Assignment 1

Prasanth Yethirajula

PART A

QA1. What is the main purpose of regularization when training predictive models?

Regularization's main role is to avoid overfitting and regularization helps stop our model from learning too much from the training data, including mistakes or "noise," which can make it do poorly on new data it hasn't seen before. It's like adding a penalty to the model when it gets too complicated, encouraging it to be more general and not too focused on the details of the training data. This helps the model perform better on new data.

There are three main types of regularization: 1. L1 regularization (Lasso) makes some data points less important. 2. L2 regularization (Ridge) smoothens the data points overall. 3. Elastic net is a mix of both L1 and L2, using both methods to improve the model.

QA2. What is the role of a loss function in a predictive model? And name two common loss functions for regression models and two common loss functions for classification models.

The loss function's job is to check how good the model is at making predictions. It does this by comparing the model's predictions to the real answers and measuring how far off they are. After setting up the problem and picking a loss function, we try to make this error as small as possible. This means we adjust the model's settings so that it makes the smallest mistake possible. For predicting numbers (like in regression models), we often use methods like the sum of squared errors (SSE) or the average difference (MAE). For deciding between categories (like in classification models), we use methods like binary cross entropy or negative log-likelihood. Picking the right way to measure mistakes is crucial because it guides how the model learns.

QA3. Consider the following scenario. You are building a classification model with many hyperparameters on a relatively small dataset. You will see that the training error is extremely small. Can you fully trust this model? Discuss the reason.

If we make a classification model with too many details for a small amount of data, the model might learn the data too well, including the mistakes or "noise." This can make it look like it's doing a great job because it has a small error when we check it with the training data. However, this model won't be reliable because it's too focused on the small details of the training data and likely won't do well with new data it hasn't seen before.

QA4. What is the role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models?

The lambda parameter helps control how much we prevent our linear model, like Lasso or Ridge regression, from becoming too complex. Think of it as a dial: turning it up (a higher lambda value) tones down the model, making it simpler by reducing the importance of the data's features. This makes the model more general but might make it less accurate on the data it was trained on. Turning the dial down (a lower lambda value) lets the model pay more attention to the details, which might make it better at predicting the training data but could also make it too focused on those details and not great at handling new, unseen data. Finding the best setting for this dial is crucial, and we do that through a process called cross-validation.

PART B

Loading all required packages

```
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

Loading required attributes from car seats

```
set.seed(2025)
Carseats<-Carseats%>%select("Sales",
"Price","Advertising","Population","Age","Income","Education")
```

QB1. Build a Lasso regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education"). What is the best value of lambda for such a lasso model?

```
set.seed(2025)
attributes<-Carseats%>%select("Price","Advertising","Population","Age","Income","Education")
Sales<-Carseats %>% select("Sales")

preProcess(Sales)
```

```
## Created from 400 samples and 1 variables
##
## Pre-processing:
##   - centered (1)
##   - ignored (0)
##   - scaled (1)
```

```
preProcess(attributes)
```

```
## Created from 400 samples and 6 variables
##
## Pre-processing:
##   - centered (6)
##   - ignored (0)
##   - scaled (6)
```

```
X<-as.matrix(attributes)
Y<-as.matrix(Sales)
```

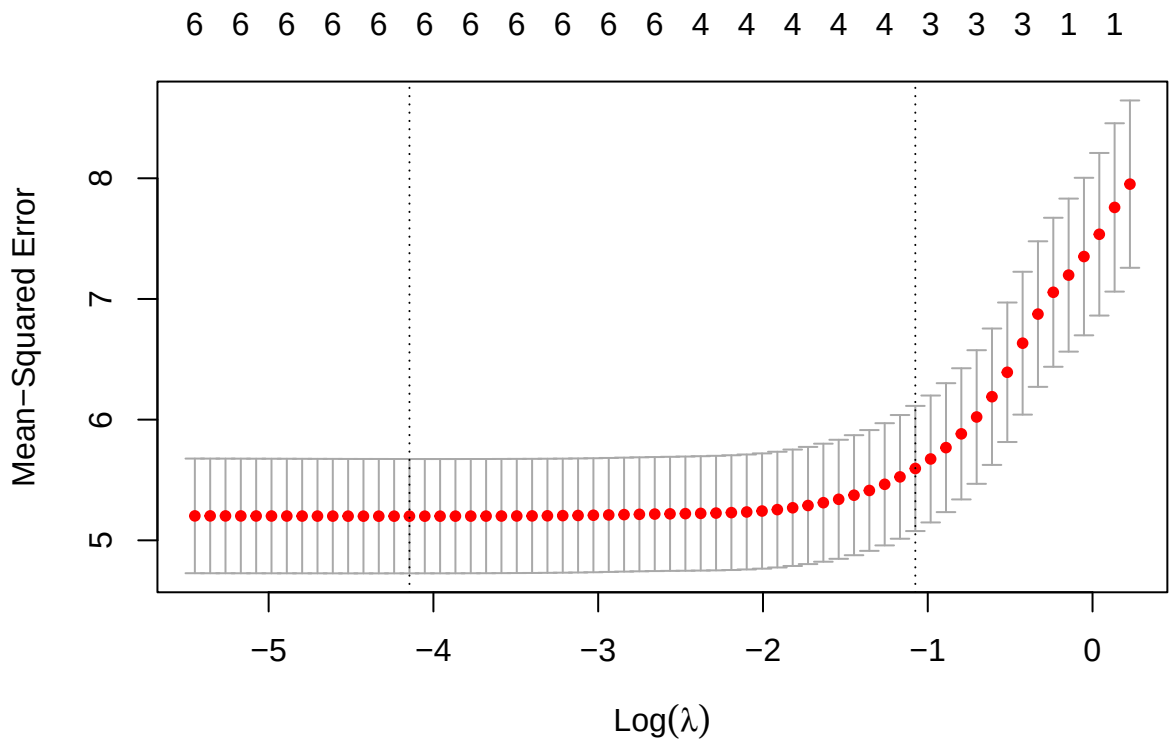
Perfoming K-fold validation to find best lamda

```
set.seed(2025)
Model<-cv.glmnet(X,Y,alpha=1)
Lamda_best<-Model$lambda.min
Lamda_best
```

```
## [1] 0.01583656
```

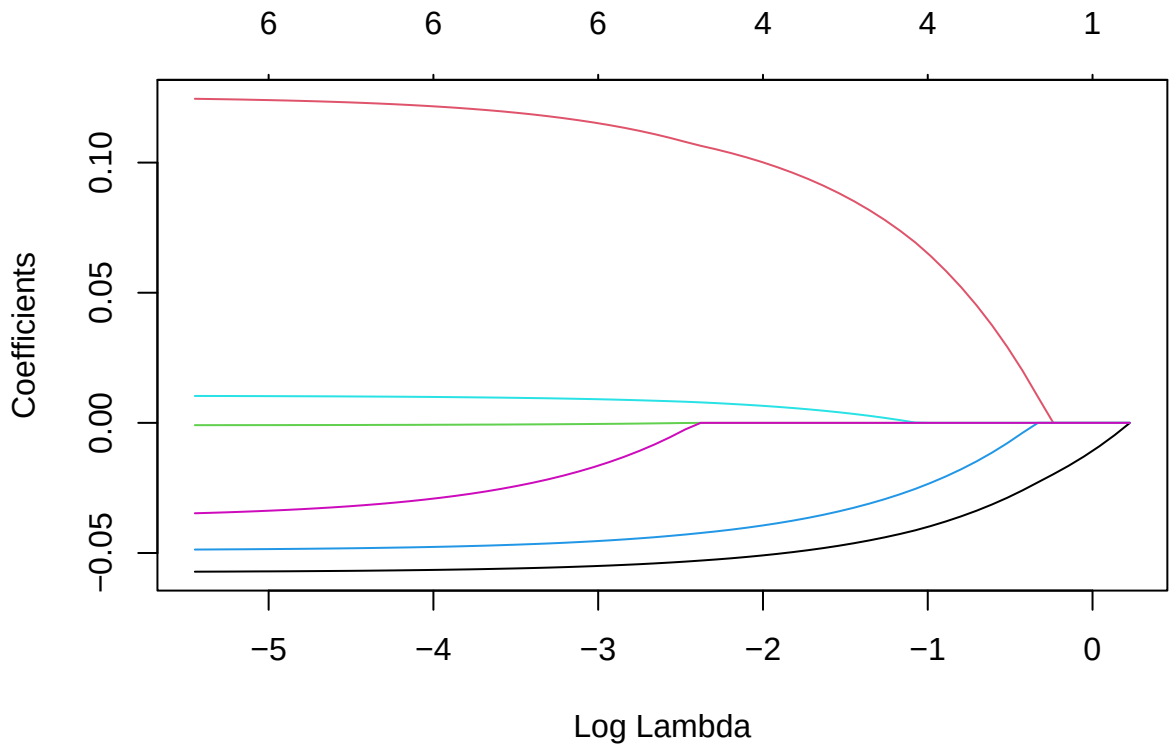
Plotting the results

```
set.seed(2025)
plot(Model)
```



Building Lasso regression model

```r
set.seed(2025)
Model_Lasso<-glmnet(X,Y,alpha=1)
plot(Model_Lasso,xvar="lambda")
```



The best value of lambda for lasso model was 0.004305309

QB2.What is the coefficient for the price (normalized) attribute in the best model ?

Building the model with best lamda

```r
set.seed(2025)
Model_best<- glmnet(X,Y,alpha = 1, lambda = Lamda_best)

coef(Model_best)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept) 15.7274114714
## Price       -0.0566235262
## Advertising  0.1221352645
## Population  -0.0007661568
## Age         -0.0478445271
## Income       0.0100173467
## Education   -0.0301007150
```

The coefficient for the price attribute in the best model was -0.0571806037

QB3.How many attributes remain in the model if lambda is set to 0.01? How that number changes if lambda is increased to 0.1? Do you expect more variables to stay in the model (i.e., to have non-zero coefficients) as we increase lambda?

Building the model with lambda value set to 0.01

```
set.seed(2025)
Model_0.01<-glmnet(X,Y,alpha = 1, lambda = .01)
coef(Model_0.01)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept) 15.8120560645
## Price       -0.0569054918
## Advertising  0.1233407471
## Population  -0.0008269765
## Age         -0.0482649383
## Income       0.0101795876
## Education   -0.0324465388
```

From the above results it is clear that no attributes are eliminated when lambda was set to 0.01

Building the model with lambda value set to 0.01

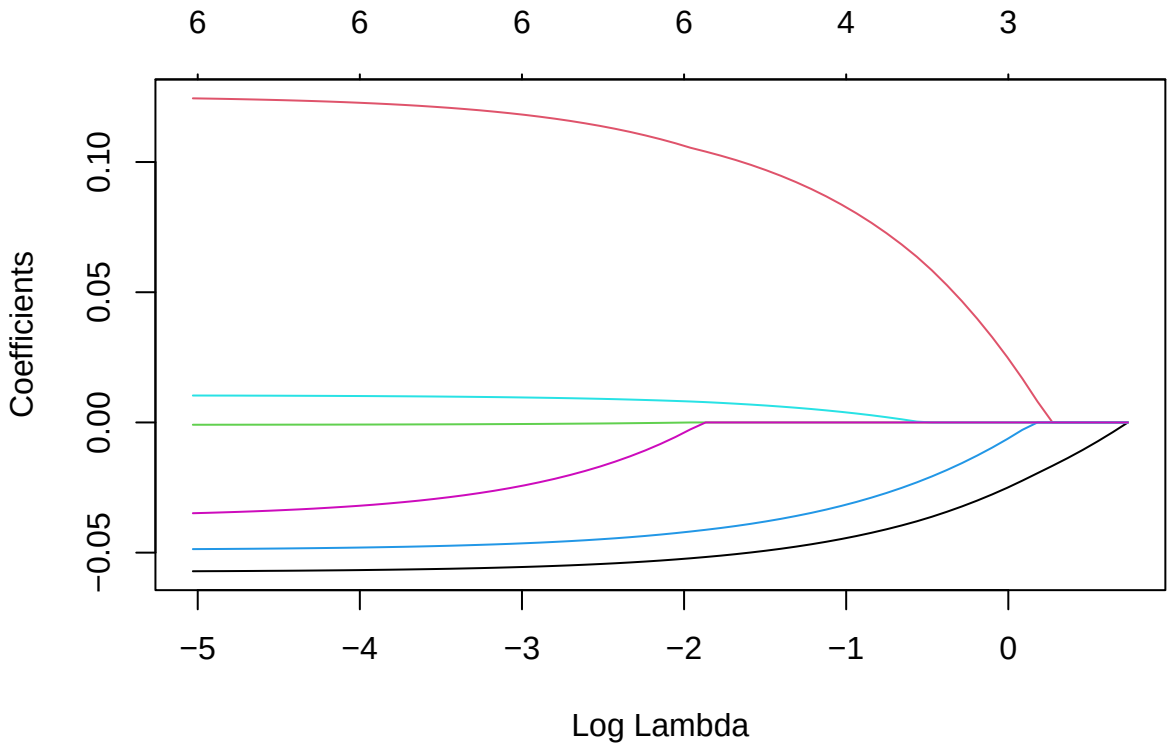
```
set.seed(2025)
Model_0.1<-glmnet(X,Y,alpha = 1, lambda = .1)
coef(Model_0.1)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) 14.59030452
## Price       -0.05257390
## Advertising  0.10536612
## Population   .
## Age         -0.04182286
## Income       0.00764389
## Education    .
```

When we make the lambda value bigger, to 0.1, it gets rid of the population and education details, and we're only left with price, advertising, age, and income. This shows us that if we increase the lambda value, we end up with fewer things in our model because the importance (or "coefficients") of those details shrinks to zero.

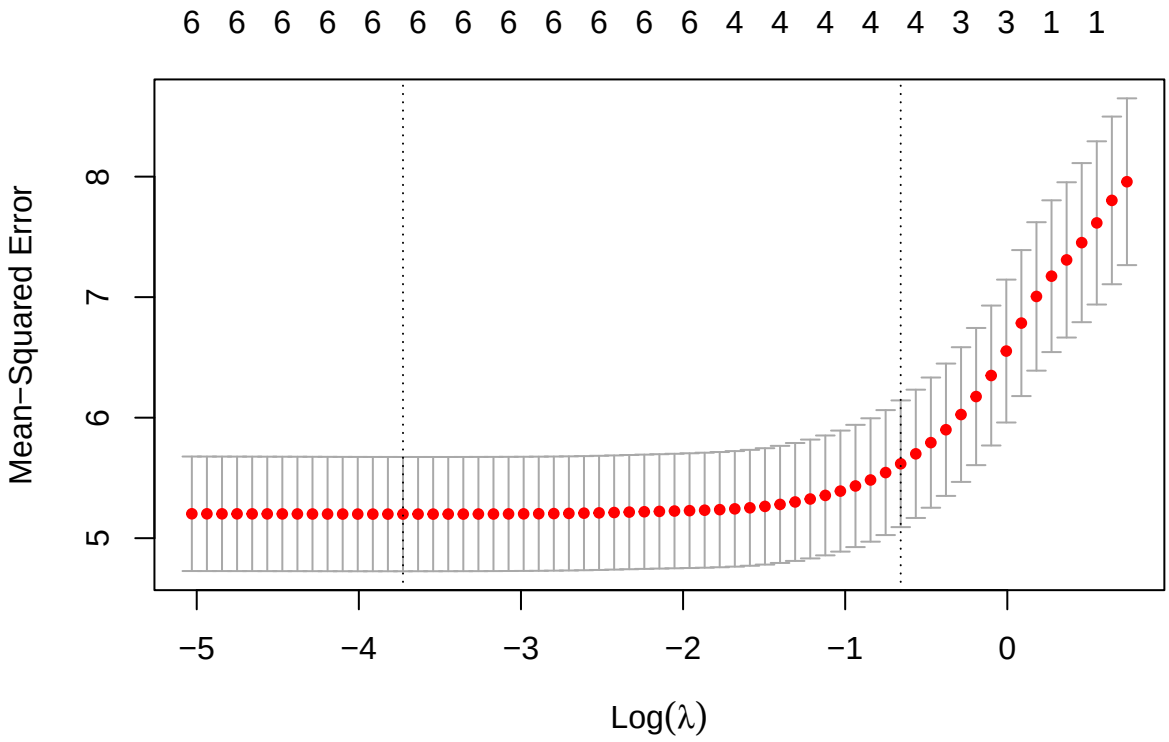
QB4. Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model?

```
set.seed(2025)
Model_elasticnet<-glmnet(X,Y, alpha=0.6)
plot(Model_elasticnet, xvar = "lambda")
```



Perfoming K-fold validation to find best lamda

```
set.seed(2025)
Model_cvelastic <- cv.glmnet(X,Y, alpha=0.6)
plot(Model_cvelastic)
```




```
el_bestlamda<-Model_cvelastic$lambda.min
el_bestlamda
```

```
## [1] 0.02404947
```

The best value of lambda for the elastic net model with alpha set to 0.6 is 0.006538062