ADVANCED DATA MINING AND PREDICTIVE ANALYTICS

shiva gadila

2023-10-21

Question 1: -

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

The main objective of regularization in data mining, much like in machine learning, is to overcome the issue of overfitting. Overfitting happens when a model turns out to be more complicated and then starts to overfit the training data and performing poorly when faced with new, unseen information. Regularization tackles this problem by trying to introduce a linear model into the objective function being optimized during training.

Regularization is very important in data mining as we deal with the very large and complicated data with many variables and features, Regularisation can lessen the risk of overfitting by reducing the model's level of difficulty and trying to get it to find a simpler solution.

Regularisation is a method for reducing a model's complexity and improving its generalizability. The goal of regularisation is to make a model more efficient. Regularization is used to update machine learning models to lower the adjusted loss function and prevent overfitting or underfitting. Regularization helps our machine learning model adjust precisely to a specific test dataset and lower errors.

Question 2:-

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 job of a loss function in a predictive model is to measure how good or bad the model's predictions are compared to the real data. It helps the model get better by showing it where it's making mistakes and guiding it to make more accurate predictions.

Regression Models:

Mean Squared Error (MSE): MSE is a way to measure how far off our model's predictions are from the actual values. It does this by looking at the average of the squared differences between what our model predicts and what's really true. The smaller the MSE, the closer our model's predictions are to reality.

$$MSE = (1/n) * \Sigma(y - \hat{y})^2$$

where:

- n is the number of data points.
- y represents the actual target values.
- ŷ represents the predicted values.

Mean Absolute Error (MAE): MAE is another way to measure the errors in our model's predictions. It calculates the average of the differences between what our model predicts and what's actually true. Like MSE, the lower the MAE, the better our model's predictions are matching reality.

$$MAE = (1/n) * \Sigma |y - \hat{y}|$$

Root Mean Squared Error (RMSE): RMSE is the square root of MSE. It has the same interpretation as MSE but is in the same units as the target variable.

Formula: RMSE = sqrt(MSE)

Classification models:

Binary Cross-Entropy Loss: This is a loss function for binary classification. It measures how different the predicted probabilities are from the true binary labels (0 or 1). It's used in logistic regression.

$$H(P, Q) = — \text{sum } x \text{ in } X\ P(x) * \log(Q(x))$$

Categorical Cross-Entropy Loss: For multiclass classification, this loss function calculates the differences between predicted class probabilities and actual labels. It's common in tasks with multiple classes, like those involving neural networks. Basically It quantifies the dissimilarity between predicted probability distributions and actual class labels.

$$\text{hinge loss} = [0, 1- yf(x)]$$

Question 3.

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.

Classification model with small dataset and many hyperparameteres showing smalls errors may give some challenges and concerns that may affect the model. You should be cautious about fully trusting that model. The following might be the reasons:

Overfitting: - smalls errors on the small data set means it is an indication of overfitting. So overfitting occurs when the dataset is too closely observed by the model. As a result of this the algorithm cannot perform as accurately against unseen data. Increasing the size of the training dataset can improve the accuracy of the model because it gives the model more chances to figure out the most significant patterns and connections between the input and output data.

Data Noise: Small datasets are more easily influenced by random variations in the data, making them less reliable for training models. To combat data noise in small datasets, clean and pre-process data, select relevant features, and use data augmentation.

When working with a small dataset, the settings of the model's hyperparameters matter a lot. Even tiny adjustments in these settings can cause big changes in how well the model works. This can make it hard to be confident that the chosen hyperparameters are truly the best for making the model work well in different situations.

Question 4 .

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

Role of Lambda :- The lambda parameter in regularized linear models, including Lasso and Ridge regression, plays a important role in preventing overfitting. It acts as a knob for regulating the strength of regularization in the model.

Ridge Regression:-In Ridge regression, the objective function is modified by adding a penalty term to the sum of squared residuals. The magnitude of the coefficients influences this penalty term, and the lambda parameter controls the strength of this penalty.

Lasso Regression:-In Lasso regression, a penalty term proportional to the absolute value of the coefficients is added. Again, the lambda parameter is responsible for controlling the strength of this penalty.

The lambda parameter is like a control knob for the model. It helps find the right balance between two things: bias and variance.

- A bigger lambda means the model leans more on past knowledge (bias), which makes it less flexible but more stable.
- A smaller lambda means the model is more flexible but might make wild guesses ,higher variance.

As a result, the value of lambda must be carefully chosen to achieve the best model performance on new data. This is usually achieved by utilizing techniques such as cross-validation to find the optimum value of lambda that reduces the false positive rate on the validation data.

CODE :-

```
#Load 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)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice
```

Load the Carseats Dataset and Data Preparation

```
# Load the Carseats dataset
attach(Carseats)

# Filter the required attributes, scale the data, and create a model for scal
ing
Carseats_Filter <- Carseats %>% select("Price", "Advertising", "Population",
"Age", "Income", "Education")
scaling_model <- scale(Carseats_Filter)

# Split the data into training and testing sets (x_train, x_test, y_train, y_
test)
set.seed(123)  # Set a random seed for reproducibility
sample_indices <- sample(1:nrow(Carseats_Filter), nrow(Carseats_Filter) * 0.7
)
x_train <- as.matrix(scaling_model[sample_indices, ])
x_test <- as.matrix(scaling_model[-sample_indices, ])
y_train <- Carseats$Sales[sample_indices]
y_test <- Carseats$Sales[-sample_indices]
```

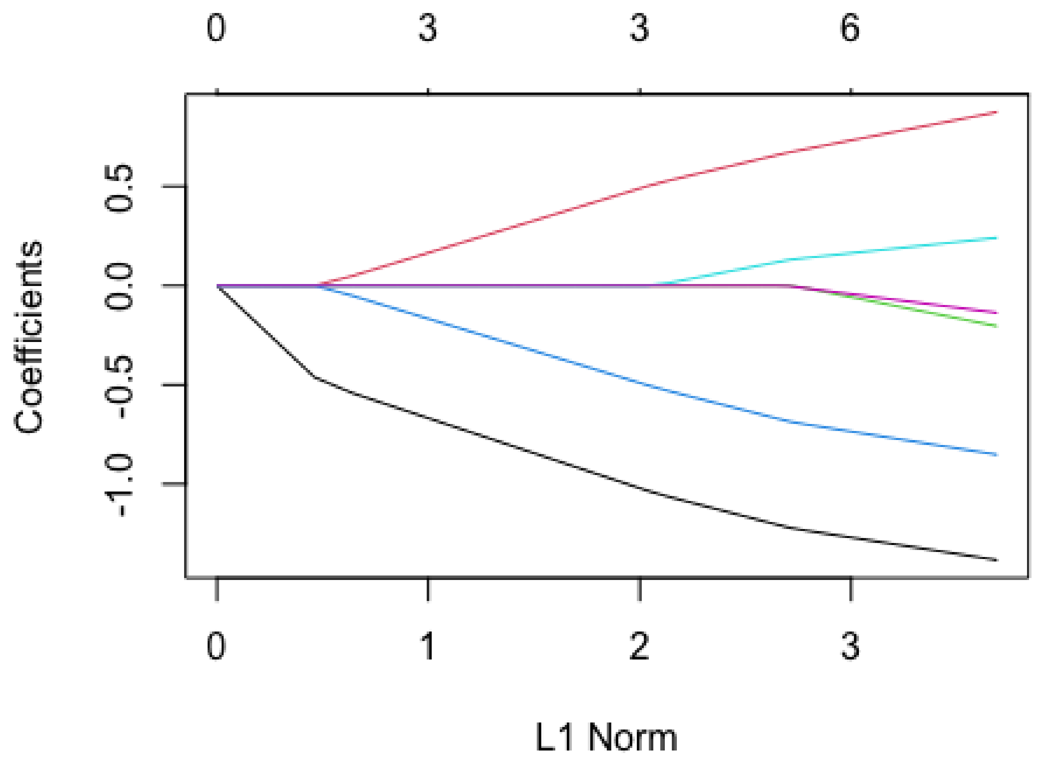
#lasso regression

```
lasso_model <- glmnet(x_train, y_train, alpha = 1)
cv_model <- cv.glmnet(x_train, y_train, alpha = 1)
```

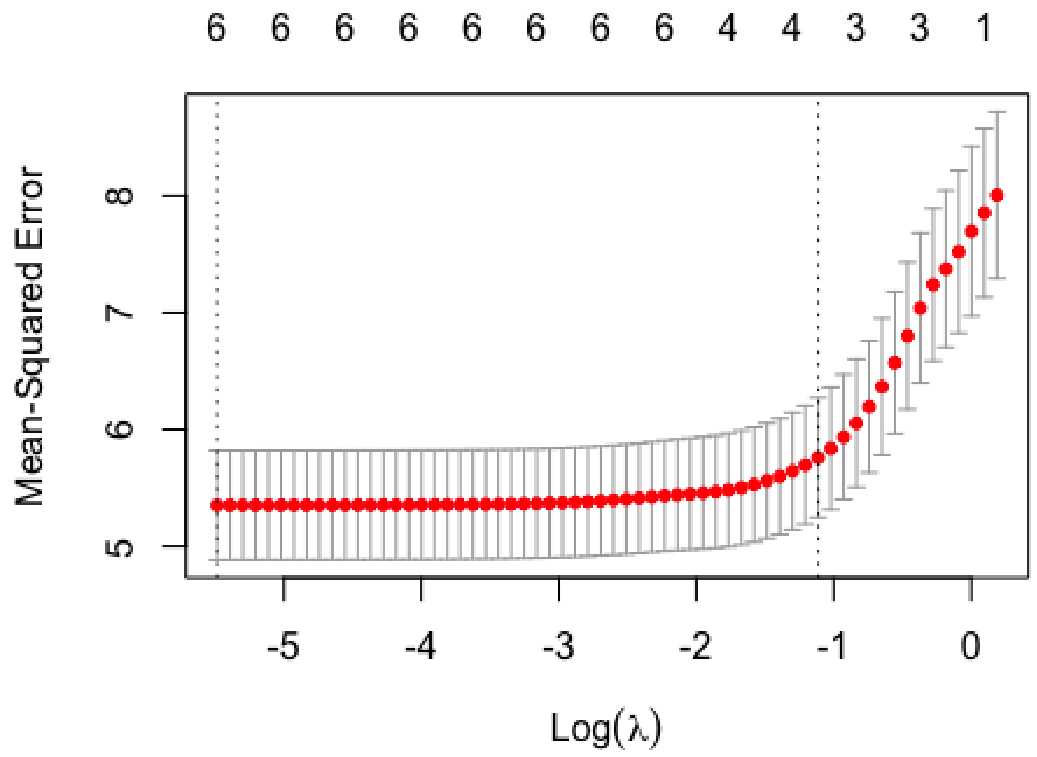
```
best_lambda <- cv_model$lambda.min
lasso_best_model <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda)

# Make predictions on the test data
predictions <- predict(lasso_best_model, s = best_lambda, newx = x_test)
actual_values <- y_test
rsquared <- 1 - sum((actual_values - as.vector(predictions))^2) / sum((actual
_values - mean(actual_values))^2)
rsquared

## [1] 0.3792085

best_lambda

## [1] 0.004143158
```

#Question 1 : Applying the Lasso Regression on the Dataset to check if it improved the R-squared value.

```
fit <- glmnet(x_train, y_train)
summary(fit)

##           Length Class     Mode
## a0         62    -none-    numeric
## beta      372    dgCMatrix S4
## df         62    -none-    numeric
## dim         2    -none-    numeric
## lambda     62    -none-    numeric
## dev.ratio  62    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        3    -none-    call
## nobs        1    -none-    numeric

plot(fit)
```



```r
# Lasso Regression with Cross-Validation
fit_lasso <- cv.glmnet(x_train, y_train, alpha = 1)  # alpha = 1 for Lasso
plot(fit_lasso)
```



```
summary(fit_lasso)

##             Length Class  Mode
## lambda      62     -none- numeric
## cvm         62     -none- numeric
## cvsd        62     -none- numeric
## cvup        62     -none- numeric
## cvlo        62     -none- numeric
## nzero       62     -none- numeric
## call         4     -none- call
## name         1     -none- character
## glmnet.fit 12      elnet  list
## lambda.min  1      -none- numeric
## lambda.1se  1      -none- numeric
## index       2      -none- numeric

print(fit_lasso)  # Print the cross-validated model

##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.0041    62   5.352 0.4683       6
## 1se 0.3283    15   5.760 0.5136       3
```

#Question 2 : What is the coefficient for the price (normalized) attribute in the best model (i.e. modelwith the optimal lambda)?

```
# Find the minimum Lambda value
min_lambda <- fit_lasso$lambda.min
min_lambda

## [1] 0.004143158

# Coefficient for the equation using the minimum Lambda
coef(fit_lasso, s = min_lambda)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  7.4482646
## Price       -1.3825897
## Advertising  0.8741696
## Population  -0.2042096
## Age         -0.8516510
## Income       0.2410709
## Education   -0.1369582

# Get the coefficients for the optimal Lasso model (at lambda.min)
coefficients_optimal_model <- coef(fit_lasso, s = min_lambda)

# Extract the coefficient for the "Price" attribute (it's the second coeffici
ent)
price_coefficient <- coefficients_optimal_model[2]

# Print the coefficient
price_coefficient

## [1] -1.38259
```

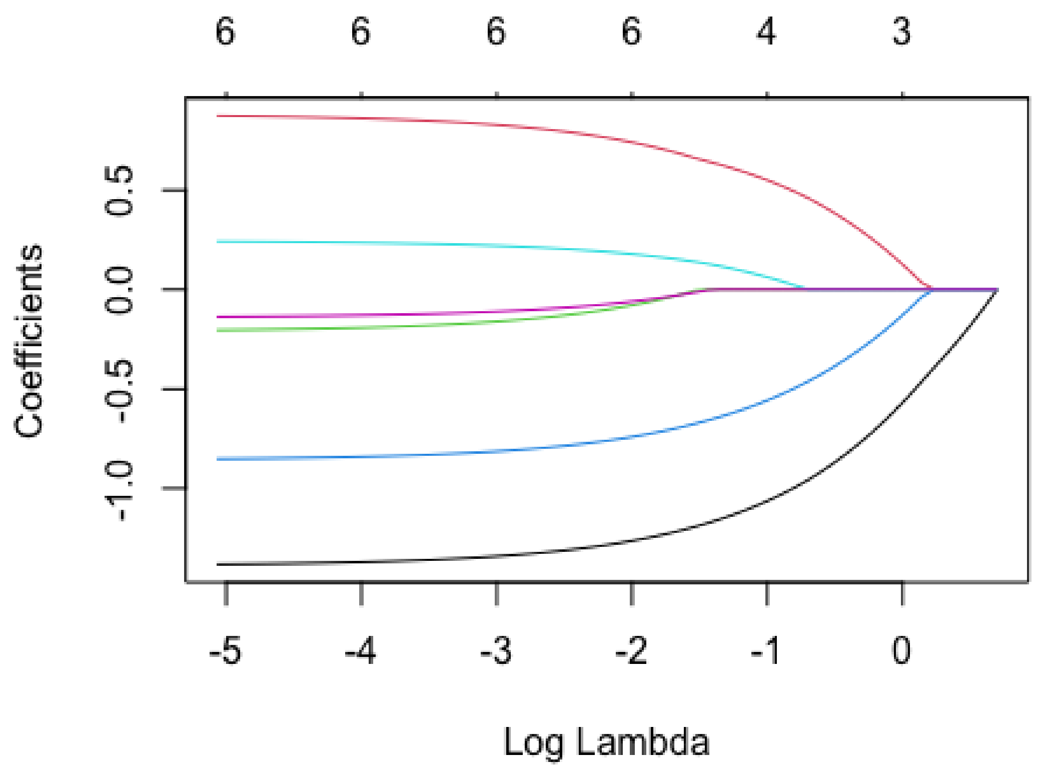
#Question 3 How many attributes remain in the model if lambda is set to 0.01? How that number changes if lambda is increased to 0.1? Do you expect more variables to stay in the model (i.e., to have non-zero coefficients) as we increase lambda.

```
# Find the coefficients for lambda = 0.01
coefficients_lambda_0.01 <- coef(fit_lasso, s = 0.01)

# Find the coefficients for lambda = 0.1
coefficients_lambda_0.1 <- coef(fit_lasso, s = 0.1)

num_non_zero_coefficients_lambda_0.01 <- sum(coefficients_lambda_0.01 != 0)
num_non_zero_coefficients_lambda_0.1 <- sum(coefficients_lambda_0.1 != 0)
```

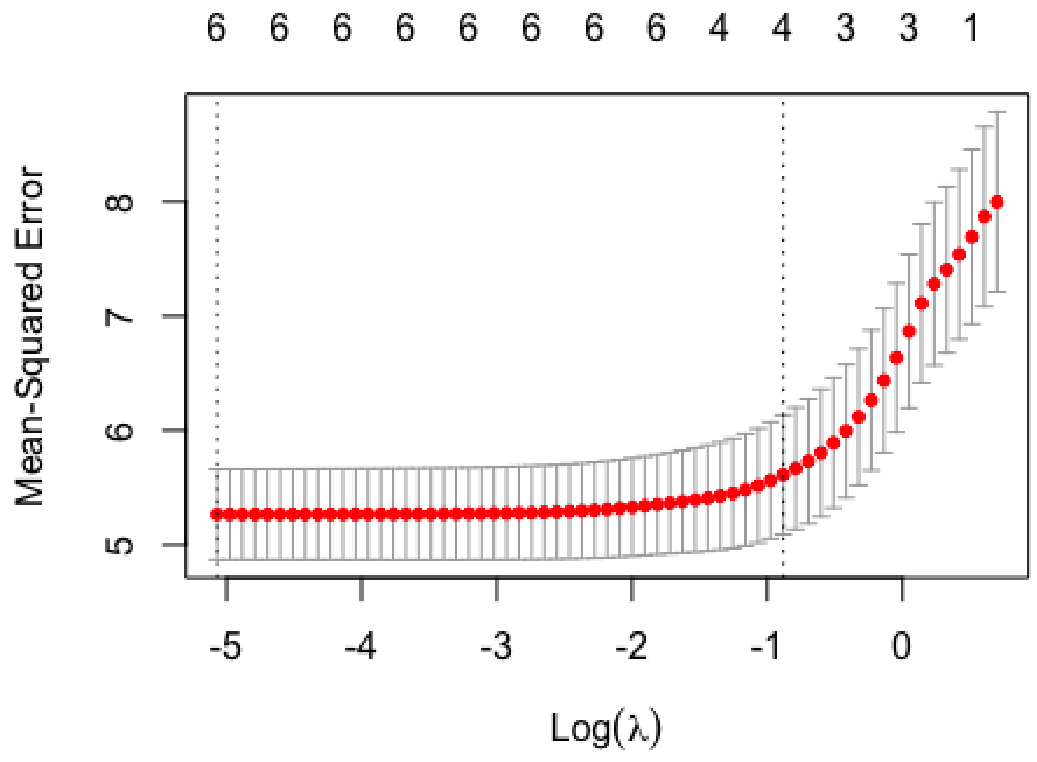
```
#results
num_non_zero_coefficients_lambda_0.01

## [1] 7

num_non_zero_coefficients_lambda_0.1

## [1] 7
```

#Question 4 QB4. Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model?

```
fit.elnet = glmnet(x_train,y_train, alpha = 0.6)
plot(fit.elnet, xvar = "lambda")
```



```
# Elastic-Net Model with Cross-Validation (alpha = 0.6)
fit_elnet <- cv.glmnet(x_train, y_train, alpha = 0.6)
plot(fit_elnet)
```



```
summary(fit_elnet)
```

```
##              Length Class  Mode
## lambda       63     -none- numeric
## cvm          63     -none- numeric
## cvsd         63     -none- numeric
## cvup         63     -none- numeric
## cvlo         63     -none- numeric
## nzero        63     -none- numeric
## call          4     -none- call
## name          1     -none- character
## glmnet.fit   12     elnet  list
## lambda.min    1     -none- numeric
## lambda.1se    1     -none- numeric
## index         2     -none- numeric
```

```
print(fit_elnet)
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 0.6)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.0063    63   5.266 0.3960       6
## 1se 0.4140    18   5.613 0.5214       4
```

```r
# Find the minimum Lambda value for the elastic-net model
min_lambda_elnet <- fit_elnet$lambda.min
min_lambda_elnet
```

```
## [1] 0.006291819
```

#The best value of lambda for such a model is 0.006291819