Machine Learning Assignment 1

Agustín Valencia - aguva779

11/19/2019

Assignment 1. Spam classification with nearest neighbors

2. Use logistic regression to classify the training and test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.5$, otherwise $\hat{Y} = 0$ and report the confusion matrices and the misclassification rates for train and test data. Analyze the obtained results.

Evaluating the model with training data :

```
## Classification Performance : train set - trigger = 0.5
##  TPR = 83.48083 % - TNR = 84.86906 % - FPR = 16.51917 % - FNR = 15.13094 %
##  Misclassification Rate =  15.47445 %
```

Now, with unseen data it can be observed that the misclassification rate increased, though numbers still consistent.

```
## Classification Performance : test set - trigger = 0.5
##  TPR = 74.92711 % - TNR = 84.2259 % - FPR = 25.07289 % - FNR = 15.7741 %
##  Misclassification Rate =  18.10219 %
```

3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.8$, otherwise $\hat{Y} = 0$

Setting a higher trigger implies that the classifier will be more selective, then it is expected to decrease the amount of mails being labeled as spam.

The training stats:

```
## Classification Performance : train set - trigger = 0.8
##  TPR = 87.10938 % - TNR = 80.61041 % - FPR = 12.89062 % - FNR = 19.38959 %
##  Misclassification Rate =  18.17518 %
```

Testing stats:

```
## Classification Performance : test set - trigger = 0.8
##  TPR = 76.30522 % - TNR = 79.57181 % - FPR = 23.69478 % - FNR = 20.42819 %
##  Misclassification Rate =  21.0219 %
```

Although the misclassification rate has increased, the false positive rate, i.e., the amount of valid email being sent to the spambox, has decreased, which from a user perspective could be more valuable than a higher accuracy on true positives.

4. Use standard kknn() with K = 30 from package *kknn*, report the misclassification rates for the training and test data and compare the results with step 2.

```
## Classification Performance : train knn - k = 30
##  TPR = 70.42802 % - TNR = 91.00467 % - FPR = 29.57198 % - FNR = 8.995327 %
##  Misclassification Rate =  16.71533 %
```

```
## Classification Performance : test knn - k = 30
##  TPR = 48.97541 % - TNR = 79.59184 % - FPR = 51.02459 % - FNR = 20.40816 %
##  Misclassification Rate =  31.31387 %
```

5. Repeat step 4 for K=1 and compare results with step 4. What effects does the decrease of K lead to and why?

```
## Classification Performance : train knn - k = 1
##  TPR = 100 % - TNR = 100 % - FPR = 0 % - FNR = 0 %
##  Misclassification Rate =  0 %
```

```
## Classification Performance : test knn - k = 1
##  TPR = 43.25323 % - TNR = 77.68396 % - FPR = 56.74677 % - FNR = 22.31604 %
##  Misclassification Rate =  35.91241 %
```

If we assign k=1 training misclassification is 0%, this means we are overfitting our model, thus the misclassification for the testing set may be bigger than other scenarios.

Assignment 3. Feature selection by cross-validation in a linear model.

1. Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions) your function should depend on:

- X: Matrix containing X measurements.
- Y: Vector containing Y measurements.
- $Nfolds$: number of folds in the cross-validation.

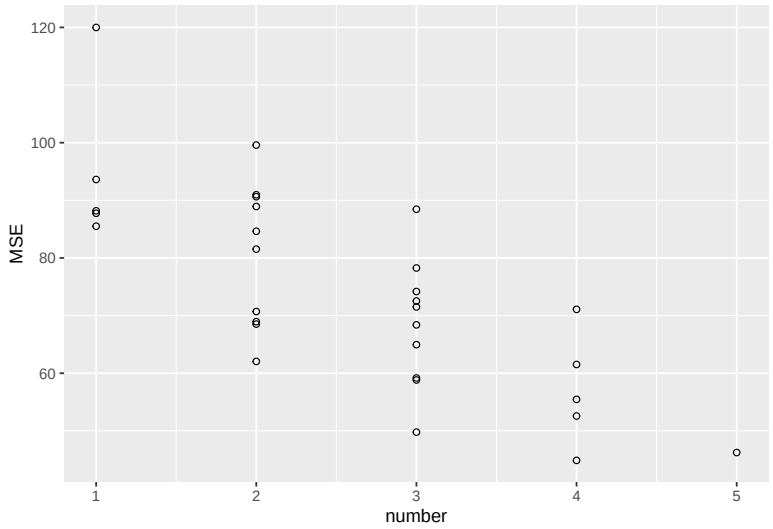
You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted and the seed 12345 should be used for that purpose.

2. Test your function on data set swiss available in the standard R repository:

- Fertility should be Y
- All other variables should be X
- Nfolds should be 5

Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

```
## CV : 44.88356
## Features : 1 0 1 1 1
```

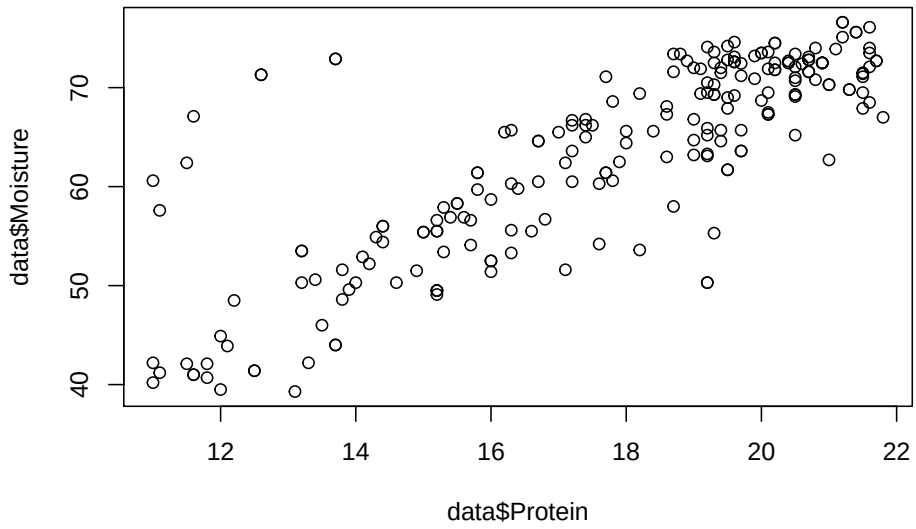


From the MSE plot it can be observed that along the number of features under consideration is increased, the MSE measured tend to decrease. This indicates that using simple models underfits the data, though using too many can also overfit the data and make the MSE scores increase again.

The minimum MSE score is reached at 4 features and its value is 44.88356. The best features subset is $x = (1, 0, 1, 1, 1)$ which comprise agriculture, education, religion and infant mortality data. As examination feature is dropped it can be intepreted that marks on army examinations do not contribute to explain Fertility changes.

Assignment 4. Linear regression and regularization

1. Import data and create a plot of Moisture versus Protein. Do you think these data are described well by a linear model?



By the plot, although there are some outliers, it seems that the data could be approximated by a linear model.

2. Consider model M_i in which Moisture is normally distributed and the expected Moisture is polynomial

$$M_i = \sum_{j=0}^{i} \beta_j x^j + \varepsilon$$
$$i = 1, \cdots, 6$$
$$\varepsilon \sim N(\mu, \sigma^2)$$

3. Divide the data (50/50) and fit models $M_i, i = 1, \cdots, 6$. For each model, record the training and validation MSE and present a plot showing how training and validation MSE depend on i. Which model is best according to this plot? How do MSE values change and why? Interpret this picture in bias-variance tradeoff.

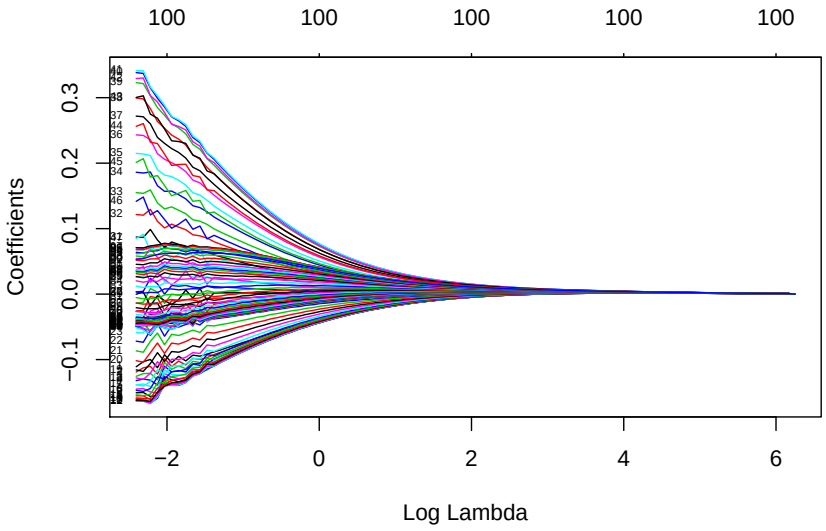
4. Perform variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC. Comment on how many variables were selected.

After running stepAIC we get that the amount of selected variables is :

```
## There were selected 64 variables
```

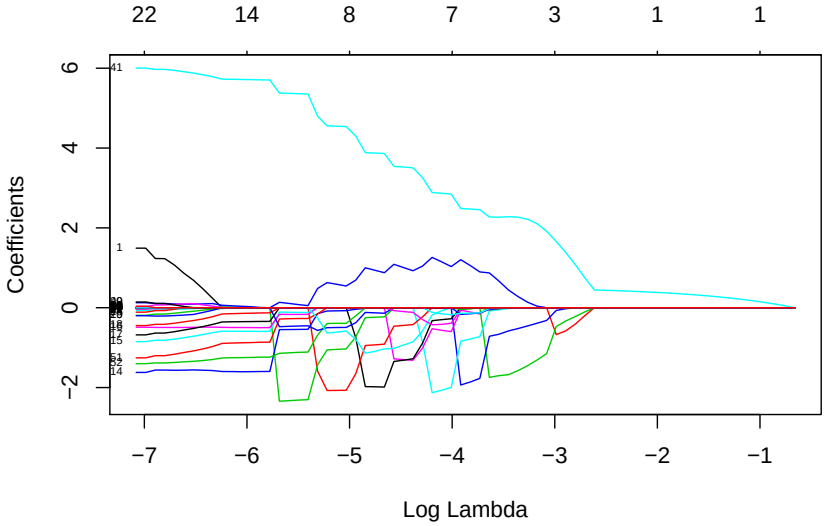
Thus, taking into account that one of them is the intercept, we have 63 selected variables out of 100.

5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how the coefficients change with λ



It can be seen that when using Ridge regression among the λ increasing, coefficients converge to zero, though the amount of parameters still 100 since Ridge do not drop them.

6. Repeat step 5 but fit with LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?



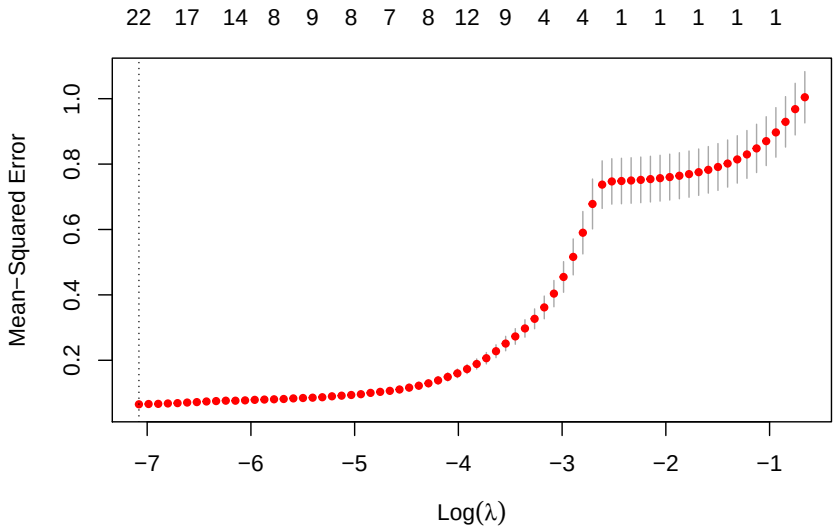
LASSO converges to zero much faster than Ridge regression and it also drops variables.

7. Use cross-validation to find optimal LASSO model (make sure that case $\lambda = 0$ is also considered by the procedure), report the optimal λ and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes λ

```
## The best performance was at lambda =  0
```

```
##
## Call:  cv.glmnet(x = as.matrix(covariates), y = response, lambda = lambdas,      alpha = 1, family =
##
```

```
## Measure: Mean-Squared Error
##
##        Lambda Measure       SE Nonzero
## min 0.0000000 0.05932 0.006345     100
## 1se 0.0008422 0.06538 0.006221      22
```



From the cross-validated model it is seen that the λ value for which it is obtained the minimum MSE score is at $\lambda_{min} = 0$. The model has 100 non-zero parameters. Along $\log(\lambda)$ increases MSE also increases

8. Compare the results from steps 4 and 7.

For (4) after performing a stepAIC over the model 36 variables were dropped, getting a final model with only 63 variables (plus the intercept). Nonetheless, for (7) after cross-validating the LASSO model it has been found that the best performance regarding MSE scores is at $\lambda = 0$ which implies no penalization, thus, no parameters will be dropped.

Appendix A - Code

Question 1

1.1

```
data <- read.xlsx("data/spambase.xlsx")
n = dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = data[id,]
test = data[-id,]
```

1.2

Fitting and util function

```
# util function
get_performance <- function(targets, predictions, text) {
    cat("Classification Performance :", text, "\n")
    t <- table(targets, predictions)
    #print("Confusion Matrix")
    #print(t)
    tn <- t[1,1]
    tp <- t[2,2]
    fp <- t[1,2]
    fn <- t[2,1]
    total <- dim(test)[1]
    tpr <- tp/(tp+fp) * 100
    tnr <- tn/(tn+fn) * 100
    fpr <- fp/(tp+fp) * 100
    fnr <- fn/(tn+fn) * 100

    #cat("Rates details:\n")
    cat(" TPR =", tpr, "% -")
    cat(" TNR =", tnr, "% -")
    cat(" FPR =", fpr, "% -")
    cat(" FNR =", fnr, "%")
    cat("\n Misclassification Rate = ", (fp+fn)/total * 100, "%\n")
}

# fit the model
fit <- glm(Spam ~ . , data = train, family = "binomial")
```

Trigger = 0.5 - training set

```
# performance on training data
pred_train <- predict(fit, newdata = train)
pred_train_at_05 <- as.integer(pred_train > 0.5)
targets <- train$Spam
get_performance(targets, pred_train_at_05, "train set - trigger = 0.5")
```

Trigger = 0.5 - testing set

```
# performance on test data
pred_test <- predict(fit, newdata = test)
pred_test_at_05  <- as.integer(pred_test > 0.5)
```

```
targets <- test$Spam
get_performance(targets, pred_test_at_05, "test set - trigger = 0.5")
```

1.3

Trigger = 0.8 - training set

```
# performance on train data
pred_train_at_08  <- as.integer(pred_train > 0.8)
targets <- train$Spam
get_performance(targets, pred_train_at_08, "train set - trigger = 0.8")
```

Trigger = 0.8 - testing set

```
# performance on test data
pred_test_at_08  <- as.integer(pred_test > 0.8)
targets <- test$Spam
get_performance(targets, pred_test_at_08, "test set - trigger = 0.8")
```

1.4

```
# Train KNN K=30
knn_model <- train.kknn(Spam ~ . , data = train, ks = 30)

# performance on training data
knn_fit <- predict(knn_model, train)
results <- as.integer(knn_fit > 0.5)
target <- train$Spam
get_performance(target, results, "train knn - k = 30")

# performance on test data
knn_fit <- predict(knn_model, test)
results <- as.integer(knn_fit > 0.5)
target <- test$Spam
get_performance(target, results, "test knn - k = 30")
```

1.5

```
# Train KNN K=1
knn_model <- train.kknn(Spam ~ . , data = train, ks = 1)

# performance on training data
knn_fit <- predict(knn_model, train)
results <- as.integer(knn_fit > 0.5)
target <- train$Spam
get_performance(target, results, "train knn - k = 1")

# performance on test data
knn_fit <- predict(knn_model, test)
results <- as.integer(knn_fit > 0.5)
target <- test$Spam
get_performance(target, results, "test knn - k = 1")
```

Question 3

3.1 - 3.2

```
#linear regression
mylin <- function(x_train, Y, x_test) {
  x_train1 <- cbind(1, x_train)
  beta <- solve( t(x_train1) %*% x_train1 ) %*% t(x_train1) %*% Y
  x_test1 <- cbind(1, x_test)
  Res <- x_test1 %*% beta
  return(Res)
}

myCV <- function(X, Y, Nfolds){
  n <- length(Y)
  p <- ncol(X)
  set.seed(12345)
  ind <- sample(n,n)
  X1 <- X[ind,]
  Y1 <- Y[ind]
  sF <- floor(n/Nfolds)
  MSE <- numeric(2^p-1)
  Nfeat <- numeric(2^p-1)
  Features <- list()
  curr <- 0

  for (f1 in 0:1)
    for (f2 in 0:1)
      for(f3 in 0:1)
        for(f4 in 0:1)
          for(f5 in 0:1){
            model <- c(f1,f2,f3,f4,f5)
            if (sum(model) == 0) {
                next()
            }
            SSE <- 0
            for (k in 1:Nfolds){
                # compute which indices should belong to current fold
                current_feat <- which(model==1)
                # implement cross-validation for model with features in
                # "model" and iteration i.
                begin_pos <- (k - 1) * (sF + 1)
                if(k == Nfolds){
                  end_pos <- length(Y1)
                }else{
                  end_pos <- k*sF
                }

                x_train <- X1[-begin_pos:-end_pos,current_feat]
                y_train <- Y1[-begin_pos:-end_pos]
                x_test <- X1[begin_pos:end_pos,current_feat]
                Ypred <- mylin(x_train,y_train,x_test)

                # Get the predicted values for fold 'k', Ypred, and the original
```

```
                # values for folf 'k', Yp.
                Yp <- Y1[begin_pos:end_pos]
                SSE <- SSE + sum((Ypred - Yp)^2)
            }
            curr <- curr + 1
            MSE[curr] <- SSE/n
            Nfeat[curr] <- sum(model)
            Features[[curr]] <- model
          }
  # plot MSE against number of features
  df <- data.frame(number=c(),MSE=c())
  for(i in 1:length(Features)){
    tmp <- data.frame(number=sum(Features[[i]]),MSE=MSE[i])
    df <- rbind(df,tmp)
  }
  p <- ggplot(df, aes(x = number, y = MSE)) + geom_point(shape=21)
  i <- which(MSE == min(MSE))
  CV = MSE[i]
  Features = Features[[i]]
  cat("CV :", CV, "\n")
  cat("Features :", Features, "\n")
  p
}

myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)
```

Question 4

4.1.

```
## Import data and plot Moisture vs Protein.
data <- read.xlsx("data/tecator.xlsx")
plot(data$Protein, data$Moisture)
```

4.3

```
# Creating data sets
p <- data$Protein
y <- data$Moisture
P <- data.frame(
    Y = y,
    P1 = p,
    P2 = p^2,
    P3 = p^3,
    P4 = p^4,
    P5 = p^5,
    P6 = p^6
)

n = dim(P)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = P[id,]
```

```
test = P[-id,]

# Training models
M1 <- lm(Y ~ ., data = train[,1:2])
M2 <- lm(Y ~ ., data = train[,1:3])
M3 <- lm(Y ~ ., data = train[,1:4])
M4 <- lm(Y ~ ., data = train[,1:5])
M5 <- lm(Y ~ ., data = train[,1:6])
M6 <- lm(Y ~ ., data = train[,1:7])

## Train Scores
eval_model <- function(model, data) {
  pred <- predict(model, data)
  errors <- pred - data$Y
  MSE <- mean(errors^2)
  return(MSE)
}

MSE_train <- c()
MSE_train[1] <- eval_model(M1, train)
MSE_train[2] <- eval_model(M2, train)
MSE_train[3] <- eval_model(M3, train)
MSE_train[4] <- eval_model(M4, train)
MSE_train[5] <- eval_model(M5, train)
MSE_train[6] <- eval_model(M6, train)

MSE_test <- c()
MSE_test[1] <- eval_model(M1, test)
MSE_test[2] <- eval_model(M2, test)
MSE_test[3] <- eval_model(M3, test)
MSE_test[4] <- eval_model(M4, test)
MSE_test[5] <- eval_model(M5, test)
MSE_test[6] <- eval_model(M6, test)

df <- data.frame(
    degree <- c(1:6),
    MSE_train,
    MSE_test
)
p <- ggplot()
p <- p +geom_point(data = df, aes( x = degree, y = MSE_train, color="Train")) +
    geom_line(data = df, aes( x = degree, y = MSE_train, color="Train"))
p <- p + geom_point(data = df, aes( x = degree, y = MSE_test, color="Test")) +
    geom_line(data = df, aes( x = degree, y = MSE_test, color="Test"))
p <- p + labs(y="MSE", colour="Dataset", title = "Mean Square Errors") + geom_line()
p
```

4.4

```
adhok_data <- data[,2:101]
Fat <- data$Fat
adhok_data <- cbind(adhok_data, Fat)
model <- lm(Fat ~ . , data=adhok_data)
```

```
step <- stepAIC(model, direction ="both")
selected_vars <- step$coefficients
cat("There were selected", length(selected_vars), "variables\n")
```

4.5

```
covariates <- scale(adhok_data[1:(length(adhok_data)-1)])
response <- scale(adhok_data$Fat)
model_ridge <- glmnet(covariates, response, alpha = 0, family = "gaussian")
plot(model_ridge, xvar="lambda", label=T)
```

4.6

```
model_lasso <- glmnet(covariates, response, alpha = 1, family = "gaussian")
plot(model_lasso, xvar="lambda", label=T)
```

4.7

```
lambdas <- append(model_lasso$lambda,0)
cv_model_lasso <- cv.glmnet(as.matrix(covariates), response, alpha=1, family="gaussian", lambda=lambdas)
cat("The best performance was at lambda = ", cv_model_lasso$lambda.min, "\n")
cv_model_lasso
plot(cv_model_lasso)
```