Data 558: Statistical Machine Learning

Spring 2023

Homework – 4

Arjun Sharma

Conceptual Questions:

1. Describe the different model selection criterions AIC, BIC, and adjusted R2, and advantages of using one over the other. Here, think about factors such as being able to explain it to general audience, obtaining simple models, theoretical support, and general applicability.

Solution:

AIC:

Akaike Information Criterion is a measure used to compare different models fitted onto the same dataset. Mathematically, it is calculated as:

$$\text{AIC} = \frac{1}{n}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

It is evident that a lower value of the AIC represents a 'better' model. As in, a lower AIC value represents a balance between the model complexity and the fit of the model. A higher value of the AIC would imply that the model is potentially too complex for the data. The AIC is a value that isn't absolute, in the sense that it would help one obtain insight as to which model fitted for a particular dataset has been fit best, relative to other models fitted on the same data.

BIC:

Bayesian Information Criterion is a measure, like the AIC used to compare different models fitted on to the same dataset. Mathematically, it is calculated as:

$$\text{BIC} = \frac{1}{n}\left(\text{RSS} + \log(n)d\hat{\sigma}^2\right)$$

Like AIC, BIC is also interpreted as "the lower, the better". However, BIC tends to penalize higher complexity models more harshly when compared to the AIC. In addition, it appears as though a large sample size would also lead to further penalization of the model. Like the AIC, the BIC also gives you a value that gives you insights on how each model compares to others fitted on the same data.

Adjusted R-squared:

The adjusted R-squared value is an absolute metric, which typically ranges from 0 to 1. Mathematically, Adjusted R-squared is represented as:

$$\text{Adjusted R-squared} = 1 - [(1-R^2)(n-1)/(n-k-1)]$$

Where $R^2 = 1 - (RSS/TSS)$

The adjusted R-squared is preferred over the R-squared metric since it takes the dimensionality of the model into account and penalizes the model for excessive features. The Adjusted R-squared will always take on values less than or equal to R^2. It is interpreted as the percentage of variance in the target that is explained by the set of predictors. For instance, Adjusted R-squared of 0.75 would mean that the model explains 75% of the variance in the target. What makes a good Adjusted R-squared value is context dependent, but the interpretation of the metric is objective, unlike AIC and BIC.

AIC vs Adjusted R-squared:

Points in favor of AIC:

- AIC has rigorous theoretical justifications which rely on asymptotic arguments (scenarios where the sample size n is very large). Despite its popularity, and even though it is quite intuitive, the adjusted R2 is not as well motivated in statistical theory as AIC.
- AIC can be used for higher complexity models. Since R-squared is the square of the correlation coefficient which can only capture linear relationships, The Adjusted R-squared may not wholly capture higher degree polynomial relationships.

Points in favor of Adjusted R-squared:

- Adjusted R-squared is objective, and easy to interpret.
- It is also easier to explain to the general audience who may not have much domain knowledge in statistics.

AIC vs BIC:

Points in Favor of AIC:

- It is relatively ‘relaxed’ in terms of penalizing model complexity. Therefore, it also allows for higher complexity models to be included in comparison with lower complexity models, when the model fit is concerned.
- It places relatively higher weightage on the goodness of fit of the model. It is favorable in situations where we are more concerned about the fit of the model.
- If we aren’t concerned about sample size, the AIC again is an appropriate metric.

Points in favor of BIC:

- BIC is appropriate in a situation where we might not have much data at our disposal and do not wish to use models of a higher complexity.

- It heavily penalizes higher complexity models, and hence is appropriate when we do not want a complex model.

BIC vs Adjusted R-squared:

Points in favor of BIC:

- BIC has rigorous theoretical justifications which rely on asymptotic arguments (scenarios where the sample size n is very large). Despite its popularity, and even though it is quite intuitive, the adjusted R2 is not as well motivated in statistical theory as BIC.
- BIC can be used for higher complexity models. Since R-squared is the square of the correlation coefficient which can only capture linear relationships, The Adjusted R-squared may not wholly capture higher degree polynomial relationships.

Points in favor of Adjusted R-squared:

- Adjusted R-squared is objective, and easy to interpret.
- It is also easier to explain to the general audience who may not have much domain knowledge in statistics.

2. Please determine whether following practices are good or not. Please explain why in either case. [Grading criterion: each question subpart is worth 4 points with 1 point assigned for the answer and 3 points for the explanation.]
   (a) We want to model a housing dataset consisting of 500 features. We decide to use Ridge Regression (ℓ2 regularization) in order to better obtain more prominent features and how they determine a house's price.
   (b) We are aware of the time cost of finding the globally optimal subset of features for a prediction task, so we resort to greedy algorithms such as forward step-wise selection. We think that the run time will be significantly reduced, even though the solution may not be optimal.
   (c) We notice that for a certain value of λ in the LASSO solution path, the training MSE is almost 0 but the cross-validation MSE is 100. To fix this, we choose a lower value of the regularization parameter λ.
   (d) Suppose we are consulting for a company that wants to know which set of features is relevant for predicting house sales. We know that some of the predictors are highly correlated. We run the LASSO and choose the regularization parameter λ via the validation set approach. We are happy because the LASSO solution at this value of λ yields only 2 non-zero coefficient predictors out of the possible 100. We go and report it to the company and tell them that these 2 predictors are relevant, while the others are irrelevant for prediction because their estimated coefficients are zero.

Solution:

(a) This is bad practice. We have 500 features in this dataset, and the goal is to obtain more prominent features and understand how they determine house price. Ridge regression does not eliminate any features, the coefficients for all features will be > 0 and hence, the process of choosing the more prominent predictors out of the said 500 will be cumbersome. Furthermore, it is practically

a reasonable assumption that not all of the 500 features are important, and that it would be computationally inexpensive to remove some of the less important features. To do this while retaining the ability to clearly comprehend the impact of each feature, lasso regression (l1 regularization) is a more appropriate approach. If the focus was purely on dimensionality reduction, I would have suggested using PCA, however that would compromise on the goal to interpret and explain the impact of each feature on the model behavior. Hence, Lasso Regression is suggested as an alternative.

(b) If we prefer simplicity in implementation and reduced computational power and time over the goodness of fit of the model – this is considered good practise. Forward step-wise selection allows us to choose each feature sequentially after assessment of each feature's performance relative to others. In this case, we obtain a reasonably well fit model while using a fraction of the computational resources and time.

If the goal was to obtain an optimal model with the best combination of parameters irrespective of time and computational resources, then this would be considered bad practise.

(c) This is very bad practise. If the training MSE is almost 0, and the testing MSE is 100, it means that the model is overfitting to the training set – this point is strengthened with the knowledge that the said metrics were obtained after cross validation. The model thus has high variance. Reducing the value of λ would lead to higher variance since the bias term would decrease. This would exacerbate the overfitting problem. If anything, the λ value must be increased in order to regularize the model by decreasing the variance as a result of increased bias. Alternatively, it is possible that the model is too complex for the data and is overfitting to the noise, the user may also use a model with lower complexity.

(d) This is bad practice. Prior to performing LASSO regression, there should have been some emphasis on removing redundant parameters, particularly by looking at the correlated features. After which, the decision to remove some of the features must have been made. In addition, coefficients being reduced to zero is not necessarily conducive to retaining only the important features. It is possible that the scale of different features is also different, and hence the smaller coefficient for one feature is a result of the scale of the concerned feature being much higher than that of other features, in which case, standardization or normalization of the features should be performed (this hasn't been mentioned as part of the procedure taken). Furthermore, the holdout validation process has been used – the inference of relevant features has been made through only one set of train and test sets.

The problem here is that we are using correlated features, which could cause redundancy, then only use the holdout validation approach. We also do not know if the data has been standardized.

It is advisable to use other methods to split the data such as k-fold cross validation or HOOCV to have multiple sets of validation and training sets, this would give us insight that is more generalized and less impacted by noise or bias. Furthermore, it is advisable to look at other processes to perform feature selection aside from just lasso. Also, it is important to standardize or normalize the data since Lasso regression is scale dependent.

3. Is there always a unique solution, $b\beta$ to ordinary least squares? If not, come up with two scenarios where there may not be a unique solution, i.e., you can obtain

bβ1 and bβ2, where both estimates minimize the RSS, but with bβ1≠ bβ2. What about for Ridge regression with λ > 0? Why or why not?

Solution:

Ordinary least Squares does not have unique solution for the following situations:

a. When p>n; when the number of predictors is greater than the number of training examples.

b. When multicollinearity is observed in the features.

In the case of Ridge Regression, for λ > 0, there is always a unique solution.

The addition of the penalty term $\lambda \sum \beta_i^2$ where I = 1,2,3,…p, to the covariance matrix ensures that all eigenvalues are greater than 0 and hence it becomes invertible. Invertibility necessitates unique solution, and hence, in the case of ridge regression for λ > 0, there is always a unique solution

4. (BONUS Question)
(a)

One useful heuristic for stopping could be to monitor the BIC of the model as the predictors are added. If the BIC score increases over a certain threshold, we can stop adding predictors to the model since the BIC penalizes more complex models very heavily.

Hypothetically, consider a threshold like; if the BIC rises by over 10% over 2 consecutive feature additions.

(b) For the full model, the BIC can be intuitively assumed high, because of the large number of features, and because the question says that only a few are relevant. Therefore we can theoretically expect to stop when

1. $RSS_{current}$ - RSS_{full} starts increasing and,

2. $BIC_{full} - BIC_{current}$ starts decreasing

Thus, we can use a theoretical optimization that simultaneously attempts to find:
a. MAX($BIC_{full} - BIC_{current}$) and

b. MIN($RSS_{current}$ - RSS_{full})

Applied Questions:

Problem 1 (25 points): Consider the linear regression model

$$\boldsymbol{y} = \beta_0 + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \in \mathbf{R}^p,$$

where $p = 10$, $\beta_0 = 0$, and the true value of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} = (0, 1, 1, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0)^T.$$

Suppose the independent error terms have distribution $\mathcal{N}(0,\, 1)$ and $X_1, \ldots, X_{10} \sim \mathcal{N}\big(0,\, 5^2\big)$, i.i.d. Generate training and test datasets, each with $n = 100$ observations, from this statistical model.

(a) Program your own best-subset selection algorithm without using built-in packages specifically for that purpose. Using BIC as your criterion, report your final model summary as well as the test error (test error will be computed by using the estimated model from the training data to make predictions on the test set, and calculating the corresponding MSE). Comment on the final model chosen by best-subset selection, in comparison to the true statistical model from which the data were generated. [Grading criterion: 2 points for writing code from scratch, 2 point for results, and 2 points for explanations]

(b) Program your own forward step-wise selection algorithm without using built-in packages specifically for that purpose. Using BIC as your criterion, report your final model summary as well as the test error. Comment on the final model chosen by forward selection, in comparison to the true statistical model from which the data were generated, as well as the model chosen by best subset selection. Are they the same? Different? [Grading criterion: 2 points for writing code from scratch, 2 point for results, and 2 points for explanations]

(c) Using built-in packages in either `R` or `Python`, perform LASSO regression. For every regularization parameter λ, plot the coefficients of the LASSO solution on the entire training data. Using BIC as your criterion, choose the optimal value of the regularization parameter λ (the LASSO solution at this value of λ will be your chosen model), and report your final model summary as well as the test error. Comment on the final model chosen by LASSO, in comparison to the true statistical model from which the data were generated, as well as the models chosen by best-subset and forward selection. Are they the same? Different?[Grading criterion: 1 points for code, 2 point for results, and 2 points for explanations]

(d) Using built-in packages in either `R` or `Python`, perform Ridge regression. For every λ, plot the coefficients of the Ridge solution on the entire training data. Using cross-validation with a criterion of your choice (i.e., does not have to be BIC), choose the optimal value of the regularization parameter λ (the Ridge solution at this value of lambda will be your chosen model), and report your final model summary as well as the test error. Comment on the final model chosen by Ridge regression, in comparison to the true statistical model from which the data were generated, as well as the models chosen by best-subset and forward selection, as well as LASSO. Are they the same? Different? [Grading criterion: 1 points for code, 2 point for results, and 2 points for explanations].

(e) Which method performs best in terms of test error? Which method performs best in terms of selecting the correct variables? Compare performance in terms of test error and selecting correct variables with OLS regression. [Grading criterion: 1 point for result, 2 points for explanations].

Solution:

```{r}
# Problem 1 - a
set.seed(1)

n <- 200
p <- 10
epsilon <- rnorm(n)

# Generate data
x_names <- paste0("X", 1:p)
xs <- replicate(p, rnorm(n, sd = 5))
betas <- matrix(c(1, 1, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0))
dataset <- data.frame(xs)
colnames(dataset) <- paste0("X", 1:p)
dataset$y <- as.vector(xs %*% betas)
dataset$y <- dataset$y + epsilon

train_data <- dataset[1:(n/2), ]
test_data <- dataset[((n/2) + 1):n, ]

best_subset_selection <- function(train, test, target_var = "y") {

  # Extract predictor variable names
  predictor_vars <- colnames(train)
  predictor_vars <- predictor_vars[predictor_vars != target_var]

  # Generate all subsets of predictors
  subsets <- lapply(1:(length(predictor_vars)), function(x) combn(predictor_vars, m = x, simplify = FALSE))

  # Perform best subset selection procedure on subsets
  best_models <- tibble(model = character(), bic = double(), n_preds = integer())

  for (subset_n in subsets) {
    min_bic <- Inf
    best_model <- NULL
    for (subset in subset_n) {
      formula_str <- paste0(target_var, " ~ ", paste(subset, collapse = " + "))
      model <- lm(as.formula(formula_str), data = train)
      bic <- BIC(model)

      if (bic < min_bic) {
        min_bic <- bic
        best_model <- formula_str
      }
    }

    best_models <- best_models %>%
      add_row(model = best_model, bic = min_bic, n_preds = length(subset))
  }

  return(best_models %>% arrange(bic, n_preds) %>% slice(1))
}

best_model <- best_subset_selection(train_data, test_data)
final_model <- lm(as.formula(best_model$model), data = train_data)
predictions <- predict(final_model, test_data)
summary(final_model)

```

```
Call:
lm(formula = as.formula(best_model$model), data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1300 -0.5297 -0.1226  0.5474  2.1225

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13195    0.09128   1.446 0.151587
X1           1.00270    0.01745  57.455  < 2e-16 ***
X2           1.02140    0.01551  65.867  < 2e-16 ***
X3           0.11395    0.01673   6.812 8.71e-10 ***
X4           0.05813    0.01662   3.497 0.000717 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.897 on 95 degrees of freedom
Multiple R-squared:  0.9883,    Adjusted R-squared:  0.9879
F-statistic:  2013 on 4 and 95 DF,  p-value: < 2.2e-16
```

```{r}
cat("Test MSE:", mean((test$y - preds)**2))
```

```
Test MSE: 0.9519375
```

We observe that the model obtained after applying the best subset process, that we only retain 4 of our features namely X1, X2, X3, and X4. There is an error term in y, and that error term could potentially contribute to noise that leads to only the 4 said features being selected. The test MSE is 0.95159375

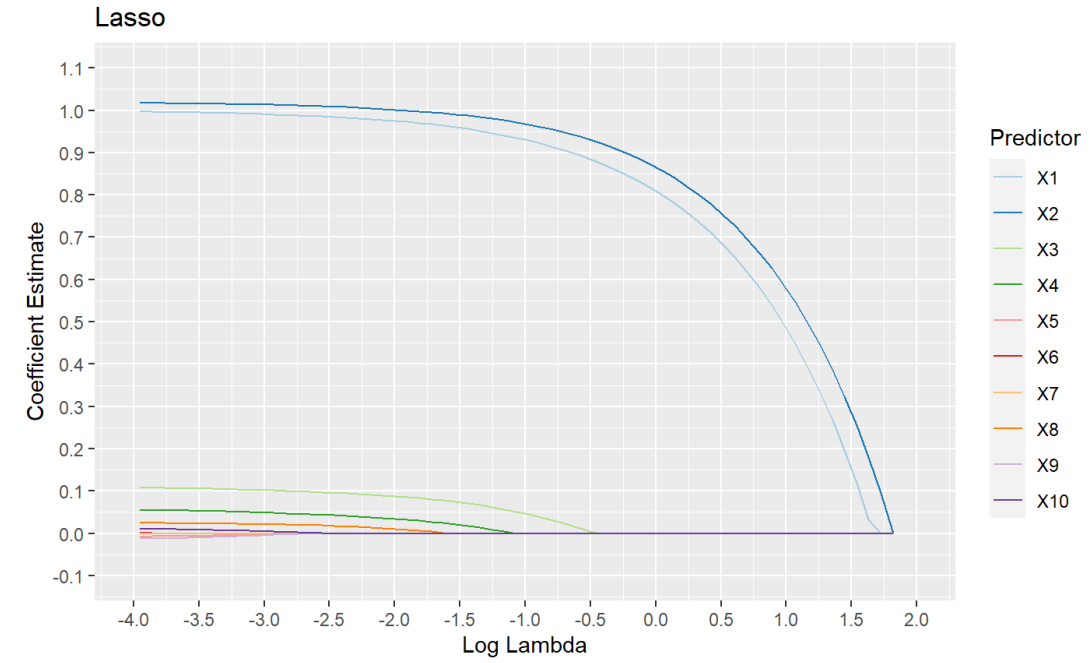
```{r}
# Problem 1 - b
forward_stepwise_selection_modified <- function(train, test, target_var = "y") {
  x_vars <- colnames(train)
  x_vars <- x_vars[x_vars != target_var]
  max_p <- length(x_vars)

  # Perform forward stepwise procedure
  null_model <- lm(as.formula(paste0(target_var, " ~ 1")), data = train)
  best_models <- tibble(model = paste0(target_var, " ~ 1"), bic = BIC(null_model), n_preds = 0)

  for (i in 1:max_p) {
    last_model <- tail(best_models$model, 1)
    remaining_vars <- x_vars[!grepl(last_model, x_vars)]
    min_bic <- Inf
    best_model <- NULL

    for (var in remaining_vars) {
      new_formula <- paste0(last_model, " + ", var)
      model <- lm(as.formula(new_formula), data = train)
      bic <- BIC(model)

      if (bic < min_bic) {
        min_bic <- bic
        best_model <- new_formula
      }
    }

    best_models <- best_models %>%
      add_row(model = best_model, bic = min_bic, n_preds = i)
  }

  return(best_models %>% arrange(bic, n_preds) %>% slice(1))
}

fstep_model_modified <- forward_stepwise_selection_modified(train_data, test_data)
final_model_modified <- lm(as.formula(fstep_model_modified$model), data = train_data)
predictions_modified <- predict(final_model_modified, test_data)
summary(final_model_modified)
```

```
Call:
lm(formula = as.formula(fstep_model_modified$model), data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1300 -0.5297 -0.1226  0.5474  2.1225

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13195    0.09128   1.446 0.151587
X2           1.02140    0.01551  65.867  < 2e-16 ***
X1           1.00270    0.01745  57.455  < 2e-16 ***
X3           0.11395    0.01673   6.812 8.71e-10 ***
X4           0.05813    0.01662   3.497 0.000717 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.897 on 95 degrees of freedom
Multiple R-squared:  0.9883,    Adjusted R-squared:  0.9879
F-statistic:  2013 on 4 and 95 DF,  p-value: < 2.2e-16
```

```{r}
cat("Test MSE:", mean((test$y - preds)**2))
```

Observing the results of the forward stepwise selection, it is noticed that the test MSE is exactly the same. Furthermore, the selected features are also the same as in the previous case.

```{r}
# Problem 1 - C
x_train <- as.matrix(train_data %>% dplyr::select(-y))
x_test <- as.matrix(test_data %>% dplyr::select(-y))
lasso_model <- glmnet(x_train, train_data$y)
coefs <- lasso_model$beta %>% as.matrix() %>% as.data.frame()
coefs$x <- rownames(coefs)
coefs %<>% pivot_longer(cols = !matches("x"), names_to = "s", values_to = "coef") %>% mutate(s =
as.numeric(str_remove(s, "s")) + 1, lambda = lasso_model$lambda[s], x = factor(x, levels = paste0("X", 1:10)))

ggplot(coefs) +
  geom_line(aes(x = log(lambda), y = coef, color = x)) +
  scale_x_continuous(limits = c(-4, 2), breaks = seq(-4, 2, .5)) +
  scale_y_continuous(limits = c(-.1, 1.1), breaks = seq(-.1, 1.1, .1)) +
  scale_color_brewer(palette = "Paired") +
  xlab("Log Lambda") +
  ylab("Coefficient Estimate") +
  ggtitle("Lasso") +
  labs(color = "Predictor")

```




```{r}
# Calculate BIC
n_obs <- lasso_model$nobs
bics <- lasso_model$df * log(n_obs) + deviance(lasso_model)

# Choose lambda with minimum BIC
best_lambda <- lasso_model$lambda[bics == min(bics)]

y_pred <- predict(lasso_model, newx = x_test_matrix, s = best_lambda)
test_mse <- mean((test_data$y - y_pred)^2)

lasso_betas <- lasso_model$beta %>% as.matrix()
data.frame(predictor = rownames(lasso_betas), coef = unname(lasso_betas[, paste0("s", which.min(bics)-1)]))


```

Description: df [10 × 2]

predictor <chr>	coef <dbl>
X1	0.98443577
X2	1.00893452
X3	0.09580075
X4	0.04256313
X5	0.00000000
X6	0.00000000
X7	0.00000000
X8	0.01766473
X9	0.00000000
X10	0.00000000

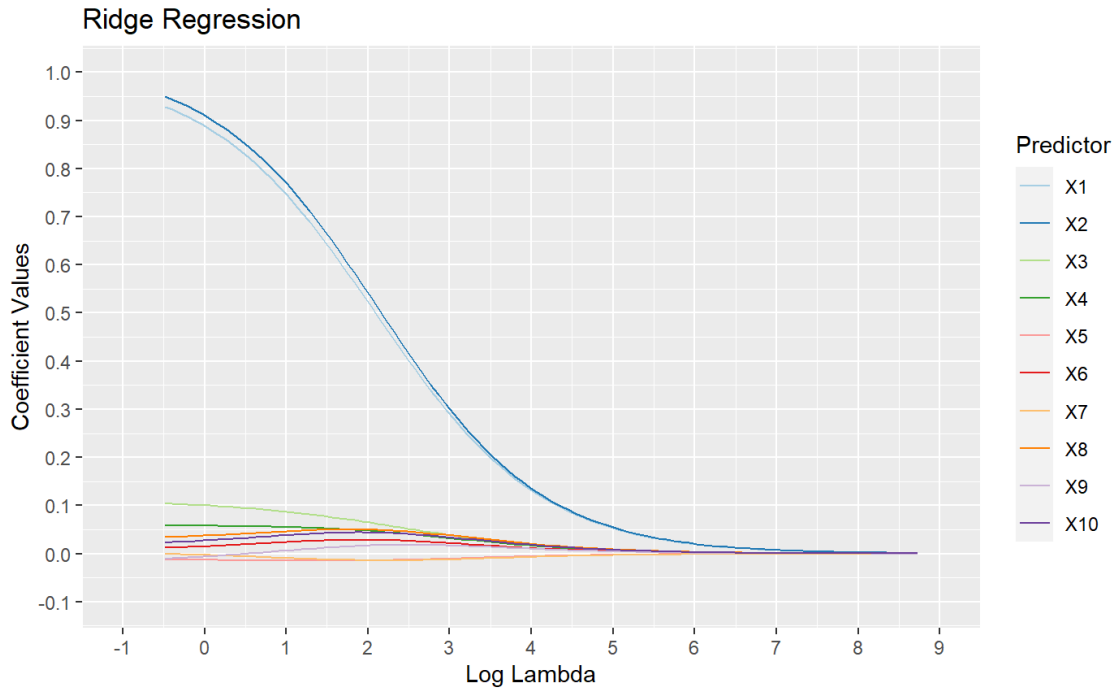
1-10 of 10 rows

```{r}

cat("Best Lambda value: ", best_lambda, "\n")
cat("Best lambda test MSE", test_mse)
```

```
Best Lambda value:  0.0860235
Best lambda test MSE 0.9732591
```

It is observed that the model selects features X1, X2, X3, X4, and X8. The best lambda value is 0.860235, and the best test MSE is 0.9732591. The coefficients for all the features X1 to X4 are similar to the previous model.

```r
#  Problem 1 - D
ridge_cv <- cv.glmnet(x = x_train_matrix, y = train_data$y, alpha = 0)

ridge_coefficients <- function(cv_model, x_train, y_train, alpha = 0) {
  coefficients <- lapply(cv_model$lambda, function(l) {
    model <- glmnet(x = x_train, y = y_train, alpha = alpha, lambda = l)
    coef_data <- coef(model) %>% as.matrix()
    return(data.frame(predictor = rownames(coef_data), lambda = l, coefficient = unname(coef_data[, 1])))
  }) %>% bind_rows()

  return(coefficients)
}

coefficients_data <- ridge_coefficients(ridge_cv, x_train_matrix, train_data$y) %>%
  filter(predictor != "(Intercept)") %>%
  mutate(predictor = factor(predictor, levels = paste0("X", 1:10)))

ggplot(coefficients_data) +
  geom_line(aes(x = log(lambda), y = coefficient, color = predictor)) +
  scale_x_continuous(limits = c(-1, 9), breaks = seq(-1, 9, 1), labels = scales::number_format(big.mark = ",")) +
  scale_y_continuous(limits = c(-.1, 1), breaks = seq(-.1, 1, .1)) +
  scale_color_brewer(palette = "Paired") +
  xlab("Log Lambda") +
  ylab("Coefficient Values") +
  ggtitle("Ridge Regression") +
  labs(color = "Predictor")
```




```{r}
predicted_values <- predict(ridge_cv, newx = x_test_matrix, s = "lambda.min")
test_mse <- mean((predicted_values - test_data$y)^2)

result <- coefficients_data %>%
  filter(lambda == ridge_cv$lambda.min) %>%
  rename(best_lambda = lambda) %>%
  mutate(best_lambda_test_mse = test_mse) %>%
  select(predictor, coefficient)

result

```

Description: df [10 × 2]

predictor <fctr>	coefficient <dbl>
X1	0.9286059022
X2	0.9493634961
X3	0.1040106544
X4	0.0584539999
X5	-0.0123203762
X6	0.0121813659
X7	-0.0005105815
X8	0.0341552800
X9	-0.0099787275
X10	0.0232683441

1-10 of 10 rows

```{r}
cat("Best lambda: ", ridge_cv$lambda.min, "\n")
cat("Best test MSE: ", test_mse)
```

```
Best lambda:  0.6161612
Best test MSE:  1.169582
```

The best lambda observed here is 0.6161612, while the best test MSE is 1.169582. It is observed that the model does not set any coefficients to 0, since it is a ridge regression model, and hence produces a more complex model. Some of the coefficients are different from the previous models as well.

```{r}
ols_fit <- lm(y ~ ., data = train)
summary(ols_fit)

```

```
Call:
lm(formula = y ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.97382 -0.56578 -0.03185  0.52975  1.82985

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.155280   0.096656   1.607  0.11170
X1           1.001414   0.018377  54.492  < 2e-16 ***
X2           1.019221   0.015761  64.665  < 2e-16 ***
X3           0.110349   0.017434   6.330 9.71e-09 ***
X4           0.059339   0.017670   3.358  0.00116 **
X5          -0.010864   0.017998  -0.604  0.54763
X6           0.005249   0.017384   0.302  0.76339
X7           0.003860   0.018494   0.209  0.83516
X8           0.027016   0.018935   1.427  0.15713
X9          -0.018560   0.018330  -1.013  0.31404
X10          0.014555   0.017746   0.820  0.41432
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9036 on 89 degrees of freedom
Multiple R-squared:  0.9889,    Adjusted R-squared:  0.9877
F-statistic:   794 on 10 and 89 DF,  p-value: < 2.2e-16
```

From the model summary above, we see which variables are statistically significant. It is observed that the subset selection and the forward step selection processes both selected X1, X2, X3 and X4, while Ridge and Lasso Regression selected more variables in addition to the ones earlier. Furthermore, the MSE in the case of Ridge and Lasso is also higher than that of the best subset selection and the forward step selection processes.

Problem 2 (16 points, 20 points with bonus): Suppose that a collection of $p = 10$ features X is generated according to a Gaussian distribution with mean

$$\mu = (10, 10, 10, 10, 10, 10, 10, 10, 10, 10)^T$$

and covariance matrix

$$\Sigma = \begin{pmatrix} 1 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 \\ .95 & 1 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 \\ .95 & .95 & 1 & .95 & .95 & .95 & .95 & .95 & .95 & .95 \\ .95 & .95 & .95 & 1 & .95 & .95 & .95 & .95 & .95 & .95 \\ .95 & .95 & .95 & .95 & 1 & .95 & .95 & .95 & .95 & .95 \\ .95 & .95 & .95 & .95 & .95 & 1 & .95 & .95 & .95 & .95 \\ .95 & .95 & .95 & .95 & .95 & .95 & 1 & .95 & .95 & .95 \\ .95 & .95 & .95 & .95 & .95 & .95 & .95 & 1 & .95 & .95 \\ .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & 1 & .95 \\ .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & .95 & 1 \end{pmatrix}.$$

Let the response variable y be generated according to the model:

$$y = X_1 + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 0.001^2)$ and X_1 represents the first variable in X. We will apply the LASSO and Ridge to this model for predicting the response from predictors.

(a) Draw $n = 100$ samples of the covariates X and then the response Y from the model above; this dataset is used for training purposes. Draw another $n = 100$ samples; this will be used to obtain test MSE. For a range of regularization parameters λ, plot the coefficients of the LASSO and Ridge solution on the entire training data. What do you notice? Then select a regularization parameter via 5-fold CV for both Ridge and LASSO. Report the regularization parameter chosen for the LASSO and Ridge. Obtain a model for each method by running the procedure on the entire training with the selected λ, and then compute the test MSE. Report the results and compare the performance of Ridge and LASSO. [Grading criterion: 2 points for code, 4 points for results, 4 points for explanations].

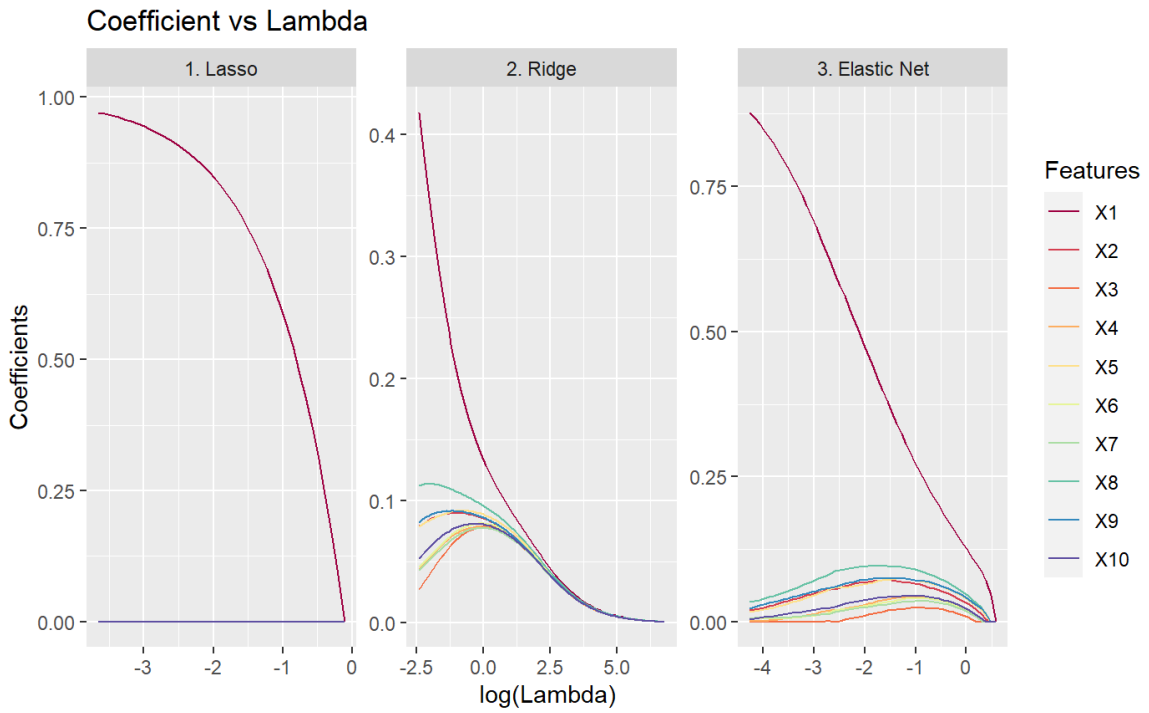
(b) Generate 50 datasets (of size $n = 100$) from the same model. Apply LASSO and Ridge with the regularization parameters you found in part (a), and compare the variability in each coefficient estimate across the datasets. What do you notice? Why do you see large or small variability? [Grading criterion: 2 point for code, 2 points for results, 2 points for explanations].

(c) BONUS 4 points – repeat part (a) and part (b) with elastic net, which is a hybrid method of LASSO and Ridge regression. What do you notice? [Grading criterion: 2 points for results, 2 points for explanations].

Solution:

```{r}

# Problem 2-a


# Set seed and variables
set.seed(1)
n_samples <- 200
n_folds <- 5

# Generate data
mean_vector <- rep(10, 10)
covariance <- matrix(rep(.95, 100), ncol = 10, nrow = 10)
diag(covariance) <- 1
data <- MASS::mvrnorm(n = n_samples, mu = mean_vector, Sigma = covariance) %>%
  as.data.frame() %>%
  rename_with(.fn = ~ str_replace(.x, "V", "X"))
e <- rnorm(n_samples, sd = .001)
data$y <- data$X1 + e
train <- data %>% slice(1:(n_samples/2))
test <- anti_join(data, train, by = names(data))
x_train <- train %>% dplyr::select(-y) %>% as.matrix()
x_test <- test %>% dplyr::select(-y) %>% as.matrix()

elastic_cv <- cv.glmnet(x = x_train_matrix, y = train$y, alpha = 0.5, nfolds = n_folds)

l1_cv <- cv.glmnet(x = x_train_matrix, y = train$y, alpha = 1, nfolds = n_folds)

l2_cv <- cv.glmnet(x = x_train_matrix, y = train$y, alpha = 0, nfolds = n_folds)
```

```{r}
elastic_coef <- glmnet_coef(elastic_cv, x_train, train$y, a = 0.5)

l1_coef <- glmnet_coef(l1_cv, x_train, train$y, a = 1)

l2_coef <- glmnet_coef(l2_cv, x_train, train$y, a = 0)

l1_coef$method <- "1. Lasso"
l2_coef$method <- "2. Ridge"
elastic_coef$method <- "3. Elastic Net"

all_coeffs <- bind_rows(l1_coef, l2_coef, elastic_coef) %>%
  mutate(x = factor(x, levels = c(paste0("X", 1:10)))) %>%
  filter(x != "(Intercept)")
ggplot(all_coeffs) +
  geom_line(aes(x = log(lambda), y = coeff, color = x)) +
  scale_color_brewer(palette = "Spectral") +
  facet_wrap(~method, scales = "free") +
  xlab("log(Lambda)") +
  ylab("Coefficients") +
  ggtitle("Coefficient vs Lambda") +
  labs(color = "Features")


```




```{r}
l1_pred <- predict(l1_cv, x_test, s = "lambda.min")
l2_pred <- predict(l2_cv, x_test, s = "lambda.min")
elastic_pred <- predict(elastic_cv, x_test, s = "lambda.min")

mse_l1 <- mean((l1_pred - test$y)**2)
mse_l2 <- mean((l2_pred - test$y)**2)
mse_elastic <- mean((elastic_pred - test$y)**2)

l1_best_coef <- l1_coef %>% filter(lambda == l1_cv$lambda.min)
l2_best_coef<- l2_coef %>% filter(lambda == l2_cv$lambda.min)
elastic_best_coef <- elastic_coef %>% filter(lambda == elastic_cv$lambda.min)

l1_best_coef$test_mse <- mse_l1
l2_best_coef$test_mse <- mse_l2
elastic_best_coef$test_mse <- mse_elastic

coeff_summary <- bind_rows(l1_best_coef,
l2_best_coef,
elastic_best_coef) %>%
rename(best_lambda = lambda) %>%
dplyr::select(method, everything())
coeff_summary
```

method <chr>	x <chr>	best_lambda <dbl>	coeff <dbl>	test_mse <dbl>
1. Lasso	(Intercept)	0.02603232	0.2888783591	0.0008243696
1. Lasso	X1	0.02603232	0.9707722284	0.0008243696
1. Lasso	X2	0.02603232	0.0000000000	0.0008243696
1. Lasso	X3	0.02603232	0.0000000000	0.0008243696
1. Lasso	X4	0.02603232	0.0000000000	0.0008243696
1. Lasso	X5	0.02603232	0.0000000000	0.0008243696
1. Lasso	X6	0.02603232	0.0000000000	0.0008243696
1. Lasso	X7	0.02603232	0.0000000000	0.0008243696
1. Lasso	X8	0.02603232	0.0000000000	0.0008243696
1. Lasso	X9	0.02603232	0.0000000000	0.0008243696

1-10 of 33 rows

Description: df [33 × 5]

method <chr>	x <chr>	best_lambda <dbl>	coeff <dbl>	test_mse <dbl>
1. Lasso	X10	0.02603232	0.0000000000	0.0008243696
2. Ridge	(Intercept)	0.08930307	0.1377158353	0.0223810544
2. Ridge	X1	0.08930307	0.4184002749	0.0223810544
2. Ridge	X2	0.08930307	0.0793135559	0.0223810544
2. Ridge	X3	0.08930307	0.0278385212	0.0223810544
2. Ridge	X4	0.08930307	0.0443510923	0.0223810544
2. Ridge	X5	0.08930307	0.0790988591	0.0223810544
2. Ridge	X6	0.08930307	0.0459836697	0.0223810544
2. Ridge	X7	0.08930307	0.0431462037	0.0223810544
2. Ridge	X8	0.08930307	0.1122188200	0.0223810544

11-20 of 33 rows Previous 1 2 3 4 Next

Description: df [33 × 5]

method <chr>	x <chr>	best_lambda <dbl>	coeff <dbl>	test_mse <dbl>
2. Ridge	X9	0.08930307	0.0827201675	0.0223810544
2. Ridge	X10	0.08930307	0.0526827239	0.0223810544
3. Elastic Net	(Intercept)	0.01415423	0.0978623748	0.0011572013
3. Elastic Net	X1	0.01415423	0.8777388615	0.0011572013
3. Elastic Net	X2	0.01415423	0.0196720296	0.0011572013
3. Elastic Net	X3	0.01415423	0.0000000000	0.0011572013
3. Elastic Net	X4	0.01415423	0.0008083907	0.0011572013
3. Elastic Net	X5	0.01415423	0.0178657538	0.0011572013
3. Elastic Net	X6	0.01415423	0.0052656785	0.0011572013
3. Elastic Net	X7	0.01415423	0.0064184082	0.0011572013

21-30 of 33 rows Previous 1 2 3 4 Next

Description: df [33 × 5]

method <chr>	x <chr>	best_lambda <dbl>	coeff <dbl>	test_mse <dbl>
3. Elastic Net	X8	0.01415423	0.0339986510	0.0011572013
3. Elastic Net	X9	0.01415423	0.0234695123	0.0011572013
3. Elastic Net	X10	0.01415423	0.0048341156	0.0011572013

31-33 of 33 rows Previous 1 2 3 4 Next

The Lasso Regression Method has a non-zero coefficient for only X1, as it encourages all other coefficients towards zero – resulting in a much less complex model compared to Ridge and Elastic Net Methods. It also has the lowest test_MSE among the 3 methods. In these considerations, Lasso regression performs the best.

In order of performance: Lasso > Elastic Net > Ridge

```r
# Problem 2 - b
set.seed(1)

generate <- function(n, mean_vec, covar) {
  epsilon <- rnorm(n, sd = .001)
  dataframe <- MASS::mvrnorm(n = 100, mu = mean_vec, Sigma = covar) %>%
    as.data.frame() %>%
    rename_with(.fn = ~ str_replace(.x, "V", "X"))
  dataframe$y <- dataframe$X1 + epsilon
  return(dataframe)
}

data <- replicate(
  50,
  generate(100, mean_vector, covariance),
  simplify = FALSE
)

l1_lambda <- l1_cv$lambda.min
l2_lambda <- l2_cv$lambda.min
elastic_lambda <- elastic_cv$lambda.min
```

```r
get_coef <- function(index, data, l1_lambda, l2_lambda, elastic_lambda) {
  x_mat <- data[[index]] %>% dplyr::select(-y) %>% as.matrix()
  l1_mod <- glmnet(x_mat, data[[index]]$y, alpha = 1, lambda = l1_lambda)
  l2_mod <- glmnet(x_mat, data[[index]]$y, alpha = 0, lambda = l2_lambda)
  elastic_mod <- glmnet(x_mat, data[[index]]$y, alpha = 0.5, lambda = elastic_lambda)
  l1_coef <- coef(l1_mod) %>% as.matrix()
  l2_coef <- coef(l2_mod) %>% as.matrix()
  elastic_coef <- coef(elastic_mod) %>% as.matrix()
  return(
    data.frame(
      dataset_num = index,
      x = rownames(l1_coef),
      l1_coef = unname(l1_coef[, 1]),
      l2_coef = unname(l2_coef[, 1]),
      elastic_coef = unname(elastic_coef[, 1])
    )
  )
}

coefs <- lapply(1:50, function(x) get_coef(x, data, l1_lambda, l2_lambda, elastic_lambda)) %>%
  bind_rows() %>%
  filter(x != "(Intercept)") %>%
  mutate(x = factor(x, levels = paste0("X", 1:10)))
```

```r
var_coefs <- coefs %>%
  group_by(x) %>%
  summarise(
    l1_coef_std = sd(l1_coef),
    l2_coef_std = sd(l2_coef),
    elastic_coef_std = sd(elastic_coef),
    .groups = "drop"
  )

var_coefs
```

x <fctr>	l1_coef_std <dbl>	l2_coef_std <dbl>	elastic_coef_std <dbl>
X1	0.002071216	0.03032718	0.016426474
X2	0.000000000	0.02016980	0.008686767
X3	0.000000000	0.02050512	0.008091338
X4	0.000000000	0.02060507	0.008737584
X5	0.000000000	0.02045466	0.010471568
X6	0.000000000	0.01910155	0.010409975
X7	0.000000000	0.01849799	0.010669097
X8	0.000000000	0.02315090	0.011209341
X9	0.000000000	0.02081795	0.010853757
X10	0.000000000	0.02378731	0.013195830

1-10 of 10 rows

The lasso regression method had the least variability. Elastic Net is 'in-between' lasso and ridge in that it showcases intermediate levels of variability relative to ridge and lasso. Ridge has the highest levels of variability in the coefficients

Problem 3 (25 points): The Framingham study is one of the well known long term follow-up studies to identify the relationship between various risk factors and diseases and to characterize the natural history of the chronic circulatory disease process. The data on various aspects have been, and continue to be, collected every two years on a cohort of individuals, with limited goals of investigating the serum cholesterol, smoking and elevated blood pressure as the risk factors of coronary heart disease.

The data, framingham.csv, can be downloaded from Canvas under the Files tab. There are 12 columns in the data file. The 1st column gives the age of the individual when they entered the study. The 2nd column provides the assigned-sex-at-birth of the individual(1-male, 2-female); the 3rd and 4th columns provide body mass index (BMI) at the baseline and at 10 years from the baseline respectively; the 5th column provides the number of cigarettes per day the individual smoked at the baseline. The columns 6 – 11 provide serum cholesterol levels at the baseline(enrollment) and then every two years through year 10. The column 12 indicates whether the individual is alive(0) or dead(1) at the end of 30 years since enrollment. That is, the data set excludes those who died during the 10 year data collection period. -9 indicates the missing data. Note that you have to convert -9 to NA or empty entries and then drop all rows with missing values before you do any analysis.

For all the following problems, you may use built-in packages in either `R` or `Python`. The goal is prediction where the response variable is whether the individual is alive or dead, and the remaining variables are covariates. Throughout, we will use regularized logistic regression (this can be achieved in `R` by including the argument `family = "binomial"` in the call to glmnet() - similar modifications can by made in `Python` and can be found in function documentation).

(a) Write a function that has three arguments (dataset, outcome variable, list of potential covariates), and returns the model chosen by logistic regression with L_1 penalty, using cross validation with your chosen criterion (e.g., accuracy rate) to select the optimal value of the regularization parameter λ. Apply your written function to this dataset. Summarize your results. [Grading criterion: 3 points for code, 3 points for results, 2 point for explanations].

(b) Write a function that has three arguments (dataset, outcome variable, list of potential covariates), and returns the model chosen by logistic regression with L_2 penalty, using cross

validation with your chosen criterion to choose the optimal value of the regularization parameter λ. Apply your written function to this dataset. Summarize your results. [Grading criterion: 3 points for code, 3 points for results, 2 point for explanations].

(c) Compare the results from the two different methods, as well as to plain logistic regression without any regularization. How do the estimated models differ (for example, the value of estimated coefficients, prediction accuracy, etc. - choose 2 aspects for comparison)? Are estimated coefficients for the same covariate across different models similar?[Grading criterion: 4 points for explanations].

(d) Given that this is a longitudinal study, between the logistic regression with L_1 penalty estimates and the logistic regression with L_2 penalty estimates, which do you expect to have a higher variance? [Grading criterion: 1 point for answer and 4 points for explanations].

Solution:

```{r}
# Problem 3 - a
set.seed(1)
df <- read.csv("C:/Users/arjun/Downloads/framingham.csv")
df[df == -9] <- NA
df <- df[complete.cases(df),] %>% as.data.frame()
df_train <- df %>% slice_sample(prop = 0.8)
df_test <- anti_join(df, df_train, by = names(df))

logistic_regression <- function(data, target, features){
x <- data[, features] %>% as.matrix()
y <- data[, target]
logistic_fit <- cv.glmnet(x, y, family = "binomial", type.measure = "class")
return(glmnet(x, y, family = "binomial", lambda = logistic_fit$lambda.min))
}


x_vars <- df %>% dplyr::select(-death) %>% names()
logistic_1<- logit_l1(df_train, "death", x_vars)
logistic_1_preds <- predict(logistic_1,
newx = as.matrix(fram_test[, x_vars]),
type = "response") %>% as.vector()
logistic_1_preds <- ifelse(logistic_1_preds > .5, 1, 0)
coef(logistic_1)

```

```
                       s0
(Intercept) -6.032661905
age          0.108449133
sex         -0.453880656
BMI0         0.032290743
BMI10       -0.031322464
cigarettes   0.026443505
chol0        0.003670408
chol2       -0.000393135
chol4       -0.002993775
chol6       -0.001102899
chol8       -0.002099009
chol10       0.003433402
```

It is observed that the model produces a non-zero coefficient for all the features in the dataset. The highest coefficient belongs to age, while the lowest coefficient is from sex.

```{r}
# Problem 3 - b
logistic_2 <- function(data, target, features){
  x <- data[, features] %>% as.matrix()
  y <- data[, target]
  logistic_fit <- cv.glmnet(x, y, family = "binomial", type.measure = "class", alpha = 0)
  return(glmnet(x, y, family = "binomial", lambda = logistic_fit$lambda.min, alpha = 0))
}

x_vars <- df %>% dplyr::select(-death) %>% names()
model_2 <- logistic_2(df_train, "death", x_vars)
model_2_preds <- predict(model_2,
                    newx = as.matrix(df_test[, x_vars]),
                    type = "response") %>% as.vector()
model_2_preds <- ifelse(model_2_preds > 0.5, 1, 0)
coef(model_2)

```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept) -5.405688e+00
age          9.303413e-02
sex         -4.310379e-01
BMI0         3.497662e-02
BMI10       -3.160729e-02
cigarettes   2.208121e-02
chol0        3.974396e-03
chol2       -9.851155e-05
chol4       -2.374808e-03
chol6       -1.428792e-03
chol8       -2.061420e-03
chol10       3.097362e-03
```

Like in the case of problem 3 – a, the model produced non-zero coefficients for all predictors. Once again, age has the highest coefficient, while chol2 has the lowest coefficient.

```{r}
# Problem 3 - c
model_3 <- glm(death ~ ., data = df_train, family = binomial)
summary(model_3)

```

```
Call:
glm(formula = death ~ ., family = binomial, data = df_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5737  -0.6631  -0.4481  -0.2801   2.6571

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.1323188  0.7636736  -8.030 9.75e-16 ***
age          0.1128480  0.0099457  11.346  < 2e-16 ***
sex         -0.5214524  0.1653590  -3.153  0.00161 **
BMI0         0.0774429  0.0375772   2.061  0.03931 *
BMI10       -0.0762832  0.0374920  -2.035  0.04189 *
cigarettes   0.0284725  0.0065483   4.348 1.37e-05 ***
chol0        0.0053407  0.0027114   1.970  0.04887 *
chol2       -0.0008596  0.0029344  -0.293  0.76958
chol4       -0.0048292  0.0030731  -1.571  0.11608
chol6       -0.0025533  0.0033009  -0.774  0.43921
chol8       -0.0038916  0.0030525  -1.275  0.20236
chol10       0.0074284  0.0032513   2.285  0.02233 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1396.7  on 1377  degrees of freedom
Residual deviance: 1195.7  on 1366  degrees of freedom
AIC: 1219.7

Number of Fisher Scoring iterations: 5
```

```{r}
model_3_preds <- predict(model_3,
newdata = df_test[, x_vars], type = "response")
model_3_preds <- ifelse(model_3_preds > .5, 1, 0)

coefs_model_3 <- data.frame(feature = rownames(as.matrix(logistic_1$beta)),
model1_coefs = matrix(logistic_1$beta)[, 1],
model2_coefs = matrix(model_2$beta)[, 1],
model_3_coefs = unname(coef(
logit))[2:(length(x_vars)+1)])
coefs_model_3
```

Description: df [11 × 4]

feature <chr>	model1_coefs <dbl>	model2_coefs <dbl>	model_3_coefs <dbl>
age	0.108449133	9.303413e-02	0.100595450
sex	-0.453880656	-4.310379e-01	-0.541112330
BMI0	0.032290743	3.497662e-02	0.082575428
BMI10	-0.031322464	-3.160729e-02	-0.076406010
cigarettes	0.026443505	2.208121e-02	0.027106902
chol0	0.003670408	3.974396e-03	0.007627897
chol2	0.000000000	-9.851155e-05	-0.001661101
chol4	-0.002993775	-2.374808e-03	-0.004868027
chol6	-0.001102899	-1.428792e-03	-0.004184781
chol8	-0.002099009	-2.061420e-03	-0.003155178

1-10 of 11 rows

```{r}
model1_accuracy <- mean(fram_test$death == logistic_1_preds)*100
model2_accuracy <- mean(fram_test$death == model_2_preds)*100
model3_accuracy <- mean(fram_test$death == model_3_preds)*100
accuracies <- data.frame(model = c("Model 1", "Model 2", "Model 3"),
accuracies = c(model1_accuracy, model2_accuracy, model3_accuracy))
accuracies
```

Description: df [3 × 2]

model <chr>	accuracies <dbl>
Model 1	80.07737
Model 2	74.85493
Model 3	74.08124

Model 3 is the plain logistic regression model. It is observed that the coefficients for all models are quire similar for the most part, barring some minor differences on the same scale, in which case the coefficients for the models 1 and 2 are closer. In addition, model 3 has the least accuracy of all the models, while Model 1 has the highest accuracy. It is observed that the model with L1 (lasso) Regularization performs the best.

Problem 3 – d:

I would expect the model with L1 penalty estimates to have a higher variance. This is because:

1. The l1 regularization penalty term is of a lower scale than that of the l2 regularization term. Considering this, by the bias-variance tradeoff, we expect the l1 penalty to have lesser impact on the increase of bias compared to l2.

2. The l1 regularization approach would encourage some coefficients to be zero, while the l2 regularization does not do that. Hence, L2 regularization also has more terms that contribute to the penalty.

Hence, the logistic regression model with l2 penalty estimates has lesser variance than a logistic regression model with the l1 penalty estimate.