Advanced Data Mining and Predictive Analytics

Assignment 1

Shivani Haridas Pitla

2023-02-28

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

Answer: Regularization is primarily used to avoid over-fitting which happens when a model is overly complicated and fits the training data too closely, leading to weak performance when applied to new data. Regularization achieves this by adding a penalty term to the model's objective function that discourages large coefficient values. This penalty term can be adjusted by a hyper-parameter for example lambda in Lasso and Ridge regression that regulates the level of regularization applied to the model.Regularization can improve a model's generalization performance by reducing its variance, at the cost of slightly increasing its bias.

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?

Answer: A loss function in a predictive model serves the purpose of measuring the difference between the output that is predicted and the actual output for a particular input. In other words, it evaluates how accurate the model's predictions are.

There are two typical loss functions in regression models:

1.Mean Squared Error (MSE)

2.Mean Absolute Error (MAE)

Two typical loss functions used in classification models are:

1.Binary Cross-Entropy

2.Categorical Cross-Entropy

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

Answer: It is not advised to fully trust this model in the mentioned case, when a classification model is developed with many hyper-parameters on a small data-set and the training error is incredibly minimal. This is due to a few factors like over-fitting lack of generalization and need for cross validation.To make sure the model is strong and works well with unseen data, it is crucial to use additional evaluation metrics like validation error, test error, and cross-validation. To lessen the likelihood of over-fitting, one could also think about collecting more data or reducing the number of hyper-parameters.

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

Answer: Ridge and LASSO regression are both regularization techniques used to prevent over-fitting and improve the generalization of linear models. They work by adding a penalty term to the linear regression objective function to shrink the magnitude of the coefficient estimates towards zero. This helps to reduce the impact of irrelevant features and avoid over-fitting. Lambda is basically a free parameter which we can control.In L1 and L2 regularization whenever the theta value gets bigger, the error will be bigger and the model will not converge. So essentially here lambda plays an important role of penalizing the higher values of theta. It makes sure that the theta value does not go too high so they will remain very small. The higher value of lambda leads to greater shrinkage of the coefficient estimates towards zero, resulting in a simpler model with less variance and potentially more bias. In LASSO regression, the penalty term is an L1 regularization penalty, which increases the objective function by the absolute magnitude of the coefficient estimations. Some estimations of the coefficients are exactly 0 as a result of this penalty.Whereas in Ridge regression the penalty term is L2 regularization which increases the objective function by the squared size of the coefficient estimates.This penalty results in small but non-zero coefficient.

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 required libraries
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 4.2.2
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.2.2
```

```
## Loaded glmnet 4.1-6
```

```
library("caret")
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
filtered_dataset <- Carseats %>% select("Sales", "Price",
"Advertising","Population","Age","Income","Education")
```

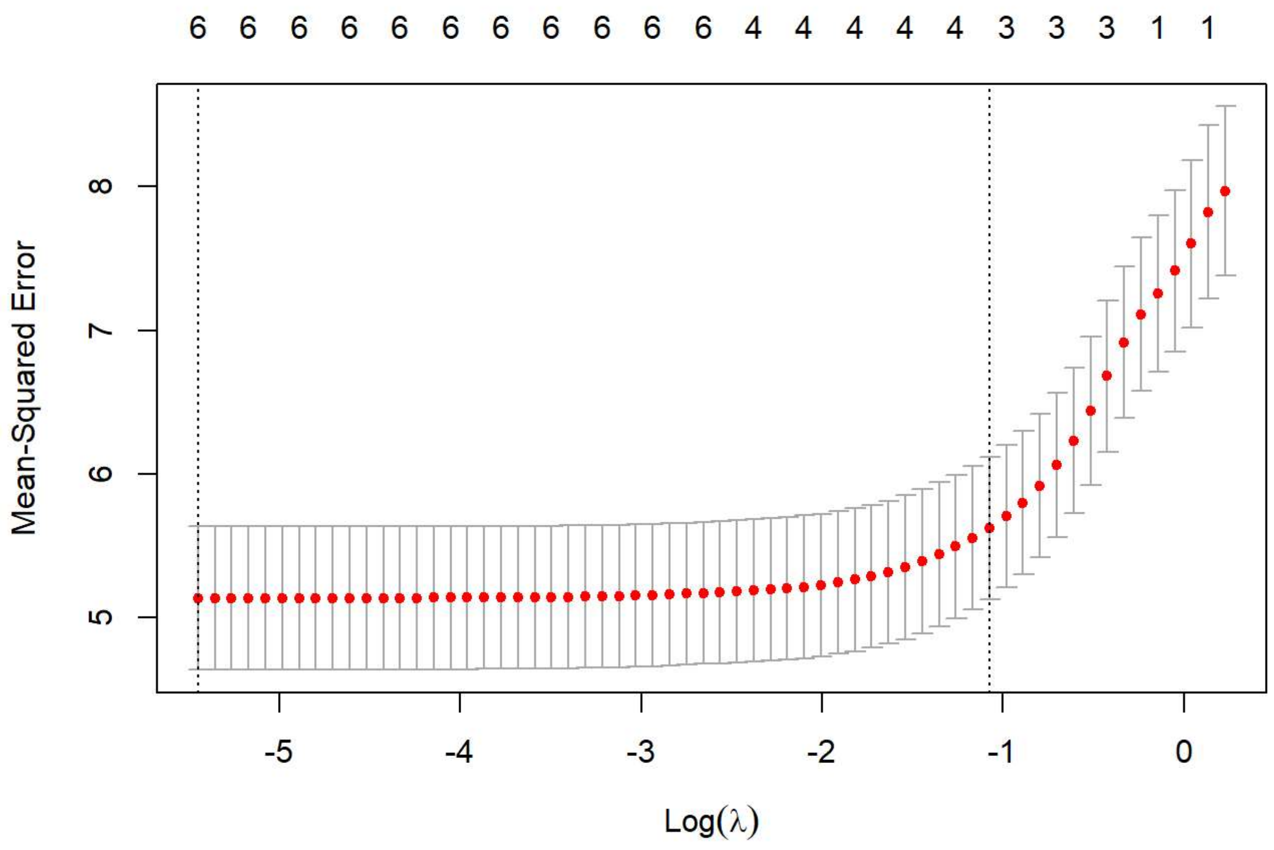
```
#scale and center data
dataset_scaled <- predict(preProcess(filtered_dataset[,-1],method=c("scale","center")),filtered_
dataset)
```

```
# Create design matrix and response vector
y<- dataset_scaled$Sales
x<- as.matrix(dataset_scaled[,-1])
```

```
#creating a sequence of values representing a range of lambda (λ) values
lambdas <- exp(seq(log(100), log(0.001), length.out = 61))
```

```
#Lasso regression with cross-validation using the cv.glmnet() function with 10 fold cross valida
tion
Lasso_regression<- cv.glmnet(x,y,alpha=1,standardize=TRUE,nfolds = 10)
```

```
#Ploting cross-validation error as a function of lambda
plot(Lasso_regression)
```



```
#Finding the lambda that gives minimum cross-validation error
best_lambda<-Lasso_regression$lambda.min
best_lambda
```

```
## [1] 0.004305309
```

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 best lambda
lasso_coef<-coef(Lasso_regression,s=best_lambda)
lasso_coef
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  7.49632500
## Price       -1.35383399
## Advertising  0.82805813
## Population  -0.13061347
## Age         -0.78854992
## Income       0.28931898
## Education   -0.09102484
```

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?

```
#Fitting the Lasso model using lambda = 0.01
lassomodel.01<- glmnet(x,y,alpha=1,lambda = 0.01)
#Finding the number of non-zero coefficients in the model when lambda=0.01
no_attri.01 <- sum(coef(lassomodel.01, s = 0.01) != 0)
no_attri.01
```

```
## [1] 7
```

All the attributes remain in the model when lambda is set to 0.01.

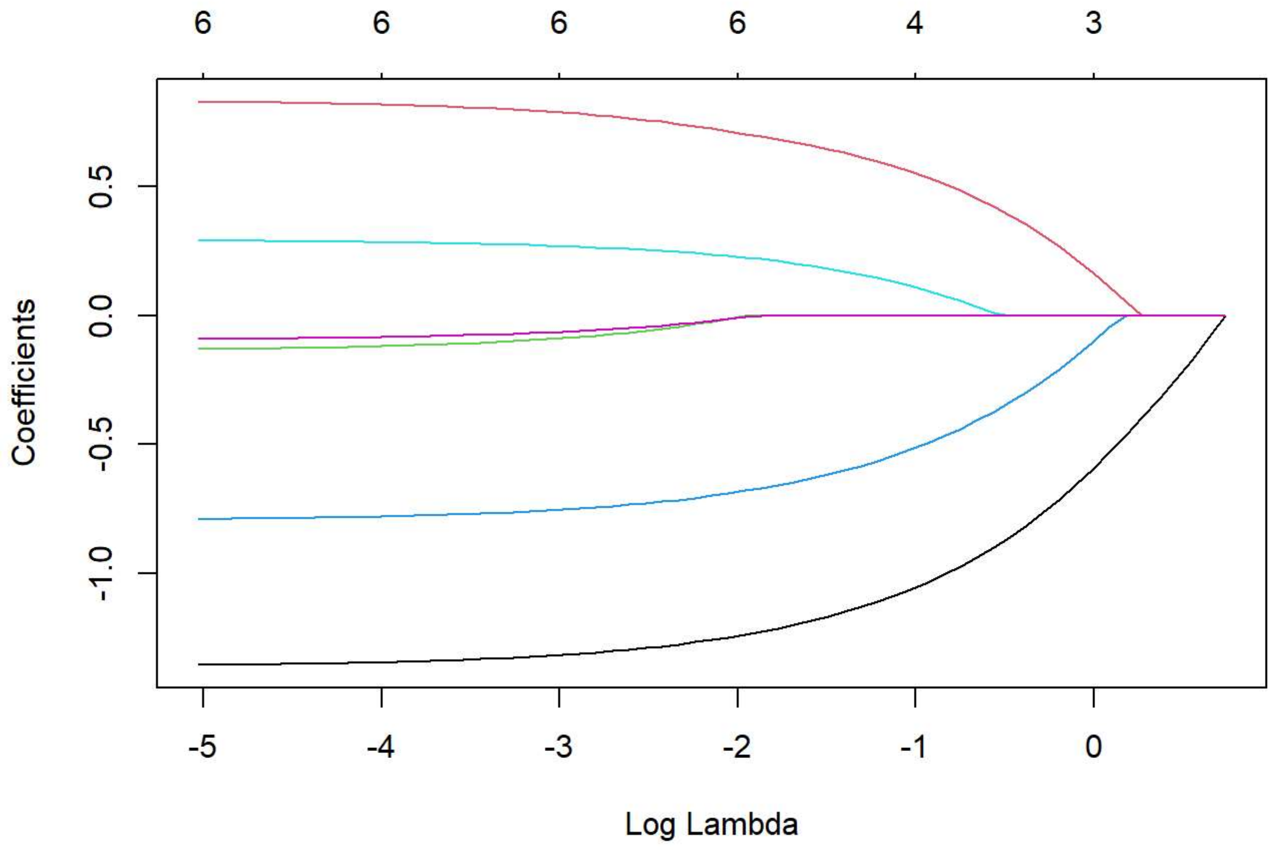
```
#Fitting the Lasso model using lambda = 0.1
lassomodel.1 <- glmnet(x,y, alpha =1,lambda=0.1)
#Finding the number of non-zero coefficients in the model when lambda=0.1
no_attri.1 <- sum(coef(lassomodel.1, s = 0.1) != 0)
no_attri.1
```

```
## [1] 5
```

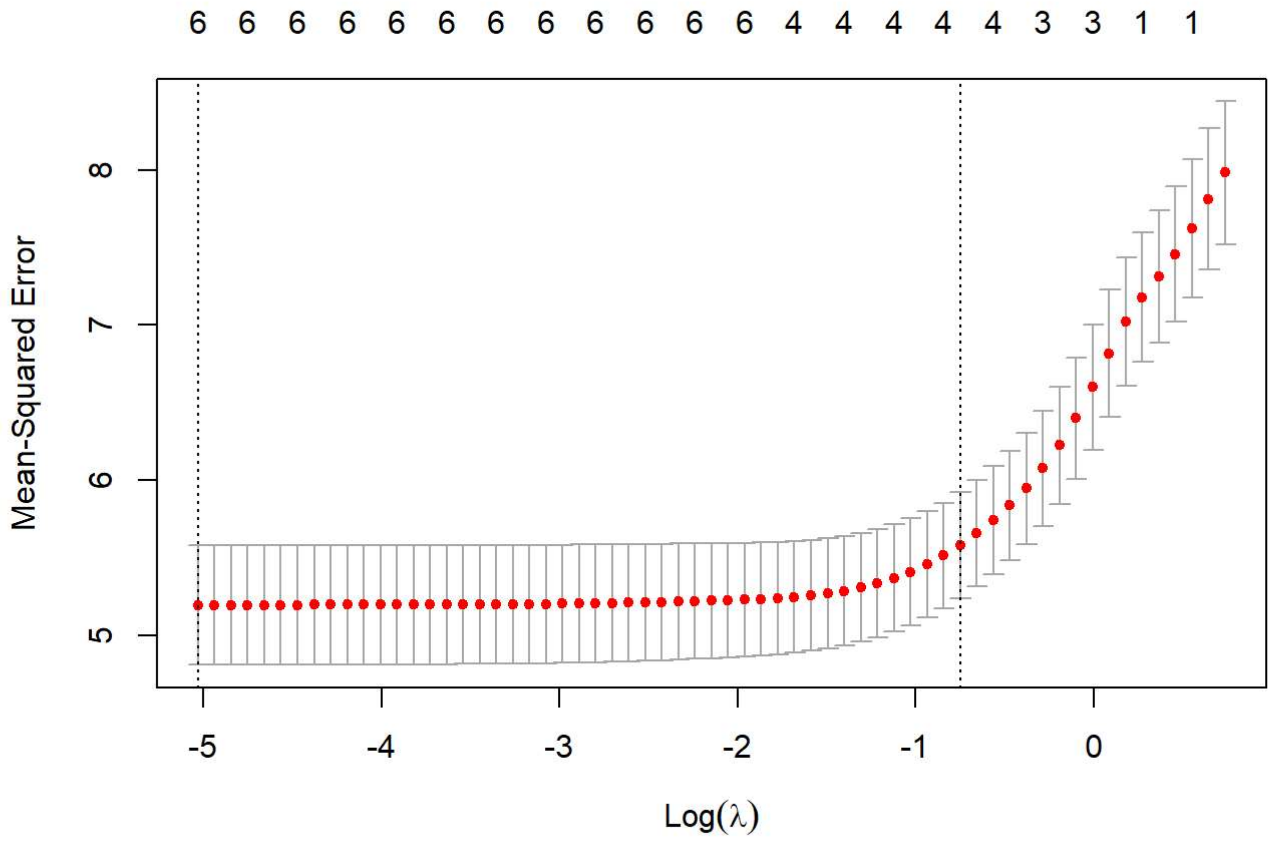
When lambda is increased to 0.1, two of the attributes are removed.

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

```
# Fitting the elastic-net model with alpha = 0.6 using cross-validation
elasticnet.model <- glmnet(x,y,alpha = 0.6)
plot(elasticnet.model, xvar = "lambda")
```



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



```
#Finding the best value of lambda that minimizes the cross-validation error
best_lamda.el <- cv.glmnet(x,y,alpha=0.6)

EL <- best_lamda.el$lambda.min
print(paste0("Best lambda :",EL))
```

```
## [1] "Best lambda :0.00787511825692889"
```