Part A

Please read the following questions carefully and answer each question.

QA1. What is the main purpose of regularization when training predictive models? 10 points

The main purpose of is regularization tries to optimize the performance on the training set to avoid underfitting but simultaneously penalize the model when the model becomes too complex to avoid overfitting. In other words, it also serves as a countermeasure to overfitting

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. 10 points

A loss function in Machine Learning is a measure of how accurately your ML model is able to predict the expected outcome i.e the ground truth. (Seif, 2019)

The loss function is the function that computes the distance between the current output of the algorithm and the expected output. It's a method to evaluate how your algorithm models the data (Pere, 2020)

The two most common loss functions for Machine Learning Regression are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE)

The two common loss functions for classification models are the" likelihood function" which is relatively simple and is commonly used in classification problems. The function takes the predicted probability for each input example and multiplies them (DataRobot, 2018). And although the output isn't exactly human-interpretable, it's useful for comparing models. and Binary Cross-Entropy (Log Loss) (Kumar, 2020)

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. 10 points

Not really, this obviously translates to model complex/overfitting the model. We have to do a balancing act here to trust the model. As we decrease the complexity of the model, the model can better follow the data points in the training set, and therefore the training error will increase, though as we decrease the complexity of the model. This is where models start to underfit by capturing less detail and gaining their generalization capability. The sweet spot is where the test error

QA4. What is the role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models? 10 points

The role of the lambda parameter is basically to regularize linear models, lambda. In other words, lambda balances between minimizing the sum squares of the error terms on the training set and shirking the model's coefficients. Higher lambda gives Increasing the inverse of lambda parameter or in other words, decreasing lambda lead to improving both errors in both lambda parameter. This applies to both Lasso and Ridge regression models as the models become more flexible.

Part B

This part of the assignment involves building generalized linear regression models to answer a number of questions. We will use the `Carseats` dataset that is part of the ISLR package (you need to install and load the library). We may also need the following packages: `caret`, `dplyr` and `glmnet`

For this assignment, we only need the following attributes: "Sales", "Price", "Advertising", "Population", "Age", "Income" and "Education". The goal of the assignment is to build models to predict the sales of the carseats ("Sales" attribute) using the other attributes.

We can use the dplyr select function to select these attributes.

```
Carseats_Filtered <- Carseats %>% select("Sales", "Price", "Advertising","Pop
ulation","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? (Hint1: Do not forget to scale your input attributes – you can use the caret preprocess() function to scale and center the data. Hint 2: glment library expect the input attributes to be in the matrix format. You can use the as.matrix() function for converting)-- 20 Points

The step-by-step taken to build the model is knitted in R, please find herewith a copy in pdf.

```
#Fit regularized model
set.seed(123)
(caret_glmnet <- train(ï..Sales ~ .^2,
                 method = "glmnet",
                 preProcess = c("center", "scale"),
                 data = cbind(ï..Sales = Carseats_Filtered$ï..Sales, imputed)))
```

```
glmnet

400 samples
  6 predictor

Pre-processing: centered (21), scaled (21)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
Resampling results across tuning parameters:

  alpha  lambda       RMSE      Rsquared   MAE
  0.10   0.002510041  2.361536  0.3178890  1.905926
  0.10   0.025100406  2.339785  0.3283492  1.883929
  0.10   0.251004065  2.316719  0.3408022  1.859437
  0.55   0.002510041  2.359528  0.3186972  1.904188
  0.55   0.025100406  2.325419  0.3361940  1.870208
  0.55   0.251004065  2.329992  0.3391446  1.869681
  1.00   0.002510041  2.357540  0.3195463  1.902436
  1.00   0.025100406  2.316441  0.3409950  1.863793
  1.00   0.251004065  2.348534  0.3379103  1.883887

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.02510041.
```

The best value of lambda for this lasso model is 0.02510041. RMSE was used to select the optimal model using the smallest value

QB2. What is the coefficient for the price (normalized) attribute in the best model (i.e. model with the optimal lambda)? --15 points

```
# Retrieve coefficients of the best model
coef(caret_glmnet$finalModel, caret_glmnet$finalModel$tuneValue$lambda)
```

```
22 x 1 sparse Matrix of class "dgCMatrix"
                                 s1
(Intercept)              7.49632500
Price                   -1.32359297
Advertising              0.58310257
Population              -0.02818796
Age                     -0.71713420
Income                   0.05132828
Education               -0.13353861
Price:Advertising        .
Price:Population         .
Price:Age                .
Price:Income             .
Price:Education          .
Advertising:Population   .
Advertising:Age          .
Advertising:Income       0.25325802
Advertising:Education    .
Population:Age          -0.08694917
Population:Income        .
Population:Education     .
Age:Income               .
Age:Education            .
Income:Education         0.14831552
```

The coefficient for the price (normalized) attribute in the best model is -1.32359297

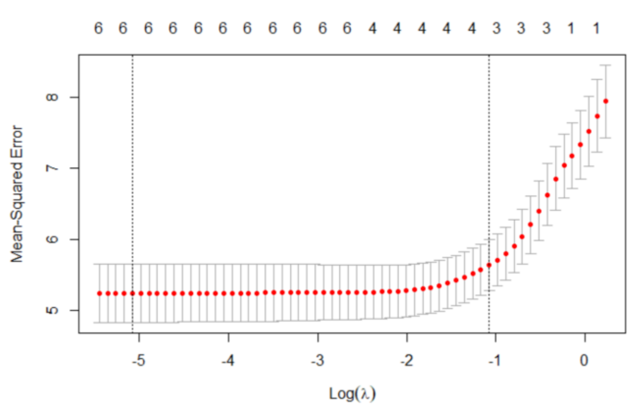
QB3. How many attributes remain in the model if lambda is set to 0.01? How does that number change if lambda is increased to 0.1? Do you expect more variables to stay in the model (i.e., to have non-zero coefficients) as we increase lambda? – 15 points

We first need to define the model equation by formulating the predictors (X) and the outcome (Y), the penalty type, and MSE for several lambdas

```
library(faraway)
set.seed(1233)
#we need to define the model equation
  X <- model.matrix(ï..Sales ~  Price + Advertising + Population + Age + Income + Education, data=
Carseats_Filtered)[,-1]
#and the outcome
  Y <- Carseats_Filtered[,"ï..Sales"]
```

#First we need to find the amount of penalty, λ by cross-validation. We will search for the λ that give the minimum MSE.

```{r}
#Penalty type (alpha=1 is lasso
#and alpha=0 is the ridge)
  fit <- cv.glmnet(x=X, y=Y,
                        alpha = 1)
 #MSE for several lambdas
 plot(fit)
```



```
print(fit)
```

```
Call:  glmnet::cv.glmnet(x = X, y = Y, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure     SE Nonzero
min 0.0062    58   5.239 0.4132       6
1se 0.3412    15   5.635 0.3624       4
```

All the attributes remain in the model if lambda is set to 0.01 as depicted below. However, the absolute values of the predictors that are now in the model shrink as we increase the lambda. This is depicted below:

```
#The model's coefficients when lambda is set to 0.01
#(λ=0.01)
coef(fit,s=0.01)
```

```
7 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept) 15.8120357462
Price       -0.0569053296
Advertising  0.1233400736
Population  -0.0008269647
Age         -0.0482649088
Income       0.0101796053
Education   -0.0324465331
```

The number changes 6 to 4 attributes if lambda is increased to 0.1. The coefficients for those variables that are removed from the model are shown by a dot. In other words, the coefficient for those variables is zero. Setting the lambda to 0.1 results in 2 coefficients forced to zero where we are left with 4 non-zero coefficients. Also the absolute value of the predictors that remain in the model shrink as we increase the lambda.

This is depicted below:

```
#The model's coefficients when lambda was set to 0.1
#(λ=0.1)
coef(fit,s=0.1)
```

```
7 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept) 14.590306998
Price       -0.052573918
Advertising  0.105366130
Population   .
Age         -0.041822865
Income       0.007643889
Education    .
```

As I increase the lambda to "1", I do not expect more variables to stay in the model to have non-zero coefficients. We now have only 1 variable in the model, resulting in 4 coefficients forced to zero. Also, the absolute value of the predictor that remains in the model shrinks as we increase the lambda. This is depicted below:

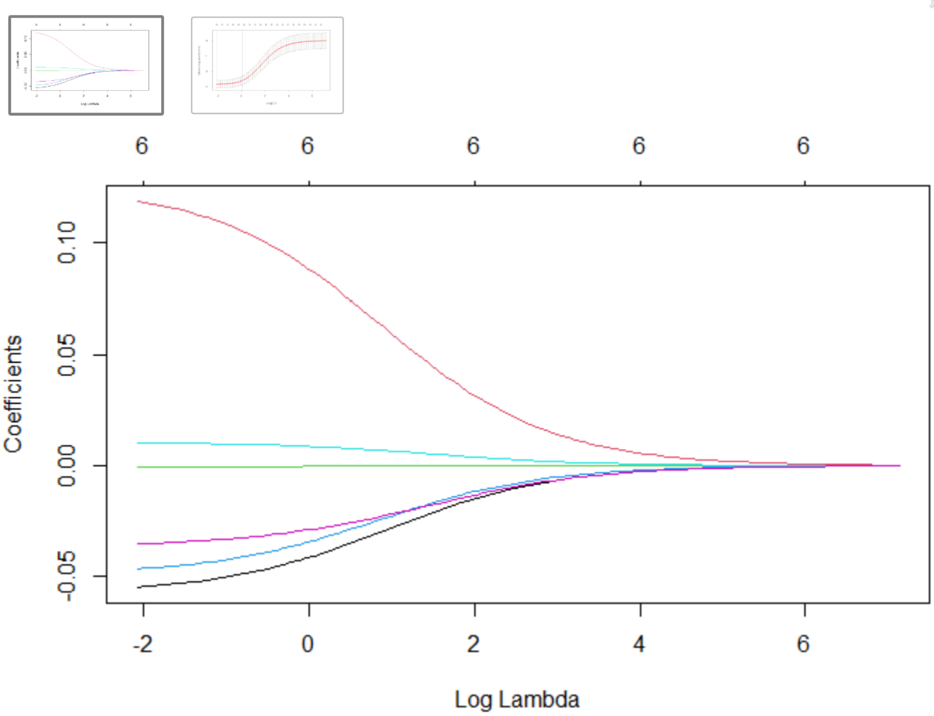
```
#(λ=1)
coef(fit,s=1)
```

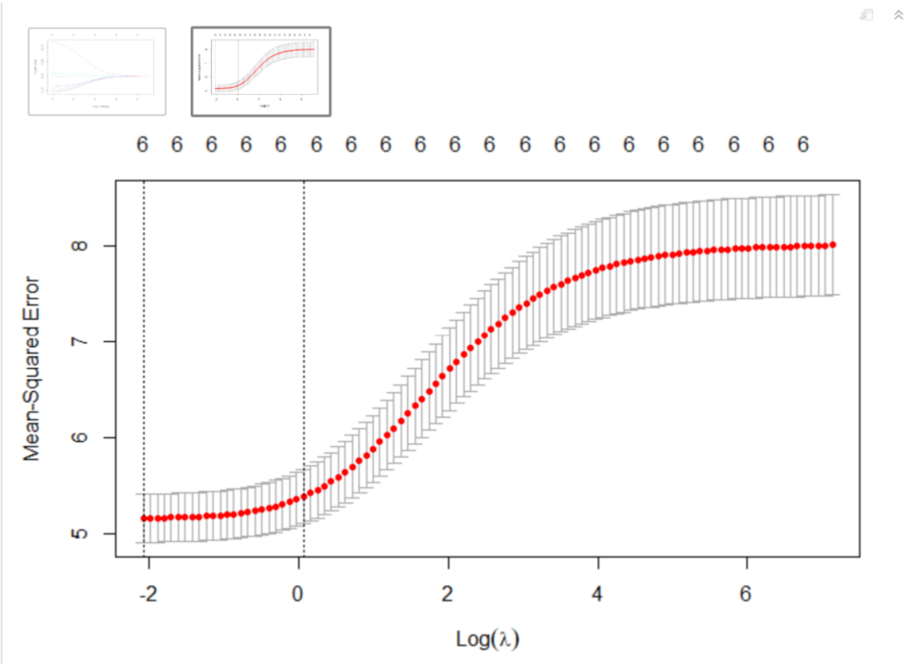
```
7 x 1 sparse Matrix of class "dgCMatrix"
                     s1
(Intercept)  8.74510987
Price       -0.01078445
Advertising  .
Population   .
Age          .
Income       .
Education    .
```

QB4. Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model? 10 points

First, I built a model with alpha set to "0". This means we have built a Ridge regression model as shown here:

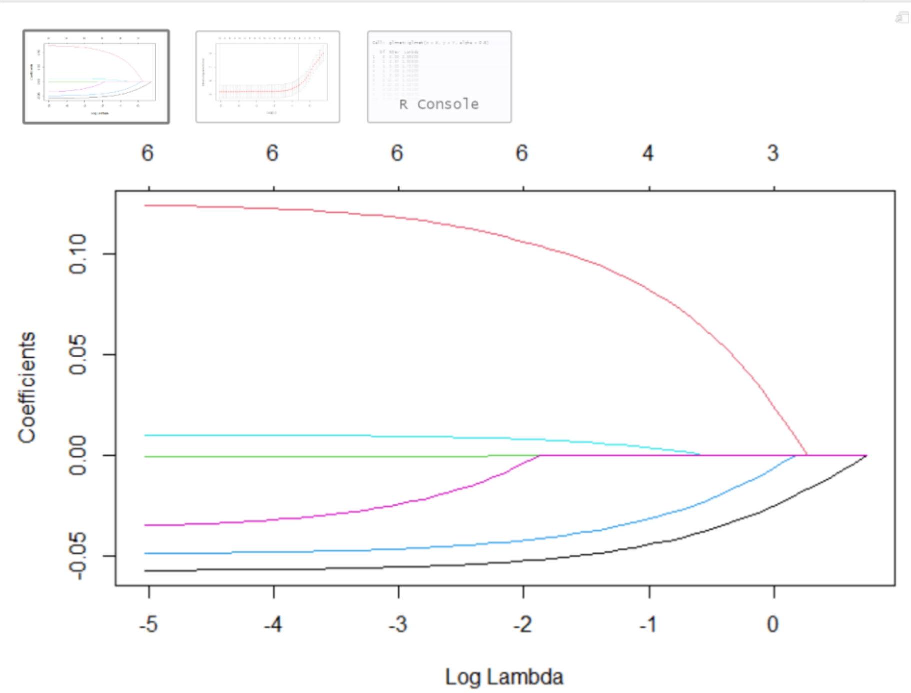
```
#Ridge
fit.ridge<-glmnet(x=X, y=Y, alpha = 0)
plot(fit.ridge, xvar = "lambda")
plot (cv.glmnet (x=X, y=Y,alpha =0))
```

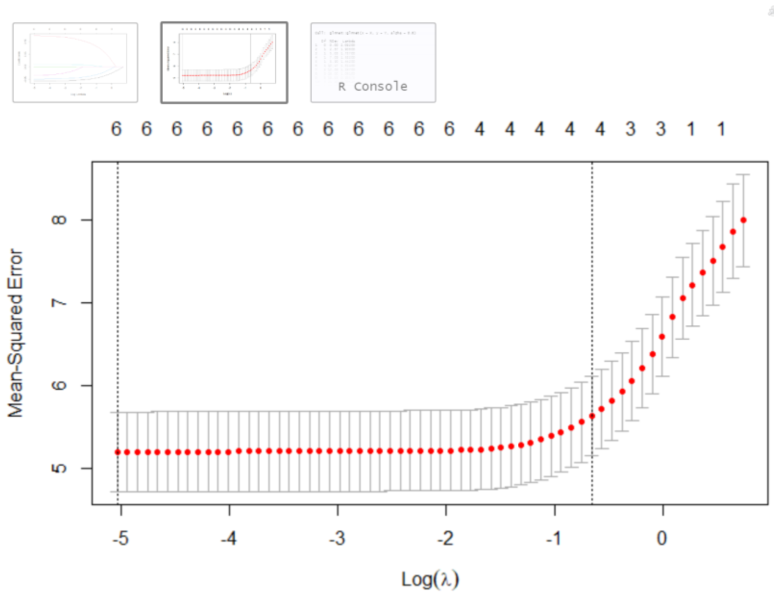




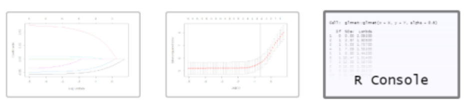
Then, changed the alpha to 0.6 which means we now have an Elastic Net model:

```
#Elastic Net
fit.elnet<-glmnet(x=X, y=Y, alpha = 0.6)
plot(fit.elnet, xvar = "lambda")
plot(cv.glmnet (x=X, y=Y,alpha =0.6))
print(fit.elnet)
```





The best value of lambda for this model is 1.09100 with the %DEV = 16, and Degree of Freedom = 3 as depicted below:




```
Call:  glmnet::glmnet(x = X, y = Y, alpha = 0.6)

   Df  %Dev  Lambda
1   0  0.00 2.09200
2   1  2.67 1.90600
3   1  5.03 1.73700
4   1  7.09 1.58200
5   1  8.90 1.44200
6   1 10.47 1.31400
7   2 12.89 1.19700
8   3 16.00 1.09100
9   3 18.95 0.99370
10  3 21.49 0.90540
11  3 23.67 0.82500
12  3 25.55 0.75170
13  3 27.15 0.68490
14  3 28.52 0.62410
15  4 29.75 0.56860
16  4 30.91 0.51810
```

Reference

DataRobot. (2018). Introduction to Loss Functions. *AI Cloud Platform*. Retrieved from https://www.datarobot.com/blog/introduction-to-loss-functions/

Kumar, S. (2020). Common Loss functions in machine learning for the Classification model. Retrieved from https://medium.com/analytics-vidhya/common-loss-functions-in-machine-learning-for-classification-model-931cbf564d42

Pere, C. (2020). What are Loss Functions? Retrieved from https://towardsdatascience.com/what-is-loss-function-1e2605aeb904

Seif, G. (2019). Understanding the 3 most common loss functions for Machine Learning Regression. Retrieved from https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3