetes, type = "response")
diabetes1.pred <- rep(0, 768)
diabetes1.pred[diabetes1.probs > 0.5] <- "1"
diabetes1.pred[diabetes1.probs < 0.5] <- "0" diabetes1.pred <-
as.factor(diabetes1.pred) confusionMatrix((dia$Outcome),
diabetes1.pred)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction          0      1
##                  0 391 109
##                  1 155 113
##
##                              Accuracy : 0.6562
##                                95% CI : (0.6215, 0.6898)
##             No Information Rate : 0.7109
##              P-Value [Acc > NIR] : 0.999567
##
##                               Kappa : 0.2121
##
## Mcnemar's Test P-Value : 0.005613
##
##         Sensitivity : 0.7161
##         Specificity : 0.5090
##         Pos Pred Value :
0.7820 ##           Neg Pred
Value : 0.4216 ## Prevalence :
0.7109 ##           Detection
Rate : 0.5091
##         Detection
Prevalence : 0.6510 ##
Balanced Accuracy : 0.6126
##
##                  'Positive' Class : 0
##
```

The misclassification rate is 1-0.6562=.3438.

Model 2

```
library(caret)
diabetes_mod2 <- glm(Outcome ~ BMI + DiabetesPedigreeFunction +
                                  Age, data = train_diabetes, family = 'binomial')
diabetes2.probs <- predict(diabetes_mod2, new_data = test_diabetes,
                                          type = "response")
diabetes2.pred <- rep(0, 768) diabetes2.pred[diabetes2.probs > 0.5] <-
"1" diabetes2.pred[diabetes2.probs < 0.5] <- "0" diabetes2.pred <-
as.factor(diabetes2.pred) confusionMatrix((dia$Outcome),
diabetes2.pred)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction           0      1
##                    0 393 107
##                    1 167 101
##
##                                  Accuracy : 0.6432
##                                    95% CI : (0.6082, 0.6772)
##               No Information Rate : 0.7292
##               P-Value [Acc > NIR] : 0.9999999
##
##                                   Kappa : 0.1718
##
## Mcnemar's Test P-Value : 0.0003648
##
##          Sensitivity : 0.7018
##          Specificity : 0.4856
##          Pos Pred Value :
0.7860 ##            Neg Pred
Value : 0.3769 ## Prevalence :
0.7292 ##            Detection
Rate : 0.5117
##          Detection
Prevalence : 0.6510 ##
Balanced Accuracy : 0.5937
##
##                    'Positive' Class : 0
##
```

The misclassification rate for model 2 is 1-0.6432 = 0.3568.

Model 3:

```
library(caret)
diabetes_mod3 <- glm(Outcome ~ Pregnancies + Glucose + BMI, data =
                                train_diabetes, family = 'binomial')
diabetes3.probs <- predict(diabetes_mod3, new_data = test_diabetes, type = "response")
diabetes3.pred <- rep(0, 768)
diabetes3.pred[diabetes3.probs > 0.5] <- "1"
diabetes3.pred[diabetes3.probs < 0.5] <- "0" diabetes3.pred <-
as.factor(diabetes3.pred) confusionMatrix((dia$Outcome),
diabetes3.pred)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction           0      1
##          0
386 114 ##
1 140 128
##
##                                Accuracy : 0.6693
##                                  95% CI : (0.6347, 0.7025)
##          No Information Rate
: 0.6849 ##          P-Value
[Acc > NIR] : 0.8343
##
##                                 Kappa : 0.2554
##
## Mcnemar's Test P-Value : 0.1167
##
##          Sensitivity : 0.7338
##          Specificity : 0.5289
##          Pos Pred Value :
0.7720 ##          Neg Pred
Value : 0.4776 ## Prevalence :
0.6849 ##          Detection
Rate : 0.5026
##          Detection
Prevalence : 0.6510 ##
Balanced Accuracy : 0.6314
##
##                    'Positive' Class : 0
##
```

The misclassification rate for model 3 is 1-0.6693 = 0.3307.

Model 4:

```
library(caret)
diabetes_mod4 <- glm(Outcome ~ BMI + Age + BloodPressure + Insulin, data =
                                train_diabetes, family = 'binomial')
diabetes4.probs <- predict(diabetes_mod4, new_data = test_diabetes, type = "response")
diabetes4.pred <- rep(0, 768)
diabetes4.pred[diabetes4.probs > 0.5] <- "1"
diabetes4.pred[diabetes4.probs < 0.5] <- "0" diabetes4.pred <-
as.factor(diabetes4.pred) confusionMatrix((dia$Outcome),
diabetes4.pred)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction           0      1
##                    0 401 99
##                    1 181 87
##
##                                Accuracy : 0.6354
##                                  95% CI : (0.6003, 0.6695)
##               No Information Rate : 0.7578
##               P-Value [Acc > NIR] : 1
##
##                                 Kappa : 0.1363
##
## Mcnemar's Test P-Value : 1.294e-06
##
##                            Sensitivity : 0.6890
##          Specificity : 0.4677
##          Pos Pred Value :
0.8020 ##            Neg Pred
Value : 0.3246 ## Prevalence :
0.7578 ##            Detection
Rate : 0.5221
##          Detection
Prevalence : 0.6510 ##
Balanced Accuracy : 0.5784
##
##                    'Positive' Class : 0
##
```

The mis-classification rate for model 4 is 1-0.6354 = 0.3646.

The model with the smallest error is model 3 (Outcome ~ Pregnancies + Glucose + BMI).

(c) Based on all continuous predictors, use the KNN method to predict whether the individual is healthy or has heart disease. Try $k = 1$, $k = 3$, $k = 5$ and $k = 10$. For each of the above $k$, report the test error rate. Which $k$ gives the smallest error rate?

For k= 1:

```
set.seed(1)
```

```
kNN_diabetes1 <- knn(train_diabetes[,-9], test_diabetes[,-9],
                                train_diabetes$Outcome, 1)
misclassification_rate1 <- sum(kNN_diabetes1 !=
train_diabetes$Outcome) / length(train_diabetes$Outcome)
misclassification_rate1
```

```
## [1] 0.4427083
```

Misclassification rate of 0.4427.

For k = 3:

```
set.seed(1)
kNN_diabetes3 <- knn(train_diabetes[,-9], test_diabetes[,-9],
                                train_diabetes$Outcome, 3)
misclassification_rate3 <- sum(kNN_diabetes3 !=
train_diabetes$Outcome) / length(train_diabetes$Outcome)
misclassification_rate3
```

```
## [1] 0.4557292
```

Misclassification rate is 0.4557

For k = 5:

```
set.seed(1)
kNN_diabetes5 <- knn(train_diabetes[,-9], test_diabetes[,-9],
                                train_diabetes$Outcome, 5)
misclassification_rate5 <- sum(kNN_diabetes5 !=
train_diabetes$Outcome) / length(train_diabetes$Outcome)
misclassification_rate5
```

```
## [1] 0.4713542
```

Misclassification rate is 0.46875.

For k = 10:

```
set.seed(1)
kNN_diabetes10 <- knn(train_diabetes[,-9], test_diabetes[,-9],
                                 train_diabetes$Outcome, 10)
misclassification_rate10 <- sum(kNN_diabetes10 !=
train_diabetes$Outcome) / length(train_diabetes$Outcom
misclassification_rate10
```

```
## [1] 0.4401042
```

Misclassification rate is 0.4167.

The K that gives the smallest misclassification rate is k = 10.

(d) Experiment with up to 10 different sets of continuous predictors and repeat part (c). Report the KNN method (including selected predictors and the chosen $k$) that yields the smallest test error.

```
train_diabetes1 <- train_diabetes %>% select(c(Pregnancies, Glucose,
                                               BloodPressure, SkinThickness, Insulin,
                                               Outcome))
test_diabetes1 <- test_diabetes %>% select(c(Pregnancies, Glucose,
                                               BloodPressure, SkinThickness, Insulin,
                                               Outcome))
set.seed(1)
kNN_diabetesmod1 <- knn(train_diabetes1[,-6], test_diabetes1[,-6], train_diabetes1$Outcome, 10)
misclassification_ratemod1 <- sum(kNN_diabetesmod1 != train_diabetes1$Outcome) / misclassification_ratemod1


                                                                                length(train_diabetes1$
```

```
## [1] 0.4348958
```

```
train_diabetes2 <- train_diabetes %>% select(c(BMI, DiabetesPedigreeFunction,
                                               Age, Outcome))
test_diabetes2 <- test_diabetes %>% select(c(BMI, DiabetesPedigreeFunction,
                                               Age, Outcome))
set.seed(1)
kNN_diabetesmod2 <- knn(train_diabetes2[,-4], test_diabetes2[,-4], train_diabetes2$Outcome, 10)
misclassification_ratemod2 <- sum(kNN_diabetesmod2 != train_diabetes2$Outcome) / misclassification_ratemod2


                                                                                length(train_diabetes2$
```

```
## [1] 0.4661458
```

```
train_diabetes3 <- train_diabetes %>% select(c(Pregnancies, Glucose, BMI,
                                               Outcome))
test_diabetes3 <- test_diabetes %>% select(c(Pregnancies, Glucose, BMI,
                                               Outcome))
set.seed(1)
kNN_diabetesmod3 <- knn(train_diabetes3[,-4], test_diabetes3[,-4], train_diabetes3$Outcome, 10)
misclassification_ratemod3 <- sum(kNN_diabetesmod3 != train_diabetes3$Outcome) / misclassification_ratemod3


                                                                                length(train_diabetes3$
```

```
## [1] 0.4270833
```

```
train_diabetes4 <- train_diabetes %>% select(c(BMI, Age, BloodPressure, Insulin,
                                               Outcome))
test_diabetes4 <- test_diabetes %>% select(c(BMI, Age, BloodPressure, Insulin,
                                               Outcome))
set.seed(1)
kNN_diabetesmod4 <- knn(train_diabetes4[,-5], test_diabetes4[,-5], train_diabetes4$Outcome, 10)
misclassification_ratemod4 <- sum(kNN_diabetesmod4 != train_diabetes4$Outcome) / misclassification_ratemod4


                                                                    length(train_diabetes4$
```

```
## [1] 0.4557292
```

The KNN method that had the lowest misclassification rate is model 3. The predictors in model 3 are Pregnancies, Glucose, and BMI and the chosen K is 10.