Assignment 1

Arbuda Sivani Majeti

2022-10-31

R Markdown

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

Answer:

Regularization in machine learning prevents the model from overfitting by adding extra information to it. It is a form of regression that shrinks the coefficient estimates towards zero. Complex models usually prone to pick noise from the training data and this might pick the uncertain patterns in the data. Regularization tends to reduce the effect of noise on the predictive performance in the model. Regularization works by adding a pena lty or complexity term or shrinkage term with Residual Sum of Squares to the complex model.There are two types of regularization techniques: *Ridge Regression - Sum of Squared residuals* Lasso Regression - Sum of Absolute Value

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

Answer:

Loss functions play an important role in any statistical model. They define an objective with which the performance of the model is evaluated against and the parameters learned by the model are determined by minimizing a chosen loss function. Loss functions define what a good prediction is and isn't. In short, choosing the right loss function dictates how well your estimator will be.With the help of some optimization function, loss function learns to resude the error in prediction.A loss function maps decisions to their associated costs. Loss functions are not fixed, they change depending on the task in hand and the goal to be met.

Loss function for regression models:

Regression involves predicting a specific value that is continuous in nature. Estimating the price of a house or predicting stock prices are examples of regression because one works towards building a model that would predict a real-valued quantity.

1) Mean Absolute Error
2) Mean Square Error

Loss function for Classification models:

Classification problems involve predicting a discrete class output. It involves dividing the dataset into different and unique classes based on different parameters so that a new and unseen record can be put into one of the classes.

1)Binary Cross Entropy Loss 2)Sparse_categorical_crossentropy

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

Answer: If a model is trained on a smaller dataset it will usually lead to overfitting. When provided with hyperparameters to the model, it will be trained well and will capture all the information in the dataset. This will eventually be a perfect model for single time use. Overfitting the model will lead to the failure of the model by it's low capability to generalize on the test or future unseen data and this will increase the test error. I could say that we cannot trust this model and it is always preferrable to have a large data with not much complex data.

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

Ridge, LASSO work on same principle. They all try to penalize the Beta coefficients so that we can get the important variables (all in case of Ridge and few in case of LASSO). They shrink the beta coefficient towards zero for unimportant variables. These techniques are well being used when we have more numbers of predictors/features than observations. Lambda is the penalty coefficient and it's free to take any allowed number while alpha is selected based on the model you want to try . When Lambda = 0, the penalty is also 0, and so we are just minimizing the sum of the squared residuals. When Lambda asymptotically increase, we arrive to a slope close to 0: so, the larger LAMBDA is, our prediction became less sensitive to the independent variable. We can use Cross-Validation, typically 10-Fold Cross Validation is used in order to determine which LAMBDA give back the lowest VARIANCE. Lambda is the Tuning Parameter that controls the bias-variance tradeoff and we estimate its best value via cross-validation.

__PART B__

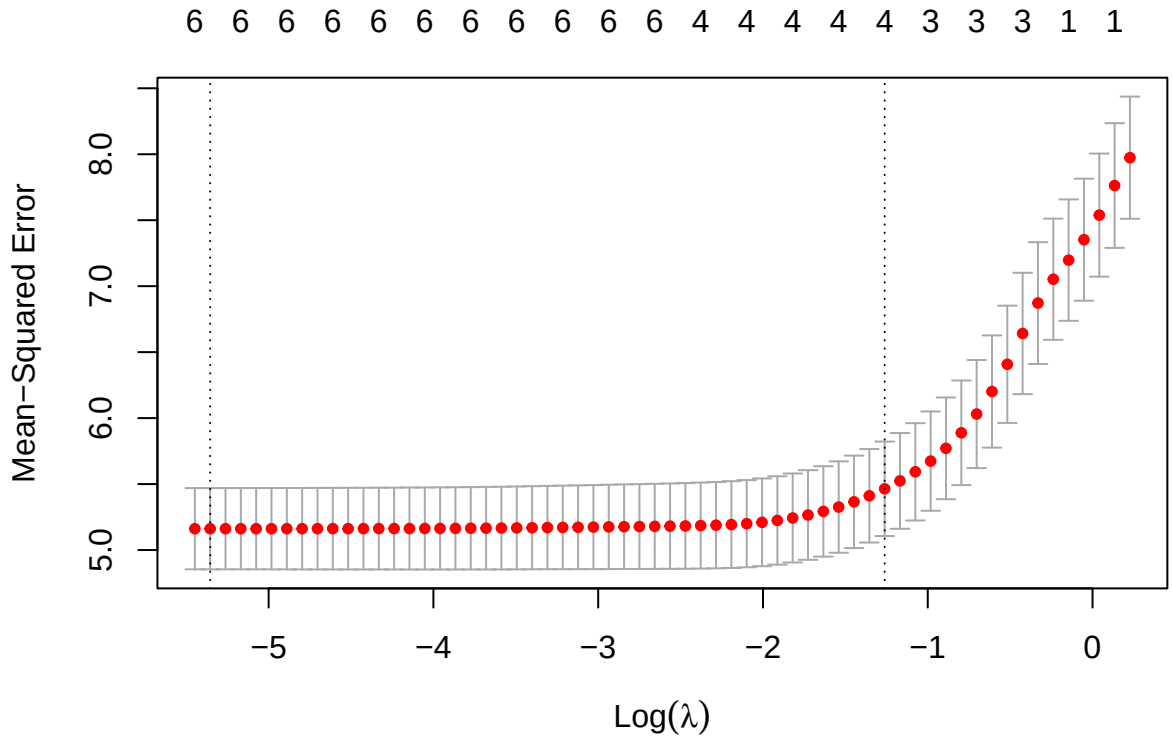
QB1. Build a Lasso regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education"). What is the best value of lambda for such a lasso model? (Hint1: Do not forget to scale your input attributes – you can use the caret preprocess() 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)

```
#loading the library
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)

CS_Dataset <- Carseats %>% select("Sales", "Price", "Advertising","Population","Age","Income","Educatio

#preprocessing the data with center and scale method
CS_Dataset_scaled <- predict(preProcess(CS_Dataset[,-1],method=c("scale","center")),CS_Dataset)

# Assigning the target variable "sales" to y
y<- CS_Dataset_scaled$Sales
# Assigning the rest of the columns as matrix form to x
x<- as.matrix(CS_Dataset_scaled[,-1])

# To get lambda sequence
lambdas<- 10^seq(2,-3,by=-0.1)
```

```
#building lasso regression model with 10 fold cross validation
Lasso<- cv.glmnet(x,y,alpha=1,standardize=TRUE,nfolds = 10)

#Storing the lambda min in a df
best.lambda <- Lasso$lambda.min

best.lambda
```

```
## [1] 0.004725071
```

```
plot(Lasso)
```



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

```
# coefficient for price with optimal lambda
coef(Lasso,s=best.lambda)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  7.49632500
## Price       -1.35335263
## Advertising  0.82747904
## Population  -0.12996793
```

```
## Age         -0.78805992
## Income       0.28899269
## Education   -0.09058271
```

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?

```
#lambda is 0.01
Lambda0.01<- glmnet(x,y,alpha=1,lambda = 0.01)

#lambda is 0.1
Lambda0.1 <- glmnet(x,y, alpha =1,lambda=0.1)

coef(Lambda0.01)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  7.49632500
## Price       -1.34733223
## Advertising  0.82026088
## Population  -0.12187685
## Age         -0.78190633
## Income       0.28488631
## Education   -0.08502707
```

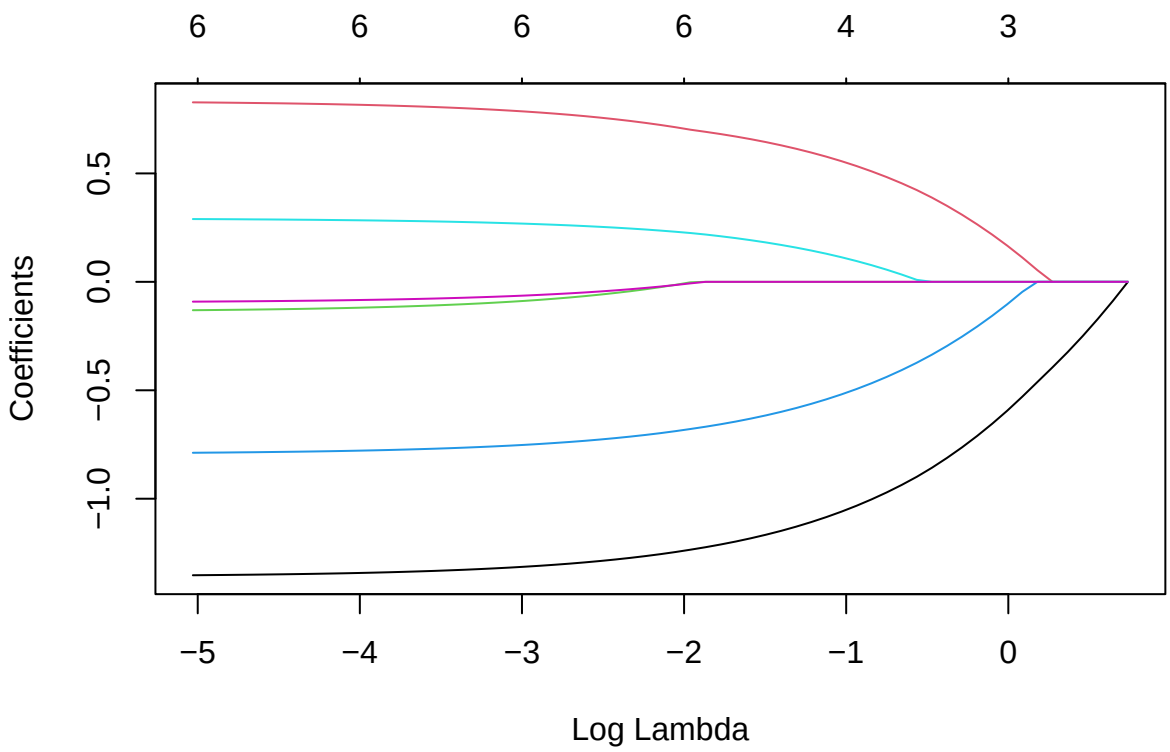
```
#All the coefficients are unchanged when the lambda = 0.01

coef(Lambda0.1)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  7.4963250
## Price       -1.2447745
## Advertising  0.7007230
## Population   .
## Age         -0.6775428
## Income       0.2139222
## Education    .
```

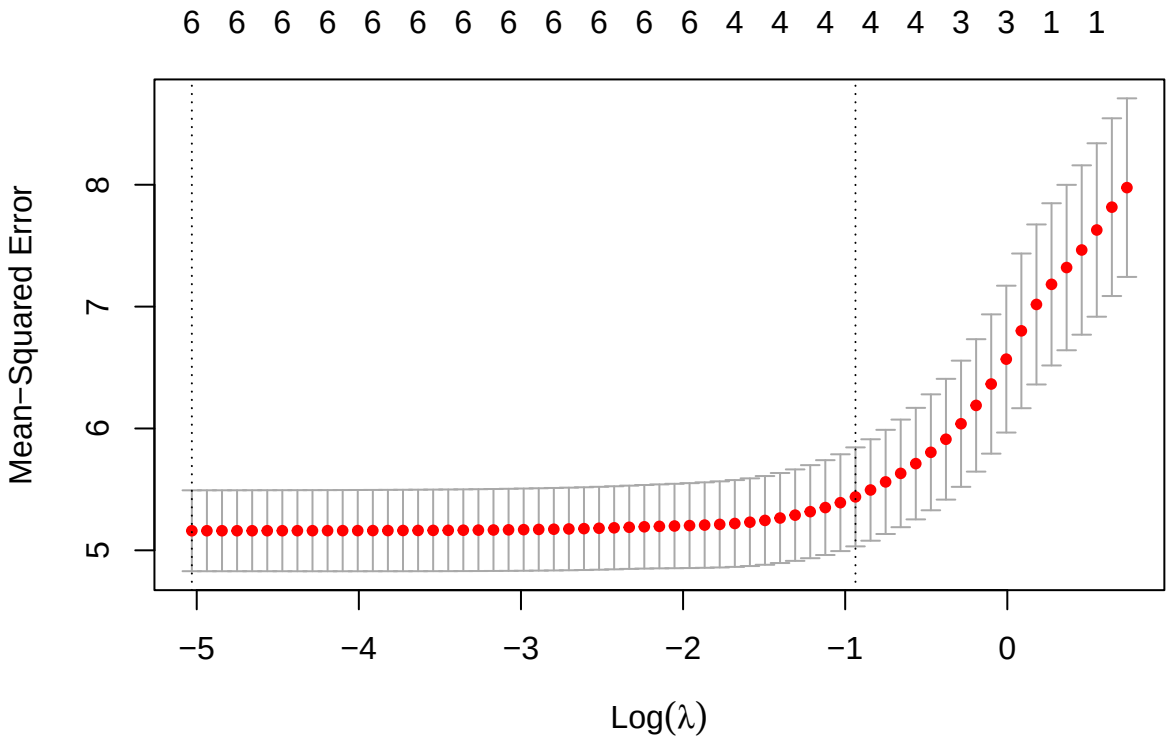
```
#Population and Education have been removed as their coefficients became 0 when lambda = 0.1
```

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

```
elnet <- glmnet(x,y,alpha = 0.6)
plot(elnet, xvar = "lambda")
```




```
plot(cv.glmnet(x,y,alpha=0.6))
```




```
lambda <- cv.glmnet(x,y,alpha=0.6)
# Identifying the best lambda
EL <- lambda$lambda.min
print(paste0("Best lambda :",EL))
```

```
## [1] "Best lambda :0.00653806152801921"
```