# Applied Statistical Modeling in R

Regression Models and Cross Validation

Author: Shivani Battu

December 10, 2023

## Construction of Logistic Regression Model

### (a) Load the package alr4 into memory

To load the package into the memory, I am using an if statement to check if the package exists in R already and load it. If the package exists in R, a statement will be returned indicating the successful loading of the package *"alr4"*.

```
library(alr4)
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
## Loading required package: effects
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
if (requireNamespace("alr4", quietly = TRUE)) {
  cat("Alr4 loaded successfully")
}
```

```
## Alr4 loaded successfully
```

**(b) From the dataset Downer, construct a dataframe using the variables calving, daysrec, ck, ast, urea, and pcv. Remove any rows with missing data.**

From the *'Downer'* dataset, we are creating a new data frame *'b_df'* and omitting the null values from the final data frame. We also calculate the count of omitted rows to track the number of rows with missing data.

```r
b_df<-alr4::Downer[, c("calving", "daysrec", "ck", "ast", "urea", "pcv")]
b_df_new<-na.omit(b_df)
```

The number of rows in original dataframe are : 435and in the latest dataframe are : 165 The number of rows which had missing data are : 270

**(c) Construct a logistic regression model with explanatory variables given in (1b) and outcome as the response. Use an 80/20 split of the data.**

To construct a logistic regression model with explanatory variables as (calving, daysrec, ck, ast, urea, and pcv) and response variable as 'Outcome' we first split the data into 80/20 range for training and testing respectively.

Nextly, we create a training set and testing set from the dataframe based on the split. Once the sets are ready, we build a logistic regression model using the **'glm()'** function

```r
# Set seed for reproducibility
set.seed(123)
c_df<-alr4::Downer[, c("calving", "daysrec", "ck", "ast", "urea", "pcv","outcome")]
c_df<-na.omit(c_df)

# Splitting the data into training and testing sets
c_train_index <- sample(1:nrow(c_df), floor(0.8 * nrow(c_df)))

#train_index <- sample(1:nrow(c_df), replace=TRUE, prob=c(0.8,0.2))
c_train_set <- c_df[c_train_index, ]
c_test_set <- c_df[-c_train_index, ]

# Define the formula and build the model
c_log_reg_model <- glm(outcome ~.-outcome, data = c_train_set, family = binomial())
cat("The summary of the model is as follows \n")
```

```
## The summary of the model is as follows
```

```r
summary(c_log_reg_model)
```

```
##
## Call:
## glm(formula = outcome ~ . - outcome, family = binomial(), data = c_train_set)
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.510e-01  1.271e+00  -0.119 0.905450
## calvingafter  1.212e+00  6.451e-01   1.879 0.060217 .
## daysrec      -4.537e-01  1.655e-01  -2.741 0.006119 **
## ck           -8.585e-05  1.249e-04  -0.688 0.491719
## ast          -2.706e-03  1.630e-03  -1.660 0.096886 .
## urea         -3.049e-01  8.771e-02  -3.476 0.000508 ***
## pcv           7.310e-02  3.670e-02   1.992 0.046413 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 171.90  on 131  degrees of freedom
## Residual deviance: 122.31  on 125  degrees of freedom
## AIC: 136.31
##
## Number of Fisher Scoring iterations: 7
```

### (d) Construct a confusion matrix. How accurate is your model?

Based on the testing set, we compute the predicted values for the Response variable i.e "Outcome-Survived/Died".

If the predicted values are greater than 0.5, we predict that the outcome is 'Survied'; alternatively if the predicted value is less than 0.5, we predict the outcome as 'Died'.

Nextly we create a confusion matrix based on the actual and predicted values to evaluate the model accuracy.

```r
library(cutpointr)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:cutpointr':
##
##     precision, recall, sensitivity, specificity
```

```r
# Predict the response for the test data
pred_values <- predict(c_log_reg_model, newdata = c_test_set, type = "response")
predicted_set <- ifelse(pred_values > 0.5, "survived", "died")
actual_set <- c_test_set$outcome

#confusion matrix
d_confusion_matrix <- table(Actual = actual_set, Predicted = predicted_set)

# Calculate accuracy
d_mod_accuracy <- sum(diag(d_confusion_matrix)) / sum(d_confusion_matrix)

# Print the confusion matrix and accuracy
print(d_confusion_matrix)
```

```
##           Predicted
## Actual     died survived
##   died       21        2
##   survived    4        6
```

The Accuracy of the model is : 81.8182 %.

Based on the accuracy we can infer that the model correctly predicts the outcome for 81.8182 % of the data points which is a decent prediction.

---

## Construction of Lasso Regression Model

**The *Boston*, from the MASS library, contains information about different suburbs in Boston and various factors that might influence house prices. Your objective is to predict house prices (medv) based on other available features and identify the most influential predictors using Lasso regression.**

### (a) Loading the Dataset: Load the Boston dataset from the MASS package.

To load the dataset into the memory, I am using the data() function. Once the data set is loaded, a statement will be returned indicating the successful loading of the package *"Boston"*.

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

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

```
## Loaded glmnet 4.1-8
```

```
## 'Boston' dataset loaded successfully
```

### (b) Lasso Regression Modeling:

- **Use Lasso regression (glmnet package) to build a predictive model for house prices.**
- **Tune the model using cross-validation (cv.glmnet function) to find the optimal lambda (regularization parameter) for Lasso regression.**

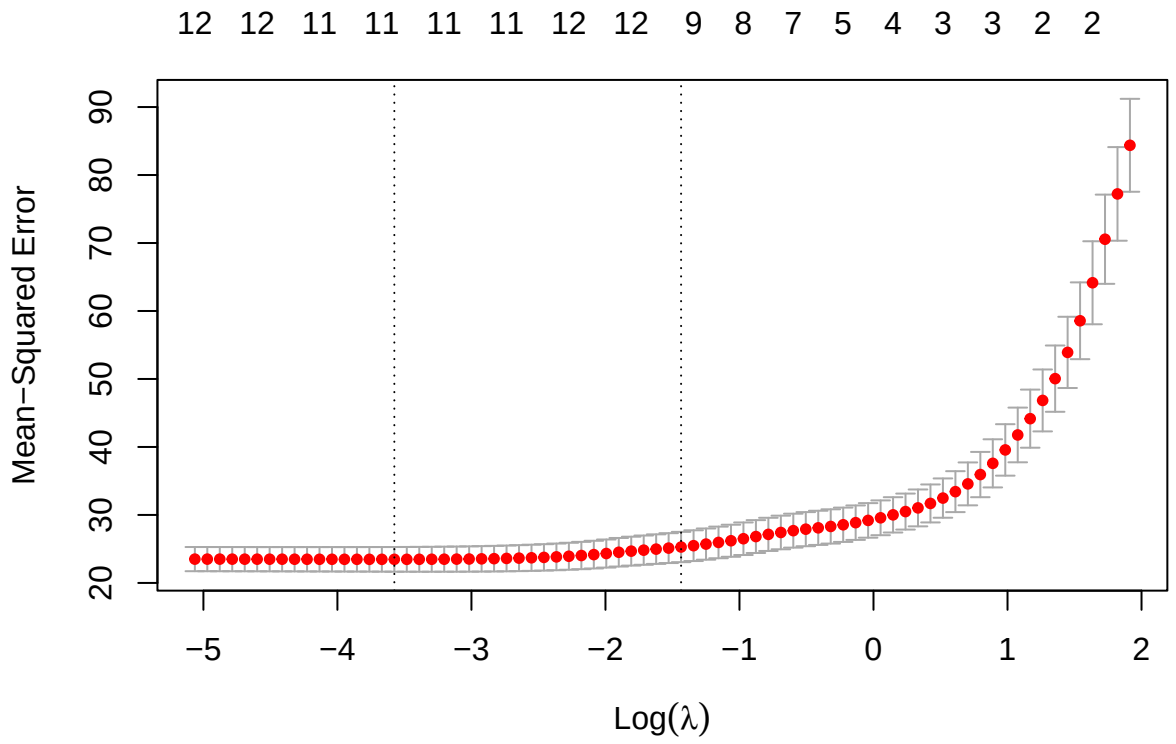
Firstly we define the explanatory and response variables to build the lasso regression model.

Using the cv.glmnet() function we fit a predictive model for house prices. The plot for the house prices using Lasso Regression model looks like below,

```
#Lasso Regression Modeling
#Separate predictors and response variable
x <- as.matrix(a_boston_data[, -14])  # Excluding the response variable (column 14)
y <- a_boston_data$medv #Considering the response variable
nfolds<-10
```

```r
type.measure <- "mse"
# Using cv.glmnet build
b_lasso_model <- cv.glmnet(x = x, y = y, alpha = 1,type.measure = type.measure, nfolds = nfolds)  # alp

# Plotting the cross-validation results
plot(b_lasso_model)
```




```r
# Identify the optimal lambda
b_optimal_lambda <- b_lasso_model$lambda.min
print(paste("Optimal Lambda for Lasso Regression is :", b_optimal_lambda))
```

```
## [1] "Optimal Lambda for Lasso Regression is : 0.0280053489032082"
```

```r
# Fit the final Lasso regression model with the optimal lambda
final_lasso_model <- glmnet(x = x, y = y, alpha = 1, lambda = b_optimal_lambda)
```

Optimal Lambda for Lasso Regression is : **0.0280053**

The coefficients of the final model are as follows

34.5481755992269, -0.0986932032938491, 0.0415882908898416, 0, 2.68163334369059, -16.3545905976131, 3.86003592555561, 0, -1.39969712143757, 0.255484621000573, -0.00993550913489892, -0.931031828057542, 0.00903142167911388, -0.522741591549162

**(c) Variable Selection:**

- **Extract coefficients from the Lasso model to identify which features have non-zero coefficients.**
- **Non-zero coefficients indicate selected features with significant predictive power.**

```
lasso_coefficients <- coef(final_lasso_model)
cat("The lasso coefficients are :\n")
```

```
## The lasso coefficients are :
```

```
lasso_coefficients
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  34.548175599
## crim         -0.098693203
## zn            0.041588291
## indus         .
## chas          2.681633344
## nox         -16.354590598
## rm            3.860035926
## age           .
## dis          -1.399697121
## rad           0.255484621
## tax          -0.009935509
## ptratio      -0.931031828
## black         0.009031422
## lstat        -0.522741592
```

```
# Identify features with non-zero coefficients
c_lasso_coefficients <- which(lasso_coefficients != 0)
cat("\nThe coefficients from Lasso Model which are non-zero are :\n")
```

```
##
## The coefficients from Lasso Model which are non-zero are :
```

```
c<- lasso_coefficients[c_lasso_coefficients]
c
```

```
##  [1]  34.548175599  -0.098693203   0.041588291   2.681633344 -16.354590598
##  [6]   3.860035926  -1.399697121   0.255484621  -0.009935509  -0.931031828
## [11]   0.009031422  -0.522741592
```

These coefficients correspond to the features Intercept, crim, zn, chaos, nox, rm, dis, rad, tax, ptratio, black, lstat.

**(d) Evaluation:**

- **Evaluate the performance of the Lasso model using cross-validated metrics such as Mean Squared Error (MSE) or others (cv.glmnet output)**
- **Assess the selected features' importance in predicting house prices.**

To evaluate the performance of Lasso Model using cross validated metrics, we find the Mean Square error as the minimum value of 'cvm' from the lasso model.

```r
cv_metrics <- b_lasso_model$cvm[which(b_lasso_model$lambda == b_optimal_lambda)]
cat("Cross-validated Mean Squared Error is :", mean(cv_metrics))
```

```
## Cross-validated Mean Squared Error is : 23.46833
```

Lower Mean Squared Error, indicates better predictive performance.

**(e) Interpretation:**

- **Interpret the results, including the selected features and their coefficients, to understand their impact on house prices.**

- **Discuss the implications and significance of the selected features in predicting house prices in Boston suburbs.**

**Importance of selected features in predicting house prices**:

(1) The magnitude of the coefficients is indicative of the feature's importance. The larger the absolute value of the coefficient, the more impact the feature has on predicting house prices.

(2) Chaos,rm and intercept highly impact the house prices(medv) and nox,dis have significantly low impact on house prices due to lower coefficient value.

(3) From the plot of the Lasso Model, we can infer that the house prices are significantly increasing with change in Lambda values.

*Vertical Lines:* These lines represent the optimal lambda chosen by cross-validation, indicating the level of regularization that gives the best model performance.

*Dots on Lines:* These dots indicate the coefficients at the optimal lambda for each feature.

---

## Construction of Polynomial Regression with degress of freedom

**Consider the faithful dataset in R, which records the waiting time between eruptions and the duration of eruptions of the Old Faithful geyser in Yellowstone National Park. Perform polynomial regression to model the relationship between waiting time and eruption duration. Utilizing 10-fold cross-validation, compute the cross-validated R2 values for polynomial regression models with degrees ranging from 1 to 4. Determine the best-fitting polynomial regression model by selecting the degree that yields the highest average R2 value across the different degrees**

**(a) Load the *faithful* dataset in R**

To load the dataset into the memory, I am using the data() function. Once the data set is loaded, a statement will be returned indicating the successful loading of the package *"faithful"*.

```r
data(faithful)
```

**(b) Implement polynomial regression models with degrees from 1 to 4.**

After implementation, the summary of the model looks as follows.

```r
degrees <- 1:4
r2_values <- numeric(length(degrees))

for (degree in degrees) {
  # Fit polynomial regression model
  poly_model <- lm(eruptions ~ poly(waiting, degree), data = faithful)
}
summary(poly_model)
```

```
##
## Call:
## lm(formula = eruptions ~ poly(waiting, degree), data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.33694 -0.25509  0.00282  0.28018  0.94928
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              3.48778    0.02496 139.748  < 2e-16 ***
## poly(waiting, degree)1 16.92565    0.41161  41.121  < 2e-16 ***
## poly(waiting, degree)2 -1.38262    0.41161  -3.359 0.000896 ***
## poly(waiting, degree)3 -4.06513    0.41161  -9.876  < 2e-16 ***
## poly(waiting, degree)4  1.69964    0.41161   4.129 4.87e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4116 on 267 degrees of freedom
## Multiple R-squared:  0.8719, Adjusted R-squared:  0.8699
## F-statistic: 454.2 on 4 and 267 DF,  p-value: < 2.2e-16
```

**(c) Use 10-fold cross-validation to compute R2 values for each model.**

```r
# (c) Use 10-fold cross-validation to compute R2 values for each model.
degrees <- 1:4
r2_values <- numeric(length(degrees))

for (degree in degrees) {
  # Fit polynomial regression model
  poly_model <- lm(eruptions ~ poly(waiting, degree), data = faithful)

  # Use 10-fold cross-validation to compute R2 values
  cv_r2 <- 1 - with(faithful, sum(residuals(poly_model)^2) / sum((eruptions - mean(eruptions))^2))

  # Store R2 value
  r2_values[degree] <- cv_r2
}
cat("Degree\tCrossValidatedR^2\n")
```

```
## Degree   CrossValidatedR^2
```

```r
for(i in 1:4){
  cat(paste(degrees[i],"\t", r2_values[i]),"\n")
}
```

```
## 1     0.81146076097331
## 2     0.816875523318087
## 3     0.863684099043419
## 4     0.871866714677926
```

```r
cat("\n The Cross Validated R^2 values are : ",r2_values)
```

```
##
##  The Cross Validated R^2 values are :  0.8114608 0.8168755 0.8636841 0.8718667
```

**(d) Identify the degree that corresponds to the highest average cross-validated R2 value. Provide the selected degree and its corresponding average R2 value as the solution.**

```r
# degree that corresponds to the highest average cross-validated R2 value
best_degree <- which.max(r2_values)
best_r2_value <- max(r2_values)

# Print the selected degree and its corresponding average R2 value
cat("The Degree:", best_degree, "\n")
```

```
## The Degree: 4
```

```r
cat("Corresponding Average R2 Value:", best_r2_value, "\n")
```

```
## Corresponding Average R2 Value: 0.8718667
```

The degree that corresponds to the highest average cross-validated R2 value is : **4**

The average R2 value corresponding to the best degree is : **0.8718667**

---