BIOST 546
WINTER QUARTER 2020

Homework # 3
Due Via Online Submission to Canvas: Tues, Feb 25 at 12 PM (Noon)

Instructions:

You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

Please remember — the easier you make it for the TA to find your answer, the easier it will be for him to give you credit for the problem!

On this assignment, some of the problems involve random number generation. Be sure to set a random seed (using the command `set.seed()`) before you begin.

1. For this problem, you will use the `iris` data, which the TA used in the solution key for Q3 of HW1. You will apply K-nearest neighbors to classify the observations.

   (a) Make a plot showing a range of values of K in K-nearest neighbors, ranging from $K = 1$ to $K = n/2$, on the x-axis. On this plot, display five curves:

      i. The training error rate
      ii. The test error rate estimated using the validation set approach
      iii. The test error rate estimated using leave-one-out cross-validation
      iv. The test error rate estimated using 5-fold cross-validation
      v. The test error rate estimated using 10-fold cross-validation

   Be sure to label each curve, as well as the figure axes.

   For this problem, please implement cross-validation yourself — i.e. do not use an `R` function that performs cross-validation for you.

We included an example implementation of k-fold cross-validation for K-NN below. Note that the validation set approach presented in lecture is (the first half of) a 2-fold cross-validation and the LOOCV is just n-fold cross-validation.

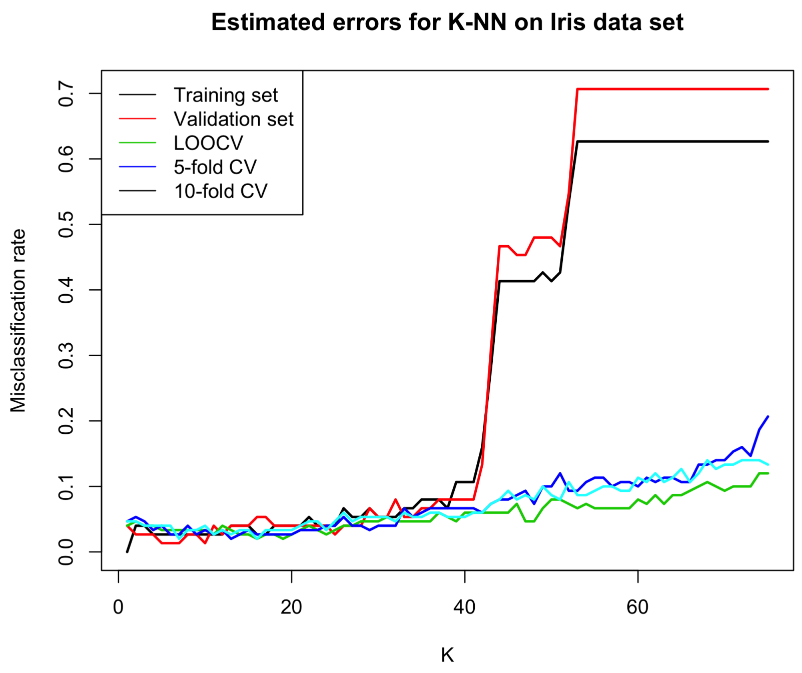

Figure 1: Estimated error rates for K-NN on `Iris` dataset with $K = 1, \cdots, 75$

```
# cross_validation
library(class)
data(iris)

# q1:
knn_k_fold_cv <- function(k,knn_k_seq,X,y,seed=1234,train_error=FALSE){
  "
  a function that implements k-fold cv for knn
  "
  # scramble data first
  set.seed(seed)
  n <- length(y)
  df_error <- matrix(rep(NA, times = length(knn_k_seq)*k), nrow = k)
  scrambled_index <- sample(c(1:n),n,replace=F)
  fold_split <- split(scrambled_index,c(1:n)%%k)
  for (i in seq_along(fold_split)){
    test_obs <- fold_split[[i]]
    X_train <- X[-test_obs,]
    X_test <- X[test_obs,]
    y_train <- y[-test_obs]
    y_test <- y[test_obs]
    for (j in knn_k_seq){
      if (train_error){
        test_prediction <- knn(train=X_train,
                               test=X_train,
                               cl=y_train,
                               k=j)
        test_error <- mean(test_prediction!=y_train)
      }else{
      test_prediction <- knn(train=X_train,
                               test=X_test,
                               cl=y_train,
                               k=j)
      test_error <- mean(test_prediction!=y_test)
      }
      df_error[i,j] <- test_error
    }
  }
  return(df_error)
}

k_seq <- c(1:75)

# train error
train_error_q1 <- knn_k_fold_cv(k=2, k_seq,
X=iris[,-5],y=iris[,5], train_error = TRUE)
plot_train_error_q1 <- train_error_q1[1,]
# validation set
validation_q1 <- knn_k_fold_cv(k=2, k_seq,
X=iris[,-5],y=iris[,5])
plot_validation_q1 <- validation_q1[1,]
# cv
cv_5_q1 <- knn_k_fold_cv(k=5, k_seq,
X=iris[,-5],y=iris[,5])
plot_cv_5_q1 <- apply(cv_5_q1,2,mean)
cv_10_q1 <- knn_k_fold_cv(k=10, k_seq,
X=iris[,-5],y=iris[,5])
plot_cv_10_q1 <- apply(cv_10_q1,2,mean)
# loocv
loocv_q1 <- knn_k_fold_cv(k=length(iris[,5]), k_seq,
X=iris[,-5],y=iris[,5])
plot_loocv_q1 <- apply(loocv_q1,2,mean)
```

```
plot_q1 <- cbind(plot_train_error_q1,plot_validation_q1,
                 plot_loocv_q1,plot_cv_5_q1,plot_cv_10_q1)
matplot(plot_q1, type = c("l"),lty=1,lwd=2,col = 1:5,
        main = 'Estimated errors for K-NN on Iris data set',
        ylab = 'Misclassification rate',
        xlab = 'K') #plot
legend("topleft", legend = c('Training set','Validation set',
                             'LOOCV','5-fold CV','10-fold CV'),
```

(b) Comment on the plot obtained in (a). Which value(s) of K results in the lowest estimated test error?

We note that 5-fold CV, 10-fold CV, and LOOCV provide similar misclassification rate estimates. As remarked in the lecture, LOOCV tends to underestimate the error rate and indeed the green curve has the lowest estimated errors.

Validation set and training set error estimates behave as expected for small K, and the jump around $K = 40$ is due to the fact that we only have $\frac{150}{2} = 75$ data points in those two sets so $K \geq 40$ would lead to very biased estimates, whereas in 5-fold cross-validation, we have $150 \times \frac{4}{5} = 120$ data points for training so using $K \geq 40$ would not be as biased.

Note that in general, we would NOT expect the training set to be this similar to the validation set error.

Estimation methods	Best K(s)
Validation set	5, 6, 7, 10
LOOCV	16, 19
5-fold CV	13
10-fold CV	7, 16

Table 1: Values of K that minimizes the test error on `Iris` data set

Given the relatively small size of our data set, multiple K's give rise to the same estimated test error in some cases. Overall values of K between 5-15 lead to good test set performance.

2. On Q4 of HW2, you fit a bunch of models to the `Auto` dataset from the `ISLR` library.

   Now, use cross-validation to estimate the test errors for several linear models on this dataset (this can be the same set of models you considered on HW2, or a different set of models if you prefer). Also, compute the training error for each model.

   Make a table listing the models that you considered, as well as the training error and estimated test error for each model. Which of the models that you considered had the lowest estimated test error?

For this problem, you can implement cross-validation yourself, or you can use an R function that performs cross-validation for you.

For Q4 in HW2, we investigated which functional form of horsepower best predicts the outcome mpg. In this question we will use 5-fold cross-validation to answer this question by comparing four different forms: linear, quadratic, cubic, and log.

```
lm_k_fold_cv <- function(k,lm_formula,X,y,seed = 1234){
  "
  a function that implements k-fold cv for lm
  "
  set.seed(seed)
  n <- length(y)
  cv_train_error <- rep(NA, times = k)
  cv_test_error <- rep(NA, times = k)
  scrambled_index <- sample(c(1:n),n,replace=F)
  fold_split <- split(scrambled_index,c(1:n)%%k)
  for (i in seq_along(fold_split)){
    test_obs <- fold_split[[i]]
    X_train <- X[-test_obs,]
    X_test <- X[test_obs,]
    y_train <- y[-test_obs]
    y_test <- y[test_obs]
    lm_train <- lm(lm_formula, data = data.frame(X_train, y=y_train))
    train_error <- mean((lm_train$residuals)^2)
    test_error <- mean((y_test-predict(lm_train,
                                   newdata= data.frame(X_test)))^2)
    cv_test_error[i] <- test_error
    cv_train_error[i] <- train_error
  }
  result_error <- c(mean(cv_train_error),mean(cv_test_error))
  names(result_error) <- c('train_error','test_error')
  return(result_error)
}

library(ISLR)
formula_1 <- y~horsepower
formula_2 <- y~horsepower+I(horsepower^2)
formula_3 <- y~horsepower+I(horsepower^3)
formula_4 <- y~horsepower+I(log(horsepower))

cv_5_error_m1 <- lm_k_fold_cv(5,formula_1,Auto,Auto$mpg)
cv_5_error_m2 <- lm_k_fold_cv(5,formula_2,Auto,Auto$mpg)
cv_5_error_m3 <- lm_k_fold_cv(5,formula_3,Auto,Auto$mpg)
```

```
cv_5_error_m4 <- lm_k_fold_cv(5,formula_4,Auto,Auto$mpg)
```

Model	Training error	Test error
Linear term	23.9	24.1
Quadratic term	19.0	19.1
Cubic term	19.2	19.3
Log term	19.3	19.5

Table 2: Estimated training and testing errors on `Auto` data set

As many of you pointed out in the last homework, adding a quadratic term imporves the cross-validated MSE by a significant amount. Overall, the qaudratic model ($\texttt{mpg} = \beta_0 + \beta_1 \cdot \texttt{horsepower} + \beta_2 \cdot \texttt{horsepower}^2$) has the lowest test error estimated using 5-fold cross-validation.

3. In this problem, we'll see a (very!!) simple simulated example where a least squares linear model is "too flexible".

   (a) First, generate some data with $n = 100$ and $p = 10,000$ features, and a quantitative response, using the following `R` commands:

   ```
   y <- rnorm(100)
   x <- matrix(rnorm(10000*100), ncol=10000)
   ```

   Write out an expression for the model corresponding to this data generation procedure. For instance, it might look something like

   $$Y = 2X_1 + 3X_2 + \epsilon, \qquad \epsilon \sim N(0, 1).$$

   $Y = f(x) = \epsilon, \epsilon \sim \mathcal{N}(0,1)$. **We note that** $\mathbb{E}[Y] = 0, Var(Y) = 1$

   (b) What is the value of the irreducible error?
   Recall that by definition irreducible error is $Var(\epsilon) = 1$

   (c) Consider a very simple model-fitting procedure that just predicts 0 for every observation. That is, $\hat{f}(x) = 0$ for all x.

      i. What is the bias of this procedure?
      $Bias(\hat{f}(x)) = \mathbb{E}[\hat{f}(x) - f(x)] = \mathbb{E}[0 - Y] = 0$

      ii. What is the variance of this procedure?
      $Var(\hat{f}(x)) = 0$ **since** $\hat{f}(x) = 0$ **is a constant**

      iii. What is the expected prediction error of this procedure?
      $\mathbb{E}[(\hat{f}(x) - Y)^2] = Bias(\hat{f}(x))^2 + Var(\hat{f}(x)) + Var(\epsilon) = 1$

      iv. Use the validation set approach to estimate the test error of this procedure. What answer do you get?
      We used 50% data for training and 50% for validation; the estimated test error (MSE) is about 1.001

```
set.seed(1999)
train_ratio <- 0.5
y <- rnorm(100)
X <- matrix(rnorm(10000*100), ncol=10000)
train_split <- sample(100, replace = F)[1:floor(100*train_ratio)]
train_y <- y[train_split]
train_X <- X[train_split,]
test_y <- y[-train_split]
test_X <- X[-train_split,]
# estimate testing error using constant prediction
mse_iii <- mean((test_y-0)^2)
```

v. Comment on your answers to (iii) and (iv). Do your answers agree with each other? Explain.

1.001 is pretty close to 1 especially when we only have 50% data for testing! In general your answer might have some variations but it should be close to 1.

(d) Now use the validation set approach to estimate the test error of a least squares linear model using $X_1, \ldots, X_{10{,}000}$ to predict Y. What is the estimated test error?

Hint: If you fit a least squares linear model to predict Y using $X_1, \ldots, X_p$ where $p \geq n$, then only the first $n-1$ coefficients will be assigned values. The rest will show up as `NA` because those coefficients aren't needed to obtain a perfect (i.e. zero) residual sum of squares on the training data. You can see all of the coefficient values by applying the `coef()` command to the output of the linear model.

Our estimated test error on a validation set is about 5.25 which is much larger than 1!

```
lm_fit_iv <- lm(train_y~., data = data.frame(train_X))
y_hat_iv <- predict(lm_fit_iv, newdata = data.frame(test_X))
mse_iv <- mean((test_y-y_hat_iv)^2)
```

(e) Comment on your answers to (c) and (d). Which of the two procedures has a smaller estimated test error? higher bias? higher variance? In answering this question, be sure to think carefully about how the data were generated.

We first note that the estimated test error (MSE) is much larger when we used the 10,000 features for prediction. In terms of bias-variance tradeoff, we know that $\hat{f}(x) = 0$ generally has much higher bias and much lower variance compared to $\hat{f}(x) = \sum_i x_i\hat{\beta}_i$. However,in this case since the former has 0 bias it really is a low bias model as well; the inflated test error in (d) is due to the high variance.

4. In lecture on February 5, 2020, we discussed "Option 1" and "Option 2", within the context of "pitfalls of cross-validation". If you missed that lecture, then please familiarize yourself with the lecture notes (posted on `Canvas`) before you continue.

   Here, we are going to continue to work with the simulated data from the previous problem, in order to illustrate the problem with Option 1.

   (a) Calculate the correlation between each feature and the response. Make a histogram of these correlations. What are the values of the 10 largest absolute correlations?

   We first generate the same data as the question above and compute the correlation between X and y.

```
set.seed(12345)
train_ratio <- 0.5
y <- rnorm(100)
X <- matrix(rnorm(10000*100), ncol=10000)
train_split <- sample(100, replace = F)[1:floor(100*train_ratio)]
train_y <- y[train_split]
train_X <- X[train_split,]
test_y <- y[-train_split]
test_X <- X[-train_split,]

empirical_cor <- cor(X,y)
hist(empirical_cor,
 main = 'Empirical correlations between feature and response',
     xlab = 'Correlation')
q <- 10
top_10_cor <- sort(abs(empirical_cor),decreasing = T)[1:q]
```

   The 10 largest correlations in absolute value are 0.369, 0.348, 0.341, 0.338, 0.332, 0.321, 0.321, 0.320, 0.317, and 0.316

   (b) Now try out Option 1 with $q = 10$ (to keep things simple, you can use the "validation set" version of Option 1, on page 1 of the 2/5/2020 lecture notes). What is the estimated test error?

```
selected_feature <- which(abs(empirical_cor)>=min(top_10_cor))
lm_fit_option_1 <- lm(train_y~.,
data = data.frame(train_X[,selected_feature]))
y_hat_option_1 <- predict(lm_fit_option_1,
                   newdata = data.frame(test_X[,selected_feature]))}
test_error_option_1 <-  mean((test_y-y_hat_option_1)^2)
```

   Recall that option 1 selects the features for a smaller linear model on the ENTIRE data set and estimate the test error on the validation set. The estimated MSE is 0.82.

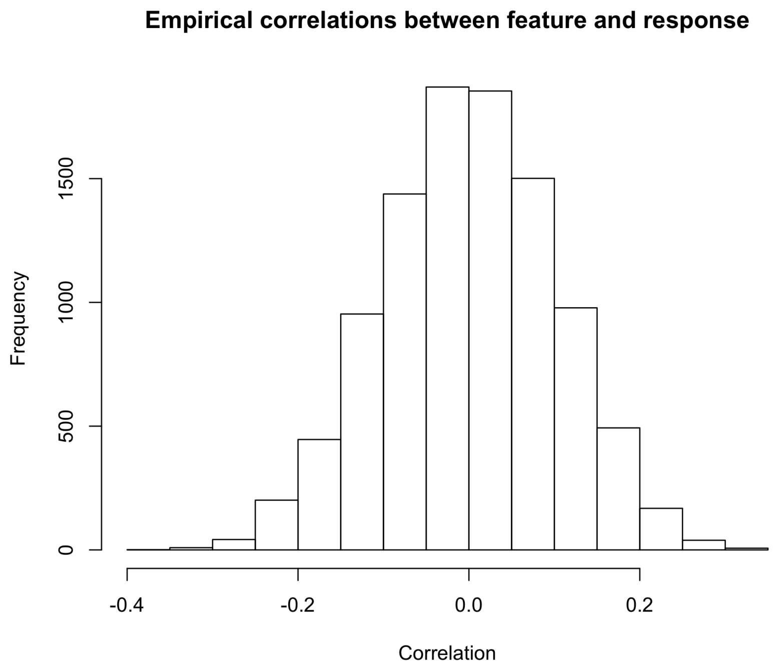

Figure 2: Plotting the correlation between feature x_i and y for $i = 1, 2, \cdots, 10,000$

(c) Now try out Option 2 with $q = 10$ (to keep things simple, you can use the "validation set" version of Option 2, on page 1 of the 2/5/2020 lecture notes). What is the estimated test error?

```
selected_feature_option_2 <-  which(rank(abs(cor(train_X,train_y)))<=q)
lm_fit_option_2 <-lm(train_y~.,
data = data.frame(train_X[,selected_feature_option_2]))
y_hat_option_2 <- predict(lm_fit_option_2,
                newdata = data.frame(test_X[,selected_feature_option_2]))
test_error_option_2 <-  mean((test_y-y_hat_option_2)^2)
```

Recall that option 2 selects the features for a smaller linear model using the TRAINING set and estimate the test error on the validation set. The estimated MSE is 1.52.

(d) Comment on your results in (b) and (c). How does this relate to the discussion of Option 1 versus Option 2 from lecture on 2/5/2020? Explain how you can see that Option 1 gave you a useless (i.e. misleading, inaccurate, wrong) estimate of the test error.

Since we generated X and y independently, we would expect the lowest test error procedure to be predicting $\hat{y} = 0$, which from our calculation is Q3 is about 1.5 on this validation set and is close to what we get in option 2. However, option 1 seems to get us a lower test error; this is due to the "feature selection"

in stage one where we peeked at the test data. If we generate another data set with the same distributions, option 2 will have a lower test error.

5. In this problem, you will analyze a (real, not simulated) dataset of your choice with a quantitative response Y, and $p \geq 50$ quantitative predictors.

   (a) Describe the data. Where did you get it from? What is the meaning of the response, and what are the meanings of the predictors?

   We downloaded the diabetes data set from `https://web.stanford.edu/~hastie/StatLearnSparsity/data.html`. The diabetes data consist of observations on 442 patients, and the response of interest is quantitative measure of disease progression one year after baseline. Features include 10 baseline variables (age, sex, body-mass index, average blood pressure, and six blood serum measurements S_1 through S_6) and the quadratic terms between them, giving a total of 64 features.

```
data_url <- 'https://www.stanford.edu/~hastie/Papers/LARS/diabetes.data'
diabetes_data <- read.csv(data,sep = "")
# create quadratic features
diabetes_data$SEX <- (diabetes_data$SEX-1)
form <- Y~(AGE+SEX+BMI+BP+S1+S2+S3+S4+S5+S6)^2+I(AGE^2)+I(BMI^2)+
  I(BP^2)+I(S1^2)+I(S2^2)+I(S3^2)+I(S4^2)+I(S5^2)+I(S6^2)
# we remove the intercept in the feature matrix
X_new <- model.matrix(form, data = diabetes_data)[,-1]
```

   (b) Fit a least squares linear model to the data, and provide an estimate of the test error. (Explain how you got this estimate.)

   We use 5-fold CV to estimate the test error of our linear model and we will record the fold we used in `fold_id` so we can use the same split for ridge and lasso regressions below. The estimated test error (MSE) is 3869.6.

```
# extract fold_id
set.seed(1234)
n <- dim(diabetes_df)[1]
scrambled_index <- sample(c(1:n),n,replace=F)
fold_split <- split(scrambled_index,c(1:n)%%k)
fold_id <- rep(NA, length = n)
for (i in seq_along(fold_split)){
  fold_id[fold_split[[i]]] <- i
}
fit.lm.cv <- lm_k_fold_cv(k=5,lm_formula = y~.,
                          X=X_new,y=diabetes_data$Y)
```

(c) Fit a ridge regression model to the data, with a range of values of the tuning parameter λ. Make a plot like the left-hand panel of Figure 6.4 in the textbook.

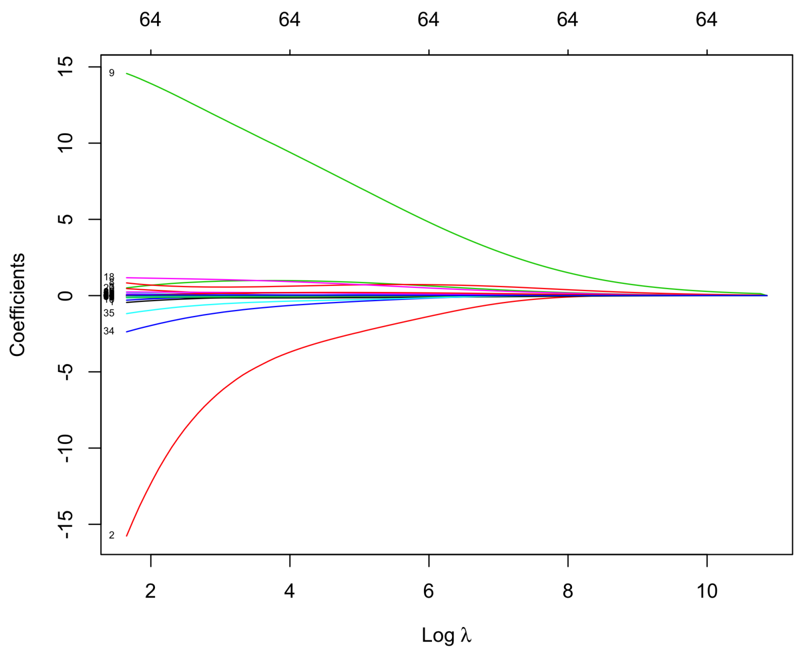

Figure 3: Standardized ridge regression coefficients as a function of log λ

```
fit.ridge <- glmnet(X_new[,-1], diabetes_data$Y, alpha=0)
plot(fit.ridge,xvar='lambda',label=TRUE, xlab=TeX('Log $\\lambda$'))
```

(d) What value of λ in the ridge regression model provides the smallest estimated test error? Report this estimate of test error. (Also, explain how you estimated test error.)

We used 5-fold cross-validation to estimate the test error (with the same fold split in linear regression). The smallest estimated test error is 3004.6, achieved by $\lambda = 5.72$

```
fit.ridge.cv <- cv.glmnet(X_new[,-1], diabetes_data$Y,
type.measure="mse", alpha=0,
     family="gaussian", foldid = fold_id)
fit.ridge.cv$lambda.min
min(fit.ridge.cv$cvm)
```

(e) Repeat (c), but for a lasso model.

```
fit.lasso <- glmnet(X_new[,-1], diabetes_data$Y, alpha=1)
plot(fit.lasso,xvar='lambda',label=TRUE, xlab=TeX('Log $\\lambda$'))
```

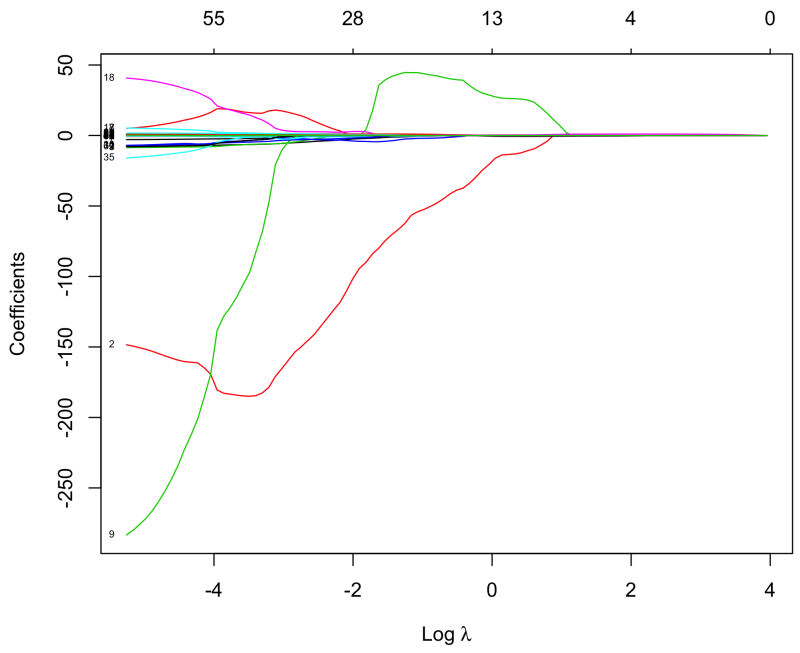

Figure 4: Standardized lasso regression coefficients as a function of log λ

(f) Repeat (d), but for a lasso model. Which features are included in this lasso model?

We used 5-fold cross-validation to estimate the test error (with the same fold split in linear regression). The smallest estimated test error is 2980.9, achieved by $\lambda = 0.41$

With $\lambda = 0.41$, the following baseline predictors are included in the lasso model: Age, Sex, S_1, S_5, BMI^2, S_6^2. These interaction terms are also selected: Age and Sex, S_2 and Sex, S_4 and Sex, Blood pressure and BMI, S_3 and BMI, S_5 and BMI, S_2 and S_5, S_3 and S_4, S_3 and S_6, and S_4 and S_6. Overall 16 out of the original 64 predictors have non-zero coefficient.

```
fit.lasso.cv <- cv.glmnet(X_new[,-1], diabetes_data$Y,
type.measure="mse", alpha=1,
     family="gaussian", foldid = fold_id)
fit.lasso.cv$lambda.min
min(fit.lasso.cv$cvm)
fit.lasso.coef <- glmnet(X_new[,-1], diabetes_data$Y, alpha=1,
 lambda = fit.lasso.cv$lambda.min)
coef(fit.lasso.coef)
```

In this problem, you may use the function in the `glmnet` package that performs cross-validation.