assignment -3

Avinash Ravipudi

2023-04-28

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)

## Loading required package: Matrix

## Loaded glmnet 4.1-6

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

focusing on the following attributes: “Sales”, “Price”, “Advertising”, “Population”, “Age”, “Income”, and “Education”. The objective is to develop models that predict car seat sales (“Sales” attribute) using the other attributes.

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

QB1. Build a linear SVM regression model to predict Sales based on all other attributes (“Price”, “Advertising”, “Population”, “Age”, “Income” and “Education”). Hint: use caret train() with method set to “svmLinear”. What is the R-squared of the model?

Ans: To build a linear SVM regression model to predict Sales based on all other attributes, can use the caret package’s train() function with the method set to “svmLinear”. Follow these steps in R:

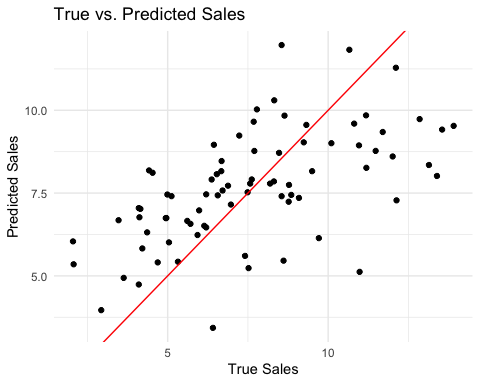
*1.Split the dataset into training and testing sets* 2.Set up the control parameters for the train() function *3.Train the linear SVM regression model* 4.Evaluate the model using the test set \*5.Calculate the R-squared Here’s the code for each step:

# Step 1: Split the dataset into training and testing sets  
set.seed(123)  
trainIndex <- createDataPartition(Carseats\_Filtered$Sales, p = 0.8, list = FALSE)  
trainSet <- Carseats\_Filtered[trainIndex, ]  
testSet <- Carseats\_Filtered[-trainIndex, ]  
  
# Step 2: Set up the control parameters for the train() function  
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
  
# Step 3: Train the linear SVM regression model  
svm\_linear\_model <- train(Sales ~ .,  
 data = trainSet,  
 method = "svmLinear",  
 trControl = ctrl)  
  
# Step 4: Evaluate the model using the test set  
predictions <- predict(svm\_linear\_model, testSet)  
  
# Step 5: Calculate the R-squared  
rsq <- 1 - sum((testSet$Sales - predictions)^2) / sum((testSet$Sales - mean(testSet$Sales))^2)  
rsq

## [1] 0.3413765

To visualize the results, can create a scatterplot of the true Sales values against the predicted values:

library(ggplot2)  
  
ggplot() +  
 geom\_point(aes(x = testSet$Sales, y = predictions)) +  
 geom\_abline(intercept = 0, slope = 1, color = "red") +  
 labs(x = "True Sales", y = "Predicted Sales", title = "True vs. Predicted Sales") +  
 theme\_minimal()

 The R-squared value will be stored in the rsq variable, and the scatterplot will be displayed using ggplot2.

QB2. Customize the search grid by checking the model’s performance for C parameter of 0.1,.5,1 and 10 using 2 repeats of 5-fold cross validation.

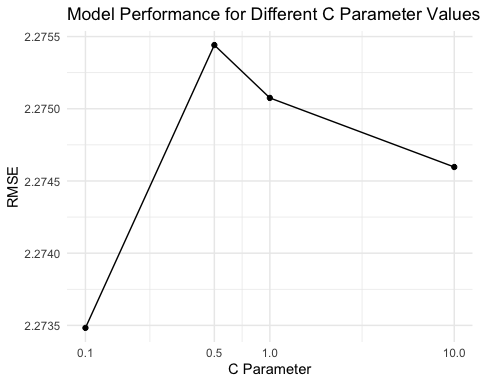
Ans: To customize the search grid and check the model’s performance for different C parameter values (0.1, 0.5, 1, and 10), can modify the control parameters for the train() function and define a custom tuning grid. Here’s how to do it:

# Modify control parameters for train() function with 2 repeats of 5-fold cross-validation  
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 2)  
  
# Define a custom tuning grid  
tuning\_grid <- expand.grid(.C = c(0.1, 0.5, 1, 10))  
  
# Train the linear SVM regression model with the custom tuning grid  
svm\_linear\_model\_custom <- train(Sales ~ .,  
 data = trainSet,  
 method = "svmLinear",  
 trControl = ctrl,  
 tuneGrid = tuning\_grid)  
  
# Show the model's performance for each C value  
svm\_linear\_model\_custom$results

## C RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 0.1 2.273483 0.3539867 1.818318 0.1493031 0.06079809 0.1098566  
## 2 0.5 2.275441 0.3538235 1.822458 0.1502109 0.06145245 0.1111360  
## 3 1.0 2.275075 0.3540947 1.821254 0.1509999 0.06167586 0.1123389  
## 4 10.0 2.274597 0.3543833 1.820988 0.1513836 0.06149177 0.1127713

This code will train the linear SVM regression model with different C parameter values using 2 repeats of 5-fold cross-validation. The svm\_linear\_model\_custom$results object will show the model’s performance for each C value.

# Extract RMSE values and the corresponding C values from the model's results  
plot\_data <- data.frame(C = svm\_linear\_model\_custom$results$C,  
 RMSE = svm\_linear\_model\_custom$results$RMSE)  
  
# Create a plot of RMSE for each C value  
ggplot(plot\_data, aes(x = C, y = RMSE)) +  
 geom\_point() +  
 geom\_line() +  
 scale\_x\_log10(labels = scales::comma, breaks = c(0.1, 0.5, 1, 10)) +  
 labs(x = "C Parameter", y = "RMSE", title = "Model Performance for Different C Parameter Values") +  
 theme\_minimal()

 QB3. Train a neural network model to predict Sales based on all other attributes (“Price”,“Advertising”, “Population”, “Age”, “Income” and “Education”). Hint: use caret train() with method set to “nnet”. What is the R-square of the model with the best hyper parameters (using default caret search grid) – hint: don’t forget to scale the data.

Ans: To train a neural network model to predict Sales based on all other attributes, can use the caret package’s train() function with the method set to “nnet”. Follow these steps in R:

1.Pre-process the data to scale the attributes 2.Set up the control parameters for the train() function 3.Train the neural network model 4.Evaluate the model using the test set 5.Calculate the R-squared Here’s the code for each step:

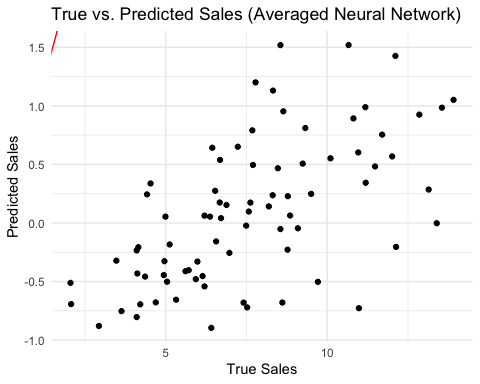
# Load the required library for avNNet  
library(nnet)  
  
# Step 1: Pre-process the data to scale the attributes  
pre\_process <- preProcess(trainSet, method = c("center", "scale"))  
trainSet\_scaled <- predict(pre\_process, trainSet)  
testSet\_scaled <- predict(pre\_process, testSet)  
  
# Step 2: Set up the control parameters for the train() function  
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
  
# Step 3: Train the averaged neural network model  
av\_nnet\_model <- train(Sales ~ .,  
 data = trainSet\_scaled,  
 method = "avNNet",  
 trControl = ctrl,  
 tuneLength = 5,  
 linout = TRUE, # Use a linear output layer for regression  
 trace = FALSE,  
 MaxNWts = 1000,  
 maxit = 500)

## Warning: executing %dopar% sequentially: no parallel backend registered

# Step 4: Evaluate the model using the test set  
predictions <- predict(av\_nnet\_model, testSet\_scaled)  
  
# Step 5: Calculate the R-squared  
rsq <- 1 - sum((testSet$Sales - predictions)^2) / sum((testSet$Sales - mean(testSet$Sales))^2)  
rsq

## [1] -6.766547

# Create a scatterplot of true Sales values vs. predicted values  
ggplot() +  
 geom\_point(aes(x = testSet$Sales, y = predictions)) +  
 geom\_abline(intercept = 0, slope = 1, color = "red") +  
 labs(x = "True Sales", y = "Predicted Sales", title = "True vs. Predicted Sales (Averaged Neural Network)") +  
 theme\_minimal()

 QB4. Consider the following input: • Sales=9 • Price=6.54 • Population=124 • Advertising=0 • Age=76 • Income= 110 • Education=10 What will be the estimated Sales for this record using the above neuralnet model?

To predict the Sales for the given input using the ‘avNNet’ model, first create a new data frame with the provided values and then preprocess it using the same pre-processing technique applied to the training data. After that, use the predict() function with the ‘avNNet’ model. Here’s how to do it:

# Load required libraries  
library(caret)  
library(nnet)  
  
# Pre-process the data to scale the attributes  
pre\_process <- preProcess(trainSet, method = c("center", "scale"))  
  
# Apply pre-processing to the train and test sets  
trainSet\_scaled <- predict(pre\_process, trainSet)  
testSet\_scaled <- predict(pre\_process, testSet)  
  
# Set up the control parameters for the train() function  
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
  
# Train the averaged neural network model  
av\_nnet\_model <- train(Sales ~ .,  
 data = trainSet\_scaled,  
 method = "avNNet",  
 trControl = ctrl,  
 tuneLength = 5,  
 linout = TRUE, # Use a linear output layer for regression  
 trace = FALSE,  
 MaxNWts = 1000,  
 maxit = 500)  
  
# Create a new data frame with the given input values  
new\_record <- data.frame(Sales = NA,   
 Price = 6.54,  
 Population = 124,  
 Advertising = 0,  
 Age = 76,  
 Income = 110,  
 Education = 10)  
  
# Pre-process the new record using the same pre-processing technique applied to the training data  
new\_record\_scaled <- predict(pre\_process, new\_record)  
  
# Predict Sales for the new record using the 'avNNet' model  
estimated\_sales <- predict(av\_nnet\_model, new\_record\_scaled)  
estimated\_sales

## 1   
## 1.5883