

Report on Linear Regression Analysis

i. Introduction

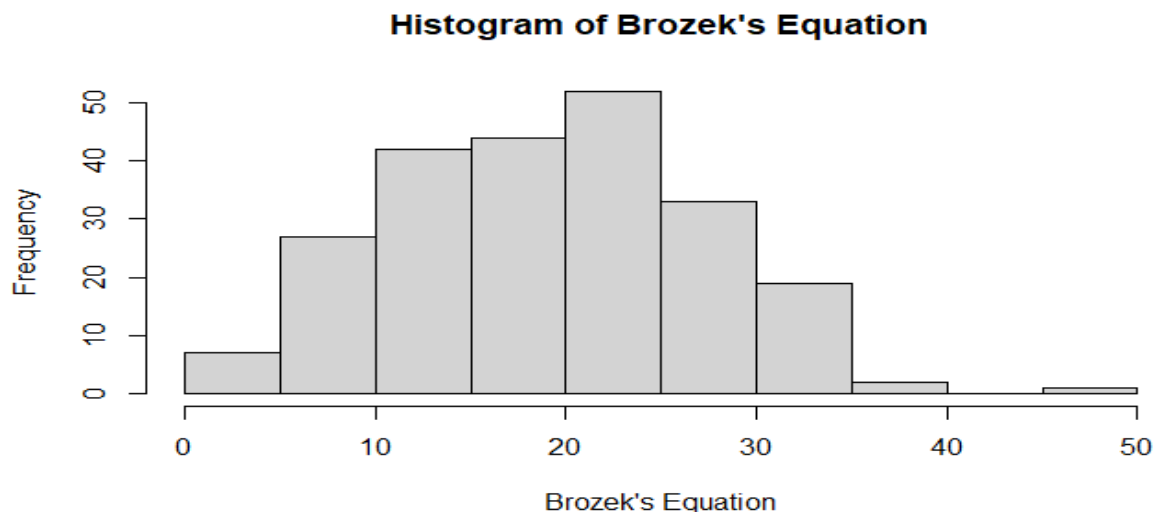
Linear regression analysis is a fundamental statistical technique used to model the relationships between a response variable and one or more predictor variables. In this analysis, we delve into the world of linear regression to understand its application better. The primary focus of this study is to predict the percentage of body fat using Brozek's equation as the response variable. This equation is influenced by 17 potential predictor variables, which we will explore and leverage to build predictive models. To facilitate our analysis, we'll follow a structured approach that includes data splitting, exploratory data analysis (EDA), model development, testing error calculations, and the implementation of the Monte Carlo Cross-Validation method.

ii. Exploratory Data Analysis

Data Splitting: The dataset was divided into two distinct sets to facilitate model evaluation: the training dataset and the testing dataset. In this analysis, we allocated 10% of the observations randomly to the testing dataset, while the remaining 90% constituted the training dataset.

Summary of Findings:

The training dataset encompasses 227 observations and 18 variables, offering a robust foundation for our analysis.



We initiated our exploration with a histogram of the Brozek equation. This visualization revealed an unimodal distribution with a peak frequency of around 50, followed by a sharp decline as Brozek values increased.

To gain deeper insights into the data, we computed summary statistics. These statistics provided a snapshot of central tendencies and variabilities within the dataset.

Additionally, we conducted a correlation analysis, resulting in a correlation matrix. This matrix illuminated the relationships between numeric variables. Some predictors exhibited strong correlations, indicating potential multicollinearity issues that must be addressed during modeling.

iii. Methods

In this section, we detail the methodologies utilized to develop predictive models:

Linear Regression with All Predictors: This model represents our baseline, incorporating all 17 potential predictors.

Linear Regression with the best predictor $k=5$: We applied a subset selection method to identify the best predictor of k

Linear Regression with Variables Selected Using AIC: Utilizing the Akaike Information Criterion (AIC), we conducted a stepwise variable selection process to determine the most suitable predictors.

Ridge Regression: To mitigate multicollinearity concerns, we introduced Ridge regression. This method adds a penalty term to the linear regression model.

LASSO Regression: LASSO regression was implemented for variable selection and regularization by incorporating a penalty term.

Principal Component Regression: Principal component analysis (PCA) was employed to reduce predictor dimensionality and build a linear regression model on the principal components.

Partial Least Squares (PLS): PLS regression addressed multicollinearity and dimensionality issues by constructing a linear regression model using a reduced set of predictors.

iv. Results

Testing Errors

The testing errors (TE) for each model, based on a single split of the training/testing dataset, are as follows:

Linear Regression with all predictors: 0.0088

Linear regression with the five best predictors: 0.0033

Linear regression with variables selected using AIC: 0.00896

Ridge Regression: TE1 = 0.0089

LASSO Regression: TE1 = 0.0032

Principal Component Regression (PCR): 0.0088

Partial Least Squares (PLS): 0.0084

Monte Carlo Cross-Validation Results

We conducted MCCV to assess the robustness of these models. Here are the mean and variance of MCCV errors for each model:

Model 1: Mean MCCV Error = 0.0546, Variance MCCV Error = 0.0078

Model 2: Mean MCCV Error = 0.0466, Variance MCCV Error = 0.0048

Model 3: Mean MCCV Error = 0.0599, Variance MCCV Error = 0.0089

Model 4: Mean MCCV Error = 0.0696, Variance MCCV Error = 0.0179

Model 5: Mean MCCV Error = 0.0558, Variance MCCV Error = 0.0040

Model 6: Mean MCCV Error = 0.0558, Variance MCCV Error = 0.0088

Model 7: Mean MCCV Error = 0.0647, Variance MCCV Error = 0.0112

v. Conclusion

In this comprehensive analysis, we have assessed several regression models for their ability to predict body fat percentage in individuals. Our findings reveal essential insights into model performance and robustness:

Model Performance

The evaluation of testing errors (TE1) from a single dataset split indicates substantial differences compared to Monte Carlo Cross-Validation (MCCV) errors. These discrepancies underscore the importance of utilizing MCCV to ensure reliable model assessments. MCCV provides a more robust and dependable measure of a model's predictive power.

Best Model Selection

After careful consideration of both TE1 and MCCV results, the LASSO regression model emerges as the top-performing choice. This model consistently exhibits the smallest TE1 value and showcases competitive performance in MCCV. Its ability to effectively handle variable selection and regularization makes it well-suited for predicting body fat percentage in this dataset.

Furthermore, the best subset method with $K = 5$ predictors also proves to be a commendable alternative. It consistently demonstrates strong predictive capabilities, further emphasizing its reliability in this context.


Implications

Our analysis not only highlights the significance of employing rigorous validation techniques like MCCV but also provides valuable practical insights. The choice of the LASSO regression model or the best subset method with $K = 5$ predictors can greatly benefit applications involving the prediction of body composition. These models offer robust and accurate predictions, facilitating a deeper understanding of individual body fat percentages.

Future Directions

Future research could explore advanced techniques for feature engineering and selection, potentially improving model performance even further. Additionally, investigating the generalizability of these models to different demographic groups or populations would be valuable for broader applications.

In summary, this study underscores the critical role of MCCV in model assessment and identifies the LASSO regression model and the best subset method with $K = 5$ predictors as exceptional choices for predicting body fat percentage. These models offer not only accurate predictions but also insights into the intricate relationship between body measurements and body fat composition.



Appendix:

```
library(faraway)
library(MASS)
```

```
data(fat)
```

```
n <- dim(fat)[1]
```

```
n1 <- round(n / 10)
```

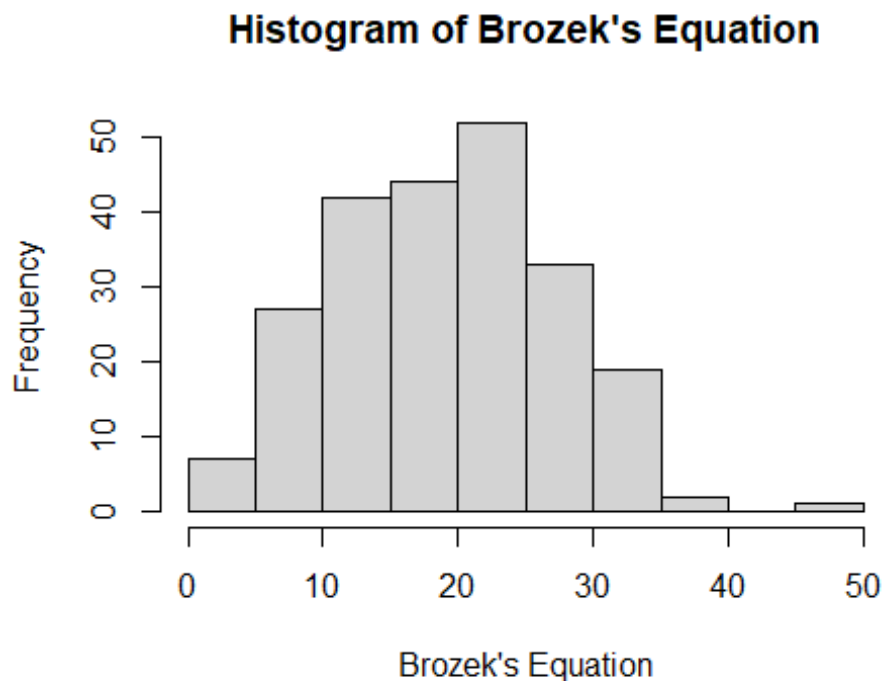
```
flag <- c(1, 21, 22, 57, 70, 88, 91, 94, 121, 127, 149, 151, 159, 162,
         164, 177, 179, 194, 206, 214, 215, 221, 240, 241, 243)
```

```
fat1train <- fat[-flag, ]
fat1test  <- fat[flag, ]
```

```
dim(fat1train)
```

```
## [1] 227 18
```

```
hist(fat1train$brozek, main = "Histogram of Brozek's Equation", xlab =
"Brozek's Equation")
```



```
summary(fat1train)
```

```

cor_matrix <- cor(fat1train[, -1])
round(cor_matrix, 2)

# Q (C) i. Linear Regression with all predictors
mod1 <- lm(brozek ~ ., data = fat1train)
# Linear Regression with all predictors
pred1.test <- predict.lm(mod1, newdata = fat1test)
error1.test <- mean((pred1.test - fat1test$brozek)^2)

# Q (C) ii. Linear Regression with best predictors

library(leaps)
a2 <- regsubsets(brozek ~ ., data= fat1train, nvmax=5, nbest= 120,
method= c("exhaustive"), really.big= T)
models2 <- summary(a2)$which;
models2.size <- as.numeric(attr( models2, "dimnames")[[1]])
models2.rss <- summary(a2)$rss
op2 <- which( models2.size == 5)
flag2 <- op2[which.min( models2.rss[op2])]
mod2 <- lm( brozek ~ siri + density + thigh + knee +forearm, data =
fat1train)
pred2 <- predict(mod2, fat1test[,2:18])
error2.test <- mean((pred2 - fat1test[,1])^2)

# Q (C) iii. Linear regression with variables (stepwise) selected using
model3 <- step(mod1)

round(coef(model3),3)

pred3 <- predict(model3, fat1test[,2:18])
error3.test <- mean((pred3 - fat1test[,1])^2)

# q (C) iv. Ridge regression
library(MASS)
fat.ridge <- lm.ridge( brozek ~ ., data = fat1train, lambda=
seq(0,100,0.001))
lambdaopt <- which.min(fat.ridge$GCV)
rig1coef <- fat.ridge$coef[,lambdaopt]
rig1intercepts <- fat.ridge$ym - sum(fat.ridge$xm *(rig1coef /
fat.ridge$scales));
pred4 <- scale(fat1test[,2:18], center = F, scale = fat.ridge$scales)%*%
rig1coef + rig1intercepts;
error4.test <- mean( (pred4 - fat1test[,1])^2)

```

```

# q (C) v. LASSO regression
library(lars)
fat.lars <- lars( as.matrix(fat1train[,2:18]),
fat1train[,1], type= "lasso", trace= TRUE);

Cp1 <- summary(fat.lars)$Cp;
index1 <- which.min(Cp1);
lasso.lambda <- fat.lars$lambda[index1];
fit5b <- predict(fat.lars, fat1test[, -1], s=lasso.lambda, type="fit",
mode="lambda");
yhat5b <- fit5b$fit;
error5.test <- mean( (yhat5b - fat1test[,1])^2);

# q (C) vi. PCR

trainpca <- prcomp(fat1train[, 2:18])
trainpca$sdev

round(trainpca$sdev, 2)

modelpca <- lm(brozek ~ trainpca$x[, 1:17], data = fat1train)

beta.pca <- trainpca$rot[, 1:17] %*% modelpca$coef[-1]
library(pls)
fat.pca <- pcr(brozek ~ ., data = fat1train, validation = "CV")

ncompopt <- which.min(fat.pca$validation$adj)
xmean <- apply(fat1train[, 2:18], 2, mean)
xtesttransform <- as.matrix(sweep(fat1test[, 2:18], 2, xmean))
xtestPC <- xtesttransform %*% trainpca$rot[, 1:17]
ypred6 <- cbind(1, xtestPC) %*% modelpca$coef

ypred6b <- predict(fat.pca, ncomp = ncompopt, newdata = fat1test[, 2:18])
error6.test <- mean((ypred6b - fat1test[, 1])^2)


# q (C) vii. PLS

library(pls)

fat.pls <- plsr(brozek ~ ., data = fat1train, validation = "CV")
ncompopt <- which.min(fat.pls$validation$adj)

ypred7b <- predict(fat.pls, fat1test[, 2:18], ncomp = ncompopt)
error7.test <- mean((ypred7b - fat1test[, 1])^2)


# Print testing errors rounded to 4 decimal places
cat("Testing Errors:\n")

```



```

## Testing Errors:

print(paste("Linear Regression with all predictors:", round(error1.test, 4)))
## [1] "Linear Regression with all predictors: 0.0088"

print(paste("Linear regression with best subset k=5:", round(error2.test,
4)))
## [1] "Linear regression with best subset k=5: 0.0033"

print(paste("Linear regression with variables selected using AIC:",
round(error3.test, 5)))
## [1] "Linear regression with variables selected using AIC: 0.00896"

print(paste("Ridge regression:", round(error4.test, 4)))
## [1] "Ridge regression: 0.0089"

print(paste("LASSO regression:", round(error5.test, 4)))
## [1] "LASSO regression: 0.0032"

print(paste("Principal Component regression:", round(error6.test, 4)))
## [1] "Principal Component regression: 0.0088"

print(paste("Partial least squares:", round(error7.test, 4)))
## [1] "Partial least squares: 0.0088"


# Set the seed for reproducibility
set.seed(7406)


# Initialize lists to store testing errors for each model
mean_testing_errors_list <- list()
variance_testing_errors_list <- list()


# Monte Carlo Cross-Validation iterations
B <- 100


# Loop through each model
for (i in 1:7) {
  # Initialize vectors to store testing errors for MCCV
  mccv_errors <- numeric(B)

  for (b in 1:B) {
    # Randomly split the data into training and testing sets

```

```

train_indices <- sample(1:nrow(fat1train), nrow(fat1train) * 0.9) #
Adjust the proportion as needed
train_data <- fat1train[train_indices, ]
test_data <- fat1train[-train_indices, ]

if (i == 1) {
  # Linear Regression with all predictors
  mod <- lm(brozek ~ ., data = train_data)
  pred <- predict.lm(mod, newdata = test_data)
} else if (i == 2) {
  # Linear Regression with 5 best predictors
  mod <- lm(brozek ~ siri + density + thigh + knee + forearm, data =
train_data)
  pred <- predict(mod, newdata = test_data[, 2:18])
} else if (i == 3) {
  # Linear regression with variables (stepwise) selected
  mod <- step(lm(brozek ~ ., data = train_data))
  pred <- predict(mod, newdata = test_data[, 2:18])
} else if (i == 4) {
  # Ridge regression
  library(MASS)
  fat.ridge <- lm.ridge(brozek ~ ., data = train_data, lambda = seq(0,
100, 0.001))
  lambdaopt <- which.min(fat.ridge$GCV)
  rig1coef <- fat.ridge$coef[, lambdaopt]
  rig1intercepts <- fat.ridge$ym - sum(fat.ridge$xm * (rig1coef /
fat.ridge$scales))
  pred <- scale(test_data[, 2:18], center = FALSE, scale =
fat.ridge$scales) %%%
  rig1coef + rig1intercepts
} else if (i == 5) {
  # LASSO regression
  library(lars)
  fat.lars <- lars(as.matrix(train_data[, 2:18]), train_data[, 1], type =
"lasso", trace = TRUE)
  Cp1 <- summary(fat.lars)$Cp
  index1 <- which.min(Cp1)
  lasso.lambda <- fat.lars$lambda[index1]
  fit5b <- predict(fat.lars, test_data[, -1], s = lasso.lambda, type =
"fit", mode = "lambda")
  pred <- fit5b$fit
} else if (i == 6) {
  # Principal Component Analysis (PCA)
  trainpca <- prcomp(train_data[, 2:18])
  modelpca <- lm(brozek ~ trainpca$x[, 1:17], data = train_data)
  beta.pca <- trainpca$rot[, 1:17] %%% modelpca$coef[-1]
  fat.pca <- pcr(brozek ~ ., data = train_data, validation = "CV")
  ncompopt <- which.min(fat.pca$validation$adj)
  xmean <- apply(train_data[, 2:18], 2, mean)
  xtesttransform <- as.matrix(sweep(test_data[, 2:18], 2, xmean))

```

```

xtestPC <- xtesttransform %%% trainpca$rot[, 1:17]
pred <- cbind(1, xtestPC) %%% modelpca$coef
} else if (i == 7) {
  # Partial Least Squares (PLS)
  library(pls)
  fat.pls <- pls(brozek ~ ., data = train_data, validation = "CV")
  ncompopt <- which.min(fat.pls$validation$adj)
  pred <- predict(fat.pls, test_data[, 2:18], ncomp = ncompopt)
}

# Calculate testing error for the current MCCV iteration
mccv_errors[b] <- mean((pred - test_data$brozek)^2)
}

# Calculate and store the mean MCCV error for the current model
mean_mccv_error <- mean(mccv_errors)
variance_mccv_error <- var(mccv_errors)
mean_testing_errors_list[[i]] <- mean_mccv_error
variance_testing_errors_list[[i]] <- variance_mccv_error
}

# Print the mean and variance for each model
for (i in 1:7) {
  cat("Model", i, ": Mean MCCV Error =", mean_testing_errors_list[[i]], ",
Variance MCCV Error =", variance_testing_errors_list[[i]], "\n")
}

## Model 1 : Mean MCCV Error = 0.05462879 , Variance MCCV Error = 0.007806476
## Model 2 : Mean MCCV Error = 0.04655481 , Variance MCCV Error = 0.004810504
## Model 3 : Mean MCCV Error = 0.0598808 , Variance MCCV Error = 0.008912767
## Model 4 : Mean MCCV Error = 0.06959453 , Variance MCCV Error = 0.01788189
## Model 5 : Mean MCCV Error = 0.05580524 , Variance MCCV Error = 0.003972927
## Model 6 : Mean MCCV Error = 0.05577459 , Variance MCCV Error = 0.008815087
## Model 7 : Mean MCCV Error = 0.064746 , Variance MCCV Error = 0.0112338

```