Linear & Ridge Regression

About Data: The data file parkinson.csv is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The purpose is to predict Parkinson's disease symptom score (motor UPDRS) from the following voice characteristics:

- Jitter(%),Jitter(Abs),Jitter:RAP,Jitter:PPQ5,Jitter:DDP - Several measures of variation in fundamental frequency
- Shimmer,Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,Shimmer:APQ11,Shimmer:DDA - Several measures of variation in amplitude
- NHR,HNR - Two measures of ratio of noise to tonal components in the voice
- RPDE - A nonlinear dynamical complexity measure
- DFA - Signal fractal scaling exponent
- PPE - A nonlinear measure of fundamental frequency variation

Q 1. Scale the data and divide it into training and test data (60/40). In the coming steps, assume that motor_UPDRS is normally distributed and is a function of the voice characteristics, and since the data are scaled, no intercept is needed in the modelling. Compute a linear regression model from the training data, estimate training and test MSE and comment on which variables contribute significantly to the model

Ans: Based on the given results, the Mean Squared Error (MSE) for both the training and test data are as follows:

```
## Training data MSE: 0.8731931
```

```
## Test data MSE: 0.9294911
```

It is important to note that without another model for comparison, it is not advisable to draw conclusions solely based on the MSE values. However, it can be observed that the MSE values are around 1, with the training data showing slightly better performance compared to the test data.

To determine the significance of variables in the model, we can examine the p-values. A low p-value (< 0.05) indicates that the null hypothesis can be rejected, suggesting that a predictor with a low p-value is likely to be a meaningful addition to the model.

```
##     Jitter...   Jitter.Abs.    Jitter.RAP   Jitter.PPQ5    Jitter.DDP
##         FALSE          TRUE         FALSE         FALSE         FALSE
##       Shimmer   Shimmer.dB.  Shimmer.APQ3  Shimmer.APQ5 Shimmer.APQ11
##          TRUE         FALSE         FALSE          TRUE          TRUE
##   Shimmer.DDA           NHR           HNR          RPDE           DFA
##         FALSE          TRUE          TRUE         FALSE          TRUE
##           PPE
##          TRUE
```

Based on the analysis, the significant variables in the model are:

- "Jitter.Abs."
- "Shimmer"
- "Shimmer.APQ5"
- "Shimmer.APQ11"
- "NHR"
- "HNR"
- "DFA"
- "PPE"

These variables have demonstrated statistical significance and can be considered important contributors to the model.

Q 2. Implement 4 following functions by using basic R commands only (no external packages):

- *Loglikelihood* function that for a given parameter vector θ and dispersion *sigma* computes the log-likelihood function log$P(T|\theta, \sigma)$ for the stated model and the training data
- *Ridge* function that for given vector θ, scalar σ and scalar λ uses function from 3a and adds up a Ridge penalty $\lambda\|\theta\|\hat{}2$ to the minus loglikelihood
- *RidgeOpt* function that depends on scalar λ, uses function from 3b and function optim() with method="BFGS" to find the optimal θ and σ for the given λ.
- *DF* function that for a given scalar λ computes the degrees of freedom of the Ridge model based on the training data.

By using function RidgeOpt, compute optimal θ parameters for $\lambda = 1$, $\lambda = 100$ and $\lambda = 1000$. Use the estimated parameters to predict the motor_UPDRS values for training and test data and report the training and test MSE values. Which penalty parameter is most appropriate among the selected ones? Compute and compare the degrees of freedom of these models and make appropriate conclusions.

Ans: The results show that the test error is lowest when lambda is equal to 100. This suggests that 100 is the appropriate penalty parameter for this model. When lambda is equal to 1, the model is too complex and is prone to overfitting. As lambda increases, the model becomes simpler and is less prone to overfitting.

The degree of freedom is a measure of the complexity of the model. It is equal to the number of non-zero coefficients. As lambda increases, the degree of freedom decreases. This is because the coefficients are shrunk more, which makes them closer to zero.

In summary, the results suggest that 100 is the appropriate penalty parameter for this ridge regression model. This is because it results in the lowest test error and the simplest model.

```
## For lambda =1
##  Trainig Error: 0.8732769
##  Test Error: 0.9290359
##  Degree of Freedom: 13.86281

## For lambda = 100
##  Trainig Error: 0.8788127
##  Test Error: 0.9262592
##  Degree of Freedom: 9.939085

## For lambda = 1000
##  Trainig Error: 0.9074928
##  Test Error: 0.9421887
##  Degree of Freedom: 5.643351
```

Appendix

```
#Loglikelihood Function
loglikelihood <- function(theta){
  X <- as.matrix(train_df[7:22])
  n <- nrow(X)
  y <- train_df[,5, drop = FALSE]
  sigma <- theta[17]
  theta_1 <- theta[-17]
  logl<- (-(n/2) * log(2 * pi * sigma^2) - sum((y - X%*%theta_1)^2) * (1/(2*sigma^2)))
  return(-logl)
}
#Ride Function
ridge <- function(theta, lambda){
  n_logl <- loglikelihood(theta)
  res <- n_logl + lambda * sum(theta^2)
  return(res)
}
#RidgeOpt Function
ridgeOpt <- function(lambda){
  optim1 <- optim(rep(1:17), ridge, method="BFGS", lambda = lambda)
  return(optim1)
}

#Degree of Freedom Function
df <- function(lambda){
X <- as.matrix(train_df[7:22])
n <- nrow(X)
hat_matrix <- X %*% solve((t(X)%*%X + lambda* diag(ncol(X)))) %*% t(X)
df <- sum(diag(hat_matrix))
return(df)
}
#Function to calculate the MSE of training and Test for different Values of lambda
lambda_res<-function(lambda)
{
x_train <- as.matrix(train_df[7:22])
x_test <- as.matrix(test_df[7:22])
opt_lambda<- ridgeOpt(lambda)
theta <- opt_lambda$par[1:16]
fit_train <- x_train %*% theta
fit_test <- x_test%*% theta
training_err<- mean((fit_train - train_df$motor_UPDRS)^2)
test_err <- mean((fit_test - test_df$motor_UPDRS)^2)
dof<-df(lambda)
return(c(training_err, test_err, dof))
}
#importing data set
df <- read.csv("D:\\ML_Data\\LAB_1\\parkinsons.csv")
#Scaling data set
s_df<- as.data.frame(scale(df))
#Dividing data frame between test data and training data.
n <- dim(s_df)[1]
set.seed(12345)
```

```r
id <- sample(1:n, floor(n*0.6))
train_df <- s_df[id,]
test_df <- s_df[-id,]
train_df <- as.data.frame(train_df)
test_df <- as.data.frame(test_df)
#computing linear regression model and removing intercept
train_fit<- lm(motor_UPDRS ~ Jitter... + Jitter.Abs. + Jitter.RAP+ Jitter.PPQ5 +
Jitter.DDP + Shimmer + Shimmer.dB. + Shimmer.APQ3 + Shimmer.APQ5 +
+ Shimmer.APQ11 + Shimmer.DDA +NHR + HNR+ RPDE + DFA + PPE +0, data = train_df)
summary(train_fit)
#MSE for training data
mse_train = mean(train_fit$residuals^2)
cat("Training data MSE:", mse_train, "\n")
#MSE for test data
test_fit = predict(train_fit, test_df)
mse_test = mean((test_df$motor_UPDRS - test_fit)^2)
cat("Test data MSE:", mse_test, "\n")
#Checking the significance of variables
summary(train_fit)$coeff[,4] < 0.05
# MSE for different values of lambda
cat("For lambda =1", "\n", "Trainig Error:", lambda_res(1)[1],
    "Test Error:", lambda_res(1)[2],
    "Degree of Freedom:", lambda_res(1)[3], "\n")
cat("For lambda = 100", "\n", "Trainig Error:", lambda_res(100)[1],
    "Test Error:", lambda_res(100)[2],
    "Degree of Freedom:", lambda_res(100)[3], "\n")
cat("For lambda = 1000", "\n", "Trainig Error:", lambda_res(1000)[1],
    "Test Error:", lambda_res(1000)[2],
    "Degree of Freedom:", lambda_res(1000)[3], "\n")
```