Training and Testing: (Linear) Predictions

```r
# Custom model complexity function
get_complexity <-  function(model) {
  return(length(coef(model)) - 1)
}

# Custom the function rmse
rmse <-  function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

1 Exercise 1

1.0.1 Q1

```r
# Read the data
data <- read.table("./data/ames1.txt", sep = "\t", header = T)
```

```r
# Eliminate OverallCond and OverallQual
library(tidyverse)
data1 <- data %>%
  select(-c(OverallCond, OverallQual))
```

1.0.2 Q2

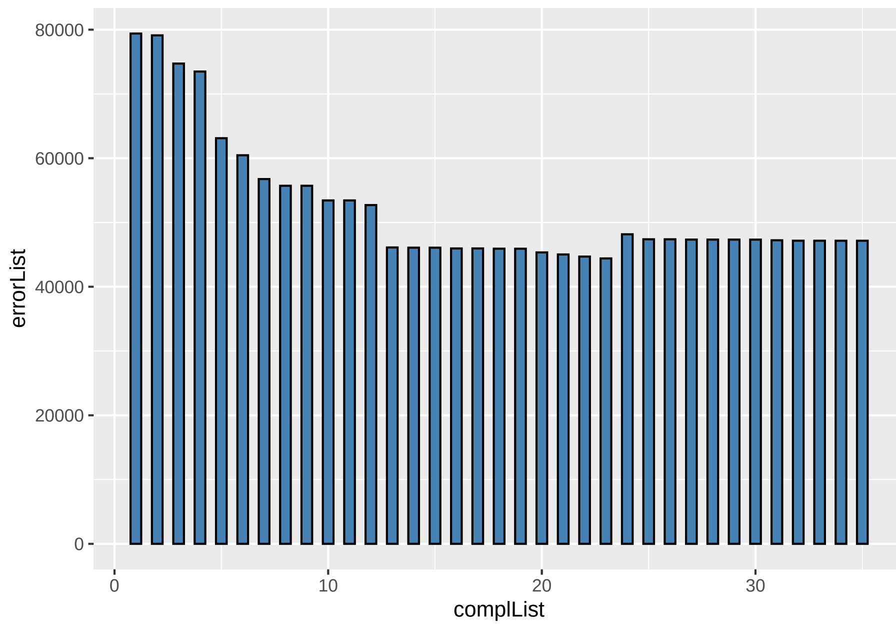
```r
# Highlight the data that we need
datalist <- data1[, -36]
datalabel <- data1[, 36]
index <- 1:ncol(datalist)

# Use the xxx to build the linear regression model
# and calculate the complexity
attributeList <- as.numeric()
model_list <- list()
for (i in seq(35)) {
  attributeList <- append(attributeList, i)
  data2 <- data.frame(datalist[, attributeList])
  lm.mod <- lm(datalabel ~ ., data = data2)
  if (get_complexity(lm.mod) < 16) {
      model_list <- append(model_list, lm.mod)
    }
}
```

1.0.3 Q3

```r
# Highlight the data
datalist <- data1[, -36]
datalabel <- data1[, 36]
index <- 1:ncol(datalist)

# Use the xxx to build the linear regression model
# and calculate the complexity and the value of rmse
attributeList <- as.numeric()
errorList <- as.numeric()
complList <- as.numeric()
for (i in seq(35)) {
```

```r
  attributeList <- append(attributeList, i)
  data2 <- data.frame(datalist[, attributeList])
  # build the linear regression model
  lm.mod <- lm(datalabel ~ ., data = data2)
  # prediction
  predicted <- predict(lm.mod, data2)
  predicted[is.na(predicted)] <- mean(na.omit(predicted))
  # calculate the rmse
  rmsError <- rmse(datalabel, predicted)
  errorList <- append(errorList, rmsError)
  # calculate the complexity
  complValue <- get_complexity(lm.mod)
  complList <- append(complList, complValue)
}
```

```r
# plot the complexity and rmse
df1 <- data.frame(complList, errorList)
library(tidyverse)
ggplot(df1, aes(complList, errorList)) +
  geom_bar(stat = 'identity', fill = 'steelblue',
           colour = 'black', width = 0.5)
```



2 Exercise 2

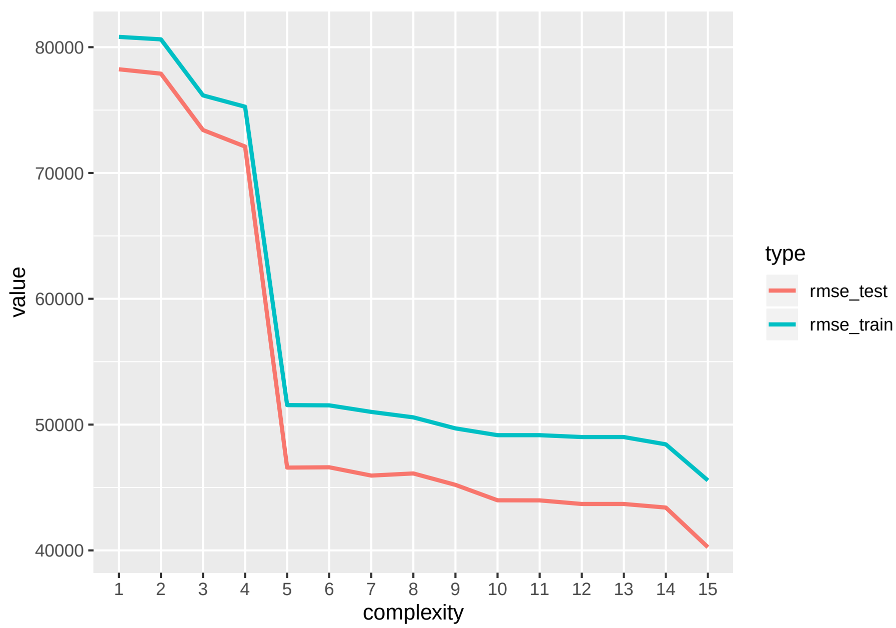
2.0.1 Q1

```
# Distinguish the training data set and test data set
set.seed(9)
Ames <- data
num_obs <-  nrow(Ames)
train_index <-  sample(num_obs, size = trunc(0.50 * num_obs))
train_data <-  Ames[train_index, ]
test_data <-  Ames[-train_index, ]

# Use the xxx to build the linear regression model
# and calculate the complexity
attributeList <- as.numeric()
model_list <- list()
complList <- as.numeric()
```

```r
rmselist_train <- as.numeric()
rmselist_test <- as.numeric()
for (i in seq(37)) {
  attributeList <- append(attributeList, i)
  data2 <- data.frame(train_data[, attributeList])
  colnames(data2)[1] <- c("Id")
  data3 <- data.frame(test_data[, attributeList])
  colnames(data3)[1] <- c("Id")
  lm.mod <- lm(train_data[, 38] ~ ., data = data2)
  if (get_complexity(lm.mod) < 16) {
      complValue <- get_complexity(lm.mod)
      complList <- append(complList, complValue)

      predicted_train <- predict(lm.mod, data2)
      predicted_train[is.na(predicted_train)] <-
        mean(na.omit(predicted_train))
      rmse_train <- rmse(train_data[, 38], predicted_train)
      rmselist_train <- append(rmselist_train, rmse_train)

      predicted_test <- predict(lm.mod, data3)
      predicted_test[is.na(predicted_test)] <-
        mean(na.omit(predicted_test))
      rmse_test <- rmse(test_data[, 38], predicted_test)
      rmselist_test <- append(rmselist_test, rmse_test)
  }
}
```

```r
# Plot the train rmse and the test rmse
df2 <- data.frame(complList, rmselist_train, rmselist_test)
colnames(df2) <- c("complexity", "rmse_train", "rmse_test")
library(tidyverse)
df3 <- df2 %>%
  gather("rmse_train", "rmse_test",
```

```
         key = "type", value = "value")
ggplot(df3, aes(x = factor(complexity), y = value,
                colour = type,  group = type)) +
  geom_line(size = 1) +
  xlab("complexity")
```



2.0.2 Q2

```
# Distinguish the training data set and test data set
set.seed(9)
Ames <- data
num_obs <-  nrow(Ames)
train_index <-  sample(num_obs, size = trunc(0.50 * num_obs))
train_data <-  Ames[train_index, ]
test_data <-  Ames[-train_index, ]

# Use the forward selection to build the linear regression model
# and calculate the complexity and the value of rmse
```

```
attributeList <- as.numeric()
model_list <- list()
complList <- as.numeric()
rmselist_train <- as.numeric()
rmselist_test <- as.numeric()
for (i in seq(37)) {
  attributeList <- append(attributeList, i)
  data2 <- data.frame(train_data[, attributeList])
  colnames(data2)[1] <- c("Id")
  data3 <- data.frame(test_data[, attributeList])
  colnames(data3)[1] <- c("Id")
  lm.mod <- lm(train_data[, 38] ~ ., data = data2)
  if (get_complexity(lm.mod) == 15) {

      predicted_train <- predict(lm.mod, data2)
      predicted_train[is.na(predicted_train)] <-
        mean(na.omit(predicted_train))
      rmse_train <- rmse(train_data[, 38], predicted_train)
      rmselist_train <- append(rmselist_train, rmse_train)

      predicted_test <- predict(lm.mod, data3)
      predicted_test[is.na(predicted_test)] <-
        mean(na.omit(predicted_test))
      rmse_test <- rmse(test_data[, 38], predicted_test)
      rmselist_test <- append(rmselist_test, rmse_test)
  }
}
```

```
# Predict the train rmse and test rmse of SalePrice
df4 <- data.frame(predicted_train, predicted_test)
head(df4)
```

```
##     predicted_train predicted_test
## 187        179972.4       180977.7
```

```
## 1286        142081.7       176593.5
## 408         156160.2       289779.0
## 595         113676.6       157534.2
## 1027        148603.3       283152.3
## 816         250541.1       181850.7
```

```
# Calculate the error of test and train of regression line
# modle when the complexity is 15
df5 <- data.frame(rmselist_train, rmselist_test)
df5
```

```
##   rmselist_train rmselist_test
## 1       45564.62      40257.16
```

2.0.3 Q3

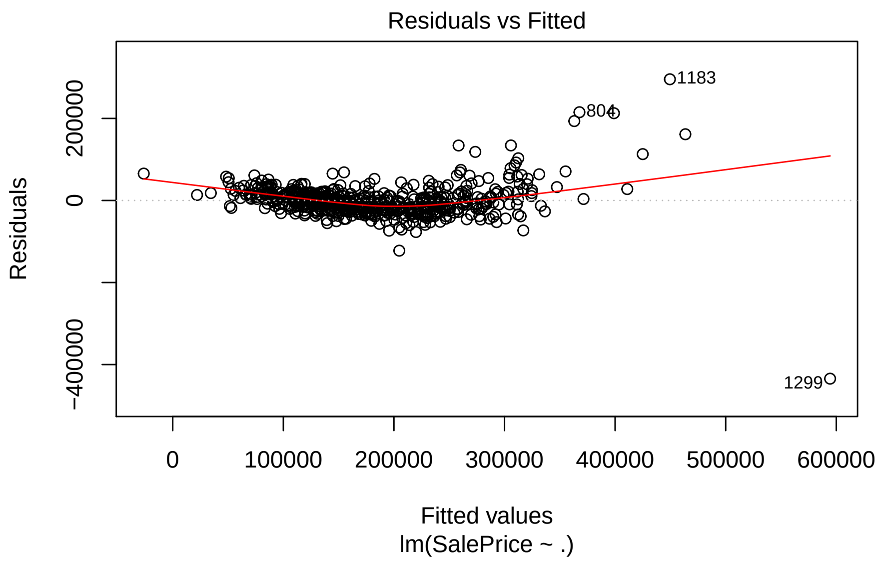
```
# Distribute the training data set and test data set
set.seed(9)
Ames <- data
num_obs <-  nrow(Ames)
train_index <-  sample(num_obs, size = trunc(0.50 * num_obs))
train_data <-  Ames[train_index, ]
test_data <-  Ames[-train_index, ]

# Using forward selection and according to the lowest
# rmse to get the best model
attributeList <- as.numeric()
rmse_min <- 100000000
attributeList1 <- as.numeric()
for (i in seq(37)) {
  attributeList <- append(attributeList, i)
  data2 <- data.frame(train_data[, attributeList])
  colnames(data2)[1] <- c("Id")
  data3 <- data.frame(test_data[, attributeList])
```

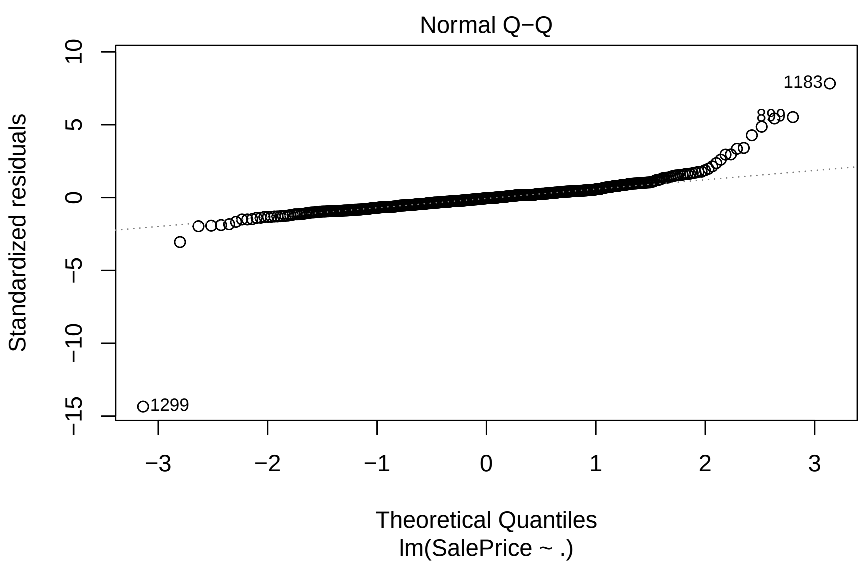
```
  colnames(data3)[1] <- c("Id")
  lm.mod <- lm(train_data[, 38] ~ ., data = data2)

  predicted_test <- predict(lm.mod, data3)
  predicted_test[is.na(predicted_test)] <-
    mean(na.omit(predicted_test))
  rmse_test <- rmse(test_data[, 38], predicted_test)
  if (rmse_test < rmse_min) {
    rmse_min <- rmse_test
    col_index <- c(colnames(data2)[attributeList], "SalePrice")
  }
}
```

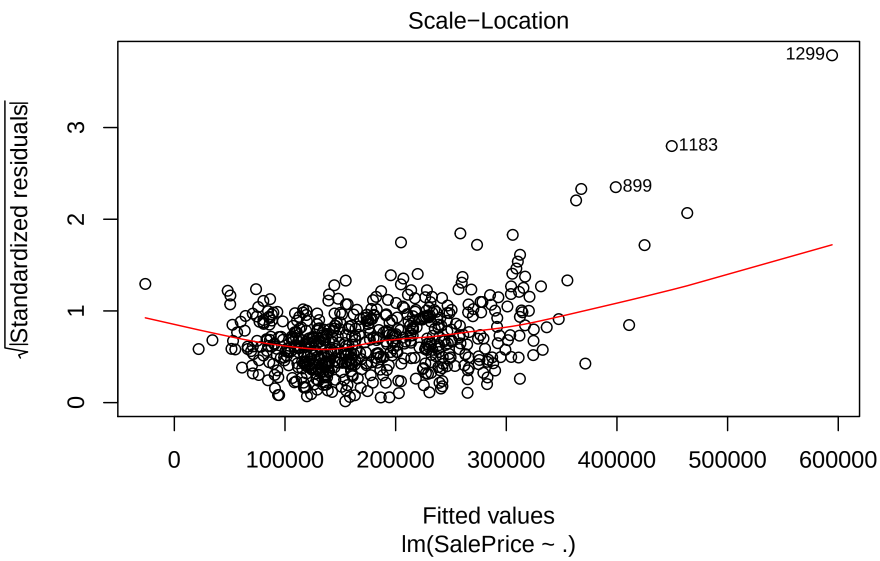
```
#Get the optimal linear regression model based on the
# test set error minimization principle
resulting_model <- lm(SalePrice~., train_data[, col_index])
# summary(resulting_model)
```

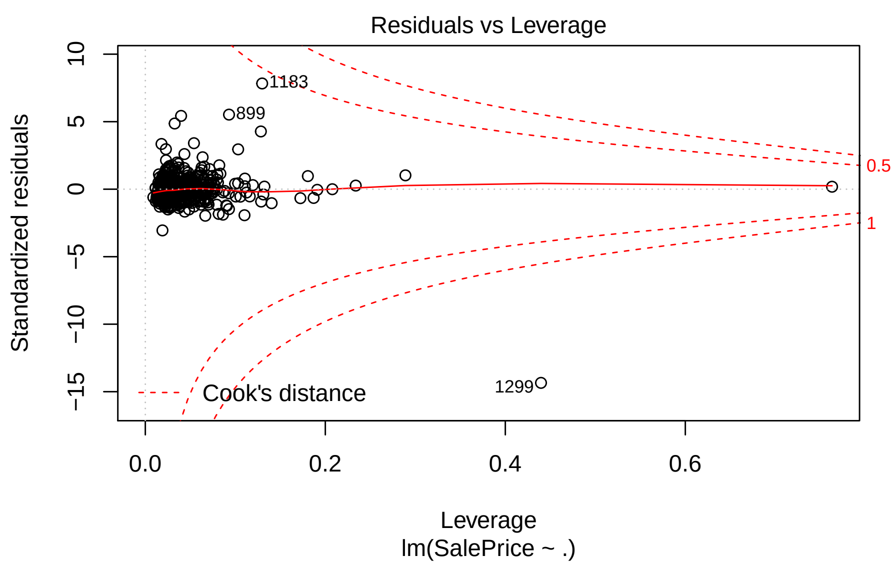
2.0.4 Q4

```
# Regression diagnosis on the optimal linear regression model
# 1residual graph and  t plot( linear test),
# 2QQgraph(data nornal test)
# 3posiition scale chart(testing for homoscedasticity)
# 4risidual and levergae chart(testing for outliers)
plot(resulting_model)
```


Residuals vs Fitted
Residuals
200000
0
−400000
0
100000
200000
300000
400000
500000
600000
Fitted values
lm(SalePrice ~ .)
1183
804
1299


Normal Q−Q
Standardized residuals
10
5
0
−5
−10
−15
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(SalePrice ~ .)
1183
1299


Scale−Location
1299
1183
899
√|Standardized residuals|
0
1
2
3
0
100000
200000
300000
400000
500000
600000
Fitted values
lm(SalePrice ~ .)


Residuals vs Leverage
1183
899
1299
Cook's distance
0.5
1
Standardized residuals
10
5
0
−5
−10
−15
0.0
0.2
0.4
0.6
Leverage
lm(SalePrice ~ .)