

Introduction to Machine Learning

Lab 2

Rasmus Holm

2016-11-11

Contents

Assignment 1	2
2	2
Assignment 2	4
1	4
2	4
3	5
4	6
5	6
6	7
7	8
8	8
Appendix	9
Code for Assignment 1	9
Code for Assignment 2	11

Assignment 1

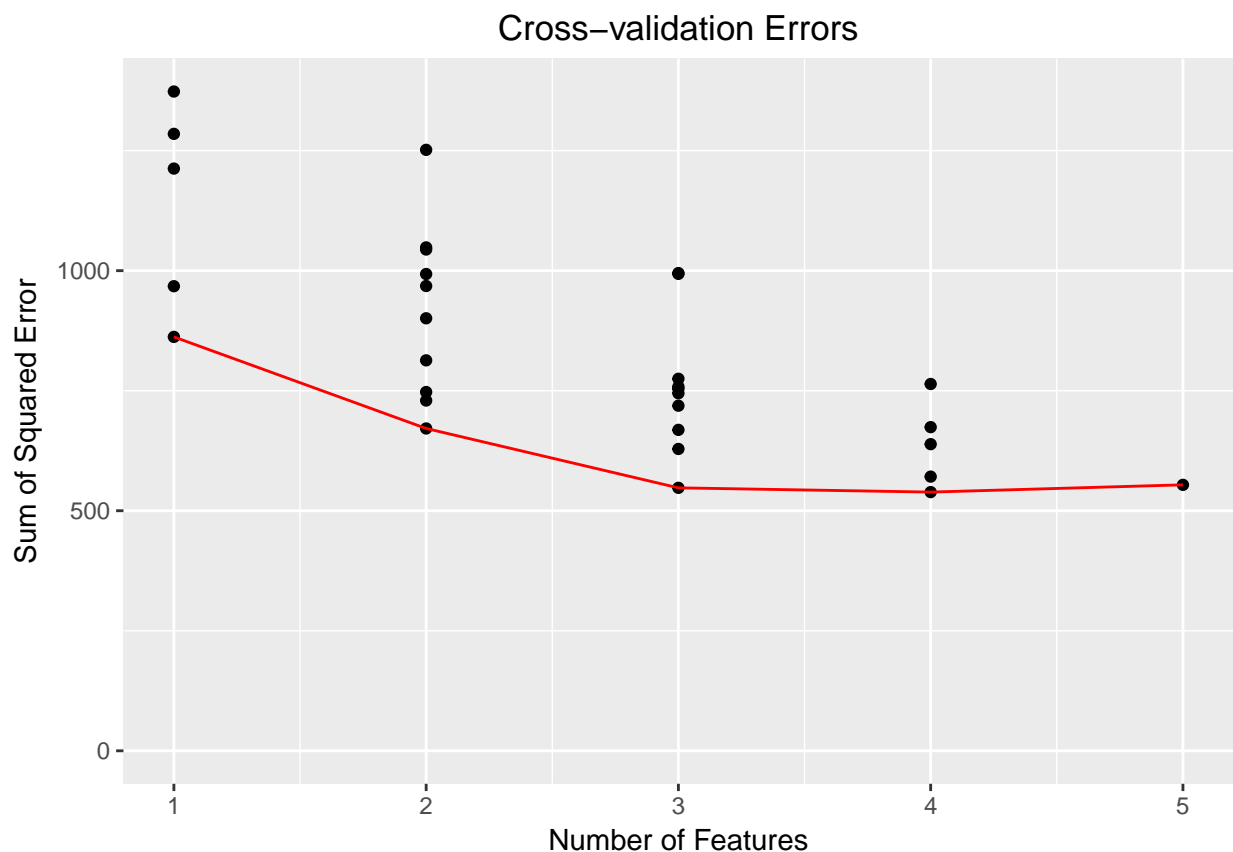
In this assignment I have used the swiss data set containing information about fertility in various provinces of Switzerland at about 1888, 47 observations in total. The features are

- Fertility: common standardized fertility measure
- Agriculture: % of males involved in agriculture as occupation
- Examination: % draftees receiving highest mark on army examination
- Education: % education beyond primary school for draftees
- Catholic: % 'catholic' (as opposed to 'protestant')
- Infant Mortality: live births who live less than 1 year

and I am interesting in predicting the fertility.

2

In order to find the best set of features in terms of sum of squared error (SSE) for linear regression I used cross-validation with 5 folds. Figure ?? shows that 4 features yield the least SSE out of all models.



With the knowledge gained from cross-validation I trained the final linear regression model on the complete data set. The model resulted in

$$\begin{aligned}\hat{y} = & \text{intercept} * 62.1013116 - \text{Agriculture} * 0.1546175 \\ & - \text{Education} * 0.9802638 + \text{Catholic} * 0.1246664 \\ & + \text{Infant Mortality} * 1.0784422.\end{aligned}$$

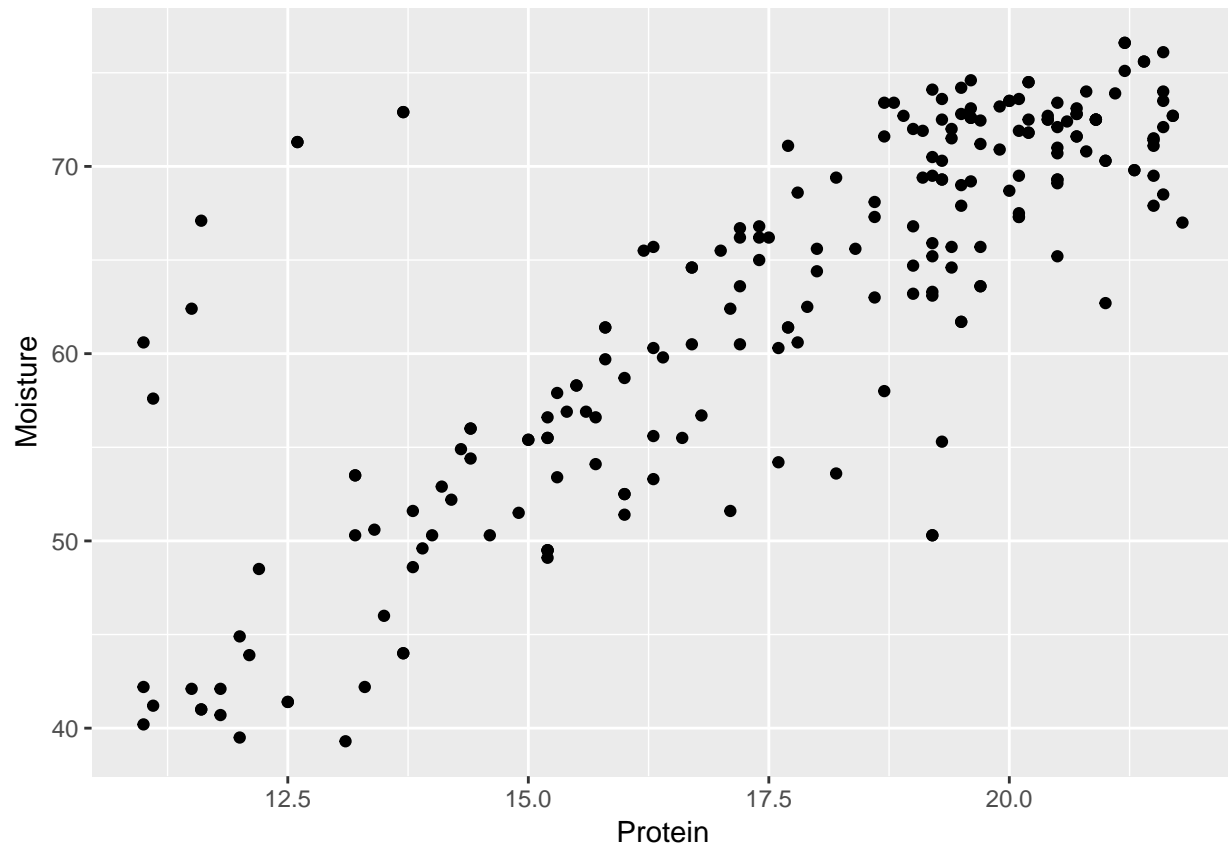
We can see that education decreases the fertility measure which is unsurprising and is present in today's world where industrial countries have in general lower fertility rate compared to developing countries. That infant mortality increases the fertility measure is not surprising either, survival is the key as with all species of animals and that means reproduction and if children dies more have to be born to balance it out. Agriculture and Catholic variables do not have as much affect on the fertility measure and it is not obvious why they are important. A lot of males occupied in agriculture probably means that food is available and it is therefore less requiring for extra sons to be born to provide it. A population with big portion being catholic are perhaps more happy in general and feel like reproducing serves a purpose.

Assignment 2

In this assignment I have used the tecator data set.

1

Figure ?? clearly shows a linear correlation between moisture and protein with a few observations deviating from the overall pattern.

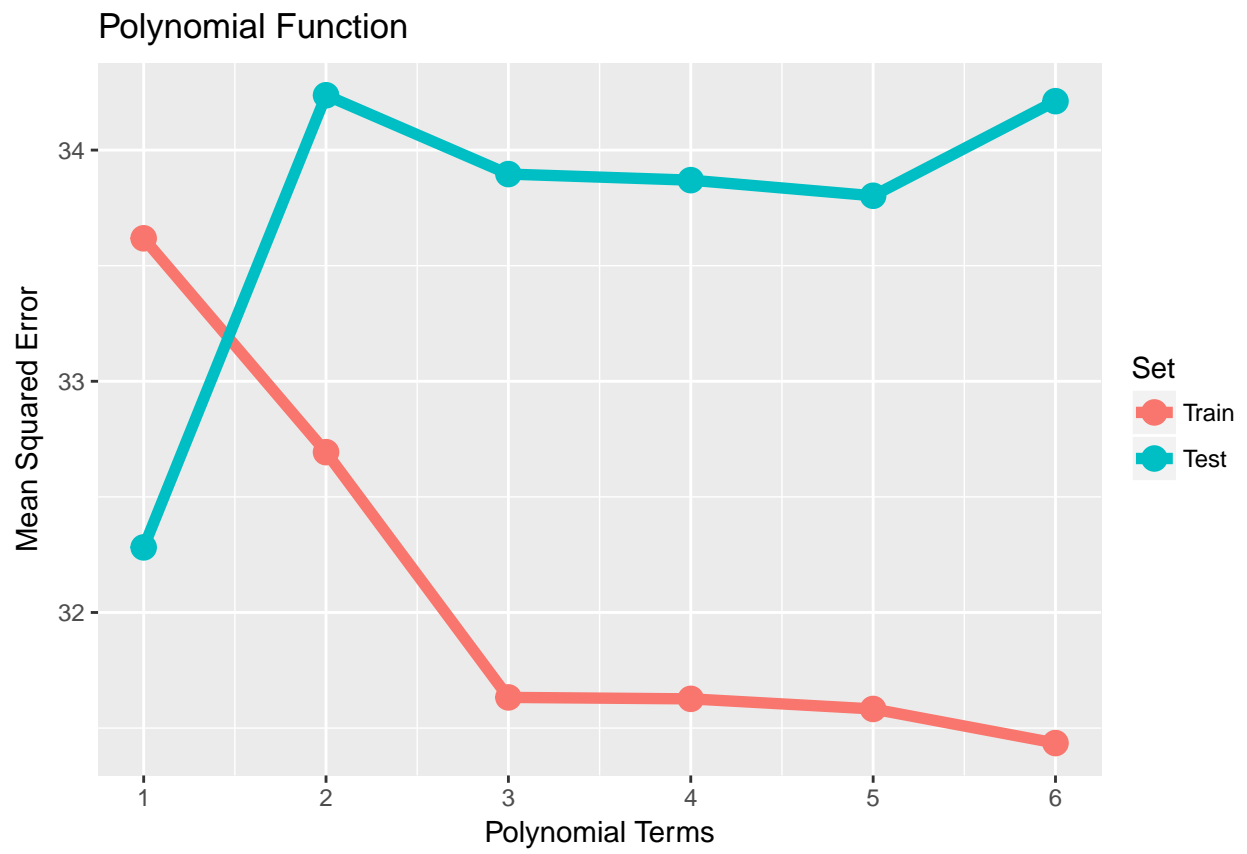


2

$$E[y(x, w)] = \sum_{i=0}^p w_i \phi_i(x) + \epsilon_i$$

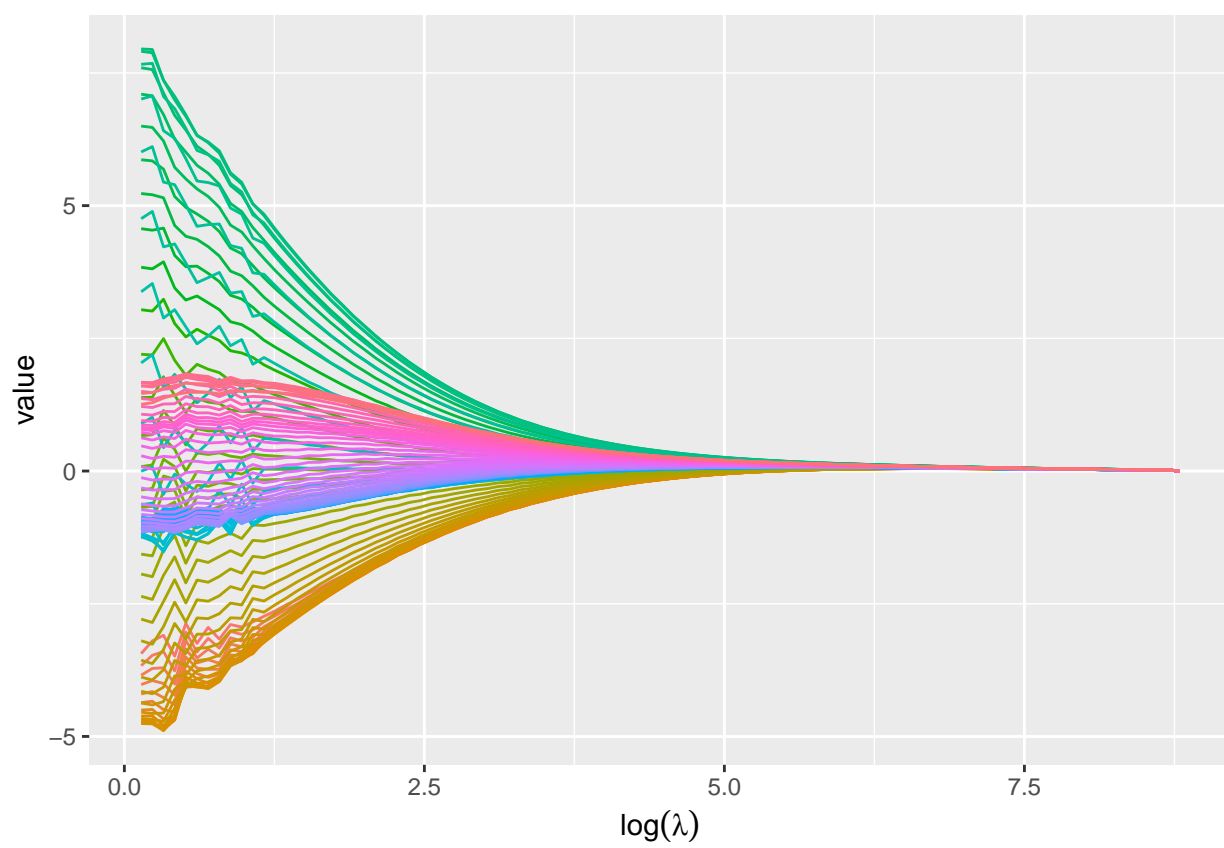
where $\phi_i(x) = x^i$. SEE Pattern Recognition 3.1.1

3

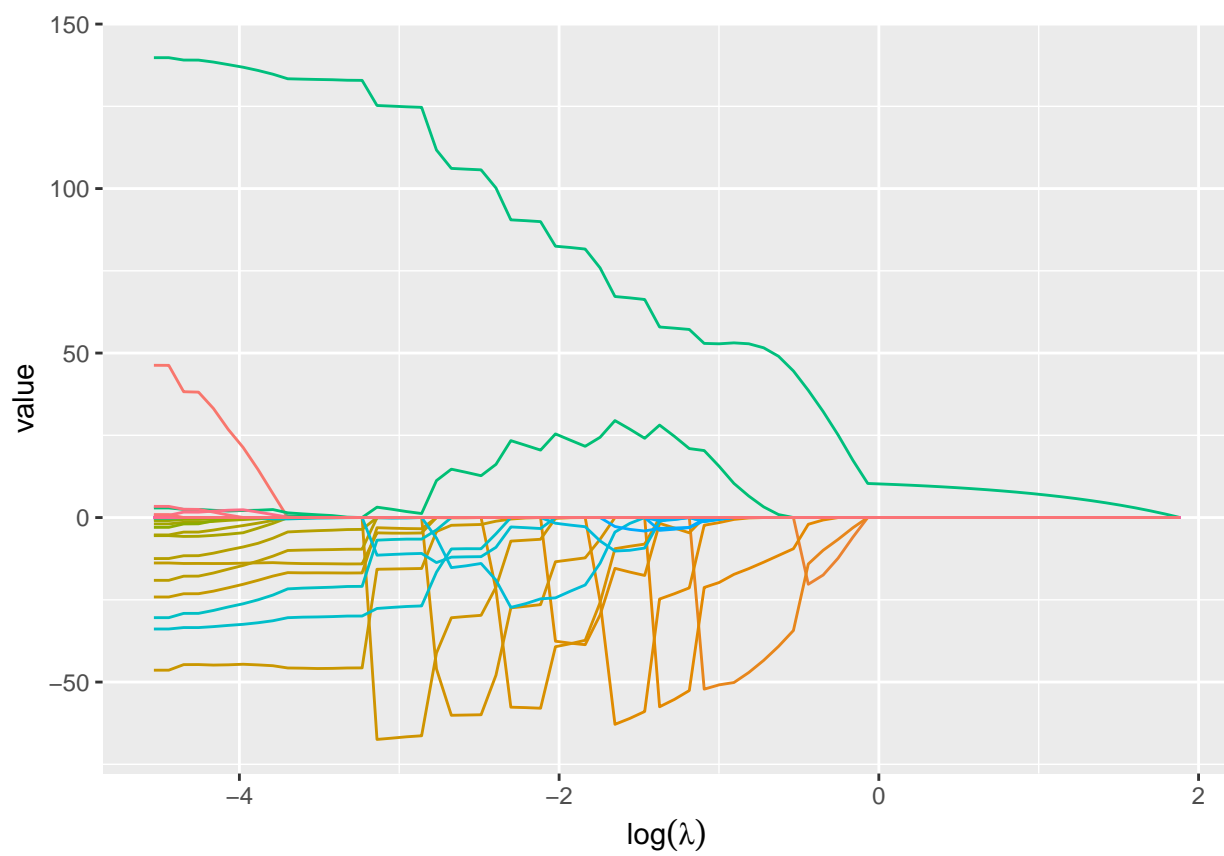


4

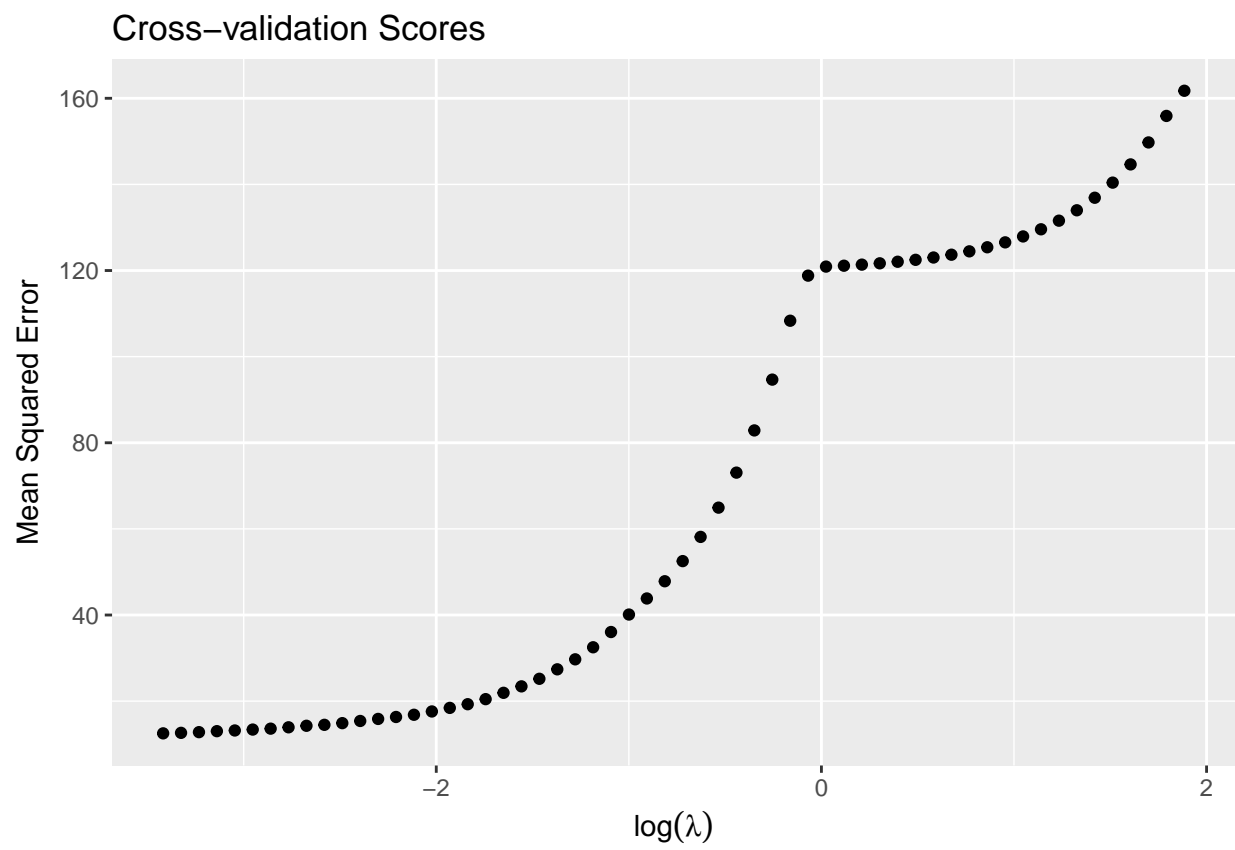
5



6



7



8

Appendix

Code for Assignment 1

```
library(ggplot2)

best_subset_selection <- function(X, y, folds) {
  n <- nrow(X)
  p <- ncol(X)

  stopifnot(folds <= n)

  sampled_idx <- sample(1:n, n)
  sets <- cross_validation_sets(n, folds)

  X <- X[sampled_idx,]
  y <- y[sampled_idx]

  best_features <- list()
  cv_scores <- rep(list(c()), p)

  for (j in 1:p) {
    feature_combinations <- combinations(p, j)
    errors <- c()
    for (feature_idx in 1:nrow(feature_combinations)) {
      features <- feature_combinations[feature_idx, ] == 1
      current_errors <- c()
      for (i in 1:folds) {
        test_validation_idx <- sets[i, 1]:sets[i, 2]
        test_idx <- test_validation_idx
        training_idx <- (1:n)[-test_validation_idx]

        lmfit <- linear_regression(as.matrix(X[training_idx, features]),
                                  y[training_idx],
                                  as.matrix(X[test_idx, features]),
                                  y[test_idx])
        current_errors <- c(current_errors, lmfit$SSE)
      }
      errors <- c(errors, mean(current_errors))
      cv_scores[[j]] <- c(cv_scores[[j]], mean(current_errors))
    }
    best_features[[j]] <- list(features=feature_combinations[which.min(errors),],
                               SSE=min(errors))
  }

  list(bf=best_features, cvs=cv_scores)
}

cross_validation_sets <- function(n, folds) {
  set_size <- as.integer(n / folds)
  remaining <- n - folds * set_size

  idx <- matrix(0, nrow=folds, ncol=2)
```

```

idx[1, 1] <- 1
idx[1, 2] <- set_size

for (i in 2:folds) {
  idx[i, 1] <- idx[i - 1, 2] + 1
  idx[i, 2] <- idx[i, 1] + (set_size - 1)

  if (remaining > 0) {
    idx[i, 2] <- idx[i, 2] + 1
    remaining <- remaining - 1
  }
}

idx
}

linear_regression <- function(X_train, y_train, X_test, y_test) {
  X_train <- cbind(rep(1, nrow(X_train)), X_train)
  X_test <- cbind(rep(1, nrow(X_test)), X_test)

  coefficients <- solve(t(X_train) %*% X_train) %*% t(X_train) %*% y_train
  coefficients <- as.vector(coefficients)
  fitted_values <- X_test %*% coefficients
  SSE <- sum((y_test - fitted_values)^2)
  list(coefficients=coefficients, fitted_values=fitted_values, SSE=SSE)
}

combinations <- function(n, m) {
  t(apply(combn(1:n, m=m), 2, function(x) replace(rep(0, n), x, 1)))
}

data <- swiss
x <- data[, -1]
y <- data[, 1]
folds <- 5
set.seed(12345)
result <- best_subset_selection(x, y, folds)
best_features <- result$bf
cv_scores <- result$cvs

best_setting <- best_features[[which.min(sapply(best_features, function(x) x$SSE))]]

lmfit <- linear_regression(as.matrix(x[, best_setting$features == 1]), y,
                          as.matrix(x[, best_setting$features == 1]), y)

## lmfit$coefficients
## colnames(x)[best_setting$features == 1]
coordinates <- lapply(1:length(cv_scores), function(feature_count) {
  cbind(x=feature_count, y=cv_scores[[feature_count]])
})
plot_data <- as.data.frame(do.call(rbind, coordinates))

best_coordinates <- lapply(1:length(cv_scores), function(feature_count) {

```

```

    cbind(x=feature_count, y=min(cv_scores[[feature_count]]))
  })
plot_line <- as.data.frame(do.call(rbind, best_coordinates))

ggplot(plot_data) +
  ggtitle("Cross-validation Errors") +
  xlab("Number of Features") +
  ylab("Sum of Squared Error") +
  geom_point(aes(x=x, y=y)) +
  geom_line(data=plot_line, aes(x=x, y=y), color="red") +
  theme(plot.title = element_text(hjust=0.5)) +
  scale_y_continuous(limits=c(0, max(plot_data$y)))

```

Code for Assignment 2

```

library(MASS)
library(glmnet)
library(readxl)
library(ggplot2)
library(Matrix)
library(reshape2)

data <- read_excel("../data/tecator.xlsx")
ggplot(data) +
  geom_point(aes(x=Protein, y=Moisture))
set.seed(12345)
n <- nrow(data)
training_idx <- sample(1:n, size=floor(n * 0.5))

train <- data[training_idx,]
test <- data[-training_idx,]

power <- 6

train_mse <- rep(0, power)
test_mse <- rep(0, power)

for (i in 1:power) {
  model <- lm(Moisture ~ poly(Protein, i), data=train)

  train_mse[i] <- mean((train$Moisture - predict(model, train))^2)
  test_mse[i] <- mean((test$Moisture - predict(model, test))^2)
}

plot_data <- data.frame(x=1:power, Train=train_mse, Test=test_mse)
plot_data <- melt(plot_data, id="x", variable.name="Set")

ggplot(plot_data) +
  geom_line(aes(x=x, y=value, color=Set), lwd=2) +
  geom_point(aes(x=x, y=value, color=Set), size=4) +
  ggtitle("Polynomial Function") +
  xlab("Polynomial Terms") + ylab("Mean Squared Error") +

```

```

    scale_x_continuous(breaks=1:power)
linear_model <- lm(Fat ~ . - Protein - Moisture - Sample, data=data)
aic <- stepAIC(linear_model, direction="both", trace=FALSE)
feature_selection_count <- length(aic$coefficients) - 1
response <- as.matrix(data[, setdiff(names(data), c("Sample", "Protein",
                                                    "Moisture", "Fat"))])

target <- data$Fat

## Ridge Regression
ridge_model <- glmnet(x=response, y=target, alpha=0, nlambda=100)
coefficients <- coef(ridge_model)
coefficients <- as.matrix(coefficients[-1,])
colnames(coefficients) <- ridge_model$lambda

plot_data <- melt(coefficients, id=rownames, varnames=c("feature", "lambda"))

ggplot(plot_data, aes(x=log(lambda), y=value, colour=feature)) +
  geom_line(show.legend=FALSE) +
  xlab(expression(log(lambda)))

## Lasso Regression
lasso_model <- glmnet(x=response, y=target, alpha=1, nlambda=100)
coefficients <- coef(lasso_model)
colnames(coefficients) <- lasso_model$lambda
coefficients <- as.matrix(coefficients[-1,])

plot_data <- melt(coefficients, id=rownames, varnames=c("feature", "lambda"))

ggplot(plot_data, aes(x=log(lambda), y=value, colour=feature)) +
  geom_line(show.legend=FALSE) +
  xlab(expression(log(lambda)))
lasso_model_cv <- cv.glmnet(response, target, alpha=1)
optimal_lambda <- lasso_model_cv$lambda.min
feature_selection_count <- sum(as.matrix(coef(lasso_model_cv)) != 0) - 1

plot_data <- data.frame(x=lasso_model_cv$lambda, y=lasso_model_cv$cvm)

ggplot(plot_data, aes(x=log(x), y=y)) + geom_point() +
  ggtitle("Cross-validation Scores") +
  xlab(expression(log(lambda))) +
  ylab("Mean Squared Error")

```