Swinging for the Fences: A Data-Driven Approach to Predicting Baseball Player Salaries with the Hitters Dataset in R

Roshan Mehta

2023-03-12

Contents

Introduction

The goal of this project is to build a model that can accurately predict a player's salary based on their performance statistics. To accomplish this, we will be using a variety of machine learning methods and

functions, after data cleaning and preparation, exploratory data analysis, feature engineering, and predictive modeling.

This project will be using the "Hitters" dataset, available in R, which contains information on the performance of Major League Baseball (MLB) hitters. The dataset contains statistics on 322 players who played at least one season in the MLB during the years 1986-87, including their salary, batting average, runs, hits, home runs, stolen bases, and other performance metrics.

We will start by cleaning and preparing the Hitters dataset to remove any missing or irrelevant data, and converting categorical variables to factors. Then, we will conduct exploratory data analysis to gain insights into the performance statistics of MLB hitters, using descriptive statistics, visualizations, and correlations to understand the relationships between different variables.

Next, we will create new features based on the existing variables in the Hitters dataset. We will use domain knowledge and statistical techniques to engineer features that may improve the accuracy of our model.

Finally, we will build and evaluate different models to predict a player's salary based on their performance statistics. We will use a train-test split to evaluate the accuracy of our models and select the best model based on its performance. Some of the models we will use include linear regression, ridge regression, and decision trees.

By the end of this project, we will have a better understanding of the performance statistics of Major League Baseball hitters, as well as a predictive model that can be used by MLB teams and analysts to estimate the salaries of prospective players based on their performance statistics.

Data Preparation and Cleaning

The "Hitters" dataset is available in the "ISLR2" package in R, and can be loaded using the `data(Hitters)` command after loading the library. In this section, we will prepare and clean the Hitters dataset for analysis. We will remove any missing or irrelevant data and convert any necessary data types.

```
library(ISLR2)
data(Hitters)

# Remove any missing data
Hitters <- na.omit(Hitters)

# Convert categorical variables to factors
Hitters$League <- as.factor(Hitters$League)
Hitters$Division <- as.factor(Hitters$Division)
Hitters$NewLeague <- as.factor(Hitters$NewLeague)

# View the cleaned dataset
head(Hitters)
```

```
##                   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Alan Ashby         315   81     7   24  38    39    14   3449   835      69
## -Alvin Davis        479  130    18   66  72    76     3   1624   457      63
## -Andre Dawson       496  141    20   65  78    37    11   5628  1575     225
## -Andres Galarraga   321   87    10   39  42    30     2    396   101      12
## -Alfredo Griffin    594  169     4   74  51    35    11   4408  1133      19
## -Al Newman          185   37     1   23   8    21     2    214    42       1
##                   CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Alan Ashby         321  414    375      N        W     632      43     10
## -Alvin Davis        224  266    263      A        W     880      82     14
```

```
## -Andre Dawson       828  838    354      N        E     200      11      3
## -Andres Galarraga    48   46     33      N        E     805      40      4
## -Alfredo Griffin    501  336    194      A        W     282     421     25
## -Al Newman           30    9     24      N        E      76     127      7
##                    Salary NewLeague
## -Alan Ashby         475.0         N
## -Alvin Davis        480.0         A
## -Andre Dawson       500.0         N
## -Andres Galarraga    91.5         N
## -Alfredo Griffin    750.0         A
## -Al Newman           70.0         A
```

The dataset contains 322 observations (i.e., players) and 20 variables (i.e., attributes). The variables in the dataset include:

- `AtBat`: Number of times at bat
- `Hits`: Number of hits
- `HmRun`: Number of home runs
- `Runs`: Number of runs
- `RBI`: Number of runs batted in
- `Walks`: Number of walks
- `Years`: Number of years in the major leagues
- `CAtBat`: Number of times at bat during his career
- `CHits`: Number of hits during his career
- `CHmRun`: Number of home runs during his career
- `CRuns`: Number of runs during his career
- `CRBI`: Number of runs batted in during his career
- `CWalks`: Number of walks during his career
- `League`: A factor with levels A and N indicating the player's league at the end of 1986 (American or National)
- `Division`: A factor with levels E and W indicating the player's division at the end of 1986 (East or West)
- `PutOuts`: Number of putouts
- `Assists`: Number of assists
- `Errors`: Number of errors
- `Salary`: 1987 annual salary on opening day in thousands of dollars
- `NewLeague`: A factor with levels A and N indicating the player's league at the beginning of 1987 **Variables pertain to the 1986 season were applicable**

Exploratory Data Analysis (EDA)

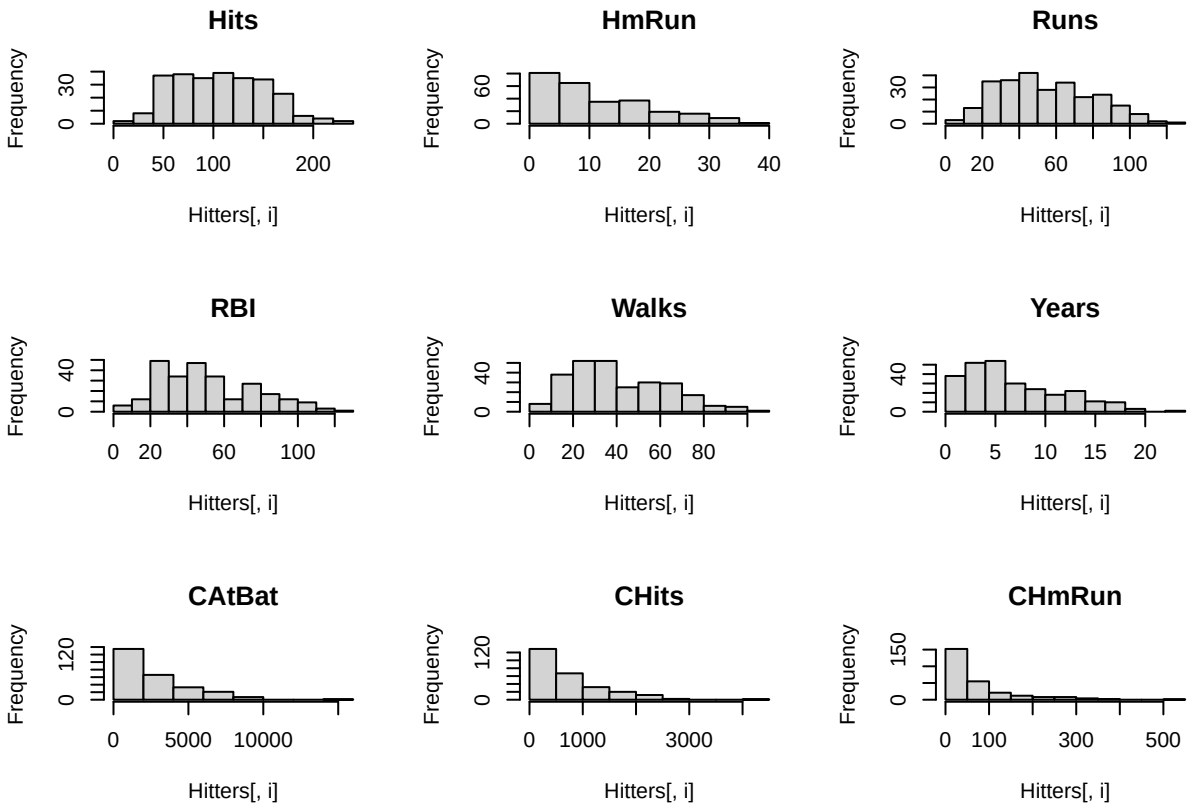
In this section, we will explore the Hitters dataset to gain insights into the performance statistics of MLB hitters. We will use descriptive statistics, visualizations, and correlations to understand the relationships between different variables.

```
# View the summary statistics of the dataset
summary(Hitters)
```

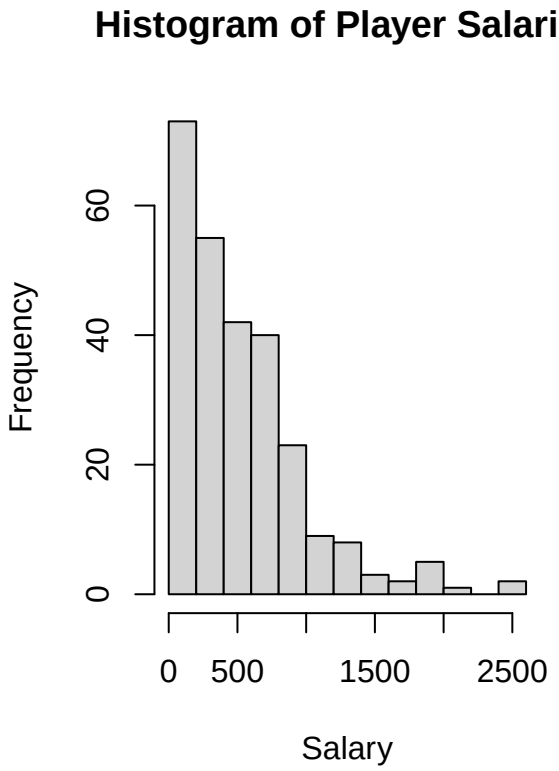
```
##      AtBat            Hits           HmRun            Runs
##  Min.   : 19.0   Min.   :  1.0   Min.   : 0.00   Min.   :  0.00
##  1st Qu.:282.5   1st Qu.: 71.5   1st Qu.: 5.00   1st Qu.: 33.50
```

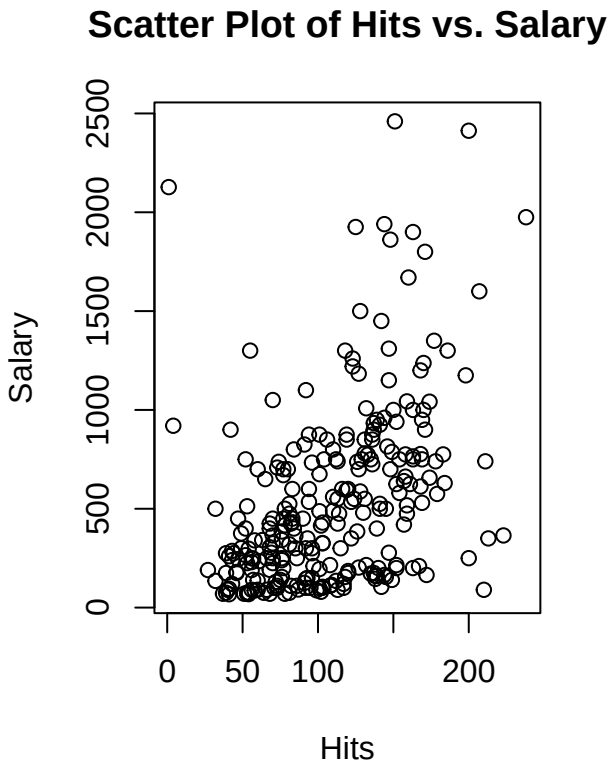
```
##  Median :413.0   Median :103.0   Median : 9.00   Median : 52.00
##  Mean   :403.6   Mean   :107.8   Mean   :11.62   Mean   : 54.75
##  3rd Qu.:526.0   3rd Qu.:141.5   3rd Qu.:18.00   3rd Qu.: 73.00
##  Max.   :687.0   Max.   :238.0   Max.   :40.00   Max.   :130.00
##       RBI             Walks            Years            CAtBat
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0
##  1st Qu.: 30.00   1st Qu.: 23.00   1st Qu.: 4.000   1st Qu.:  842.5
##  Median : 47.00   Median : 37.00   Median : 6.000   Median : 1931.0
##  Mean   : 51.49   Mean   : 41.11   Mean   : 7.312   Mean   : 2657.5
##  3rd Qu.: 71.00   3rd Qu.: 57.00   3rd Qu.:10.000   3rd Qu.: 3890.5
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
##      CHits            CHmRun           CRuns             CRBI
##  Min.   :   4.0   Min.   :  0.00   Min.   :   2.0   Min.   :   3.0
##  1st Qu.: 212.0   1st Qu.: 15.00   1st Qu.: 105.5   1st Qu.:  95.0
##  Median : 516.0   Median : 40.00   Median : 250.0   Median : 230.0
##  Mean   : 722.2   Mean   : 69.24   Mean   : 361.2   Mean   : 330.4
##  3rd Qu.:1054.0   3rd Qu.: 92.50   3rd Qu.: 497.5   3rd Qu.: 424.5
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.0
##      CWalks       League  Division    PutOuts          Assists
##  Min.   :   1.0   A:139   E:129    Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  71.0   N:124   W:134    1st Qu.: 113.5   1st Qu.:  8.0
##  Median : 174.0                    Median : 224.0   Median : 45.0
##  Mean   : 260.3                    Mean   : 290.7   Mean   :118.8
##  3rd Qu.: 328.5                    3rd Qu.: 322.5   3rd Qu.:192.0
##  Max.   :1566.0                    Max.   :1377.0   Max.   :492.0
##      Errors           Salary       NewLeague
##  Min.   : 0.000   Min.   :  67.5   A:141
##  1st Qu.: 3.000   1st Qu.: 190.0   N:122
##  Median : 7.000   Median : 425.0
##  Mean   : 8.593   Mean   : 535.9
##  3rd Qu.:13.000   3rd Qu.: 750.0
##  Max.   :32.000   Max.   :2460.0
```

```r
# Visualize the distributions of the variables
par(mfrow=c(3,3))
for(i in 2:10) {
  hist(Hitters[,i], main=colnames(Hitters)[i])
}
```

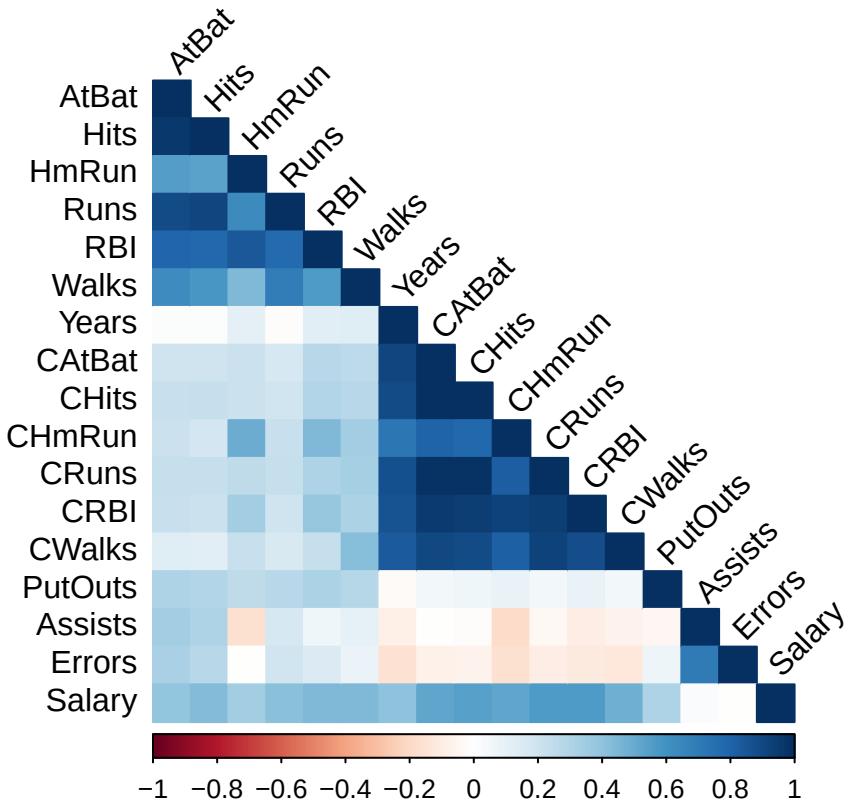



```
par(mfrow=c(1,2))
# Create a histogram of the Salary variable
hist(Hitters$Salary, main="Histogram of Player Salaries", xlab="Salary")

# Create a scatter plot of Hits vs. Salary
plot(Hitters$Hits, Hitters$Salary, main="Scatter Plot of Hits vs. Salary", xlab="Hits", ylab="Salary")
```





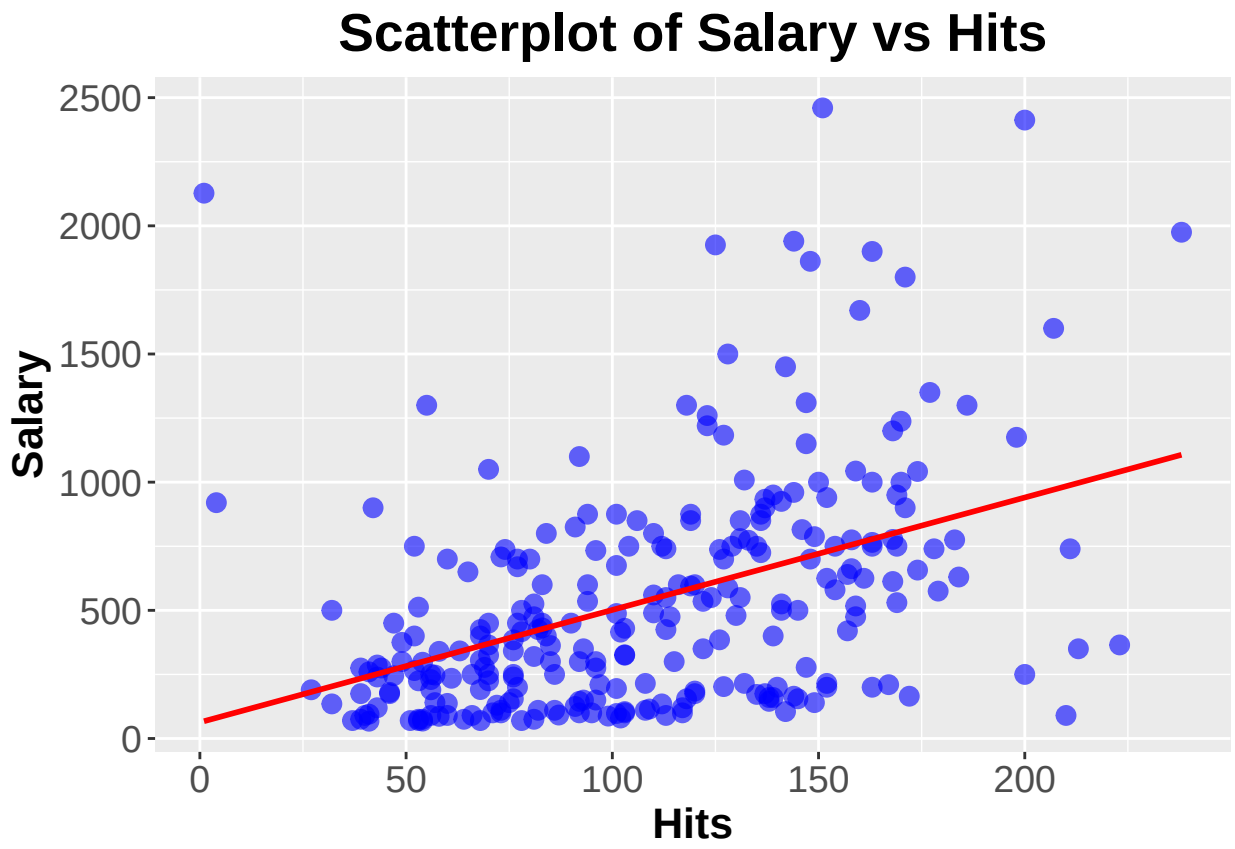

```r
par(mfrow=c(1,1))
# Compute correlation matrix
corr_matrix <- cor(select_if(Hitters, is.numeric))
# Create correlation plot
corrplot(corr_matrix, method="color", type="lower", tl.col="black", tl.srt=45)
```



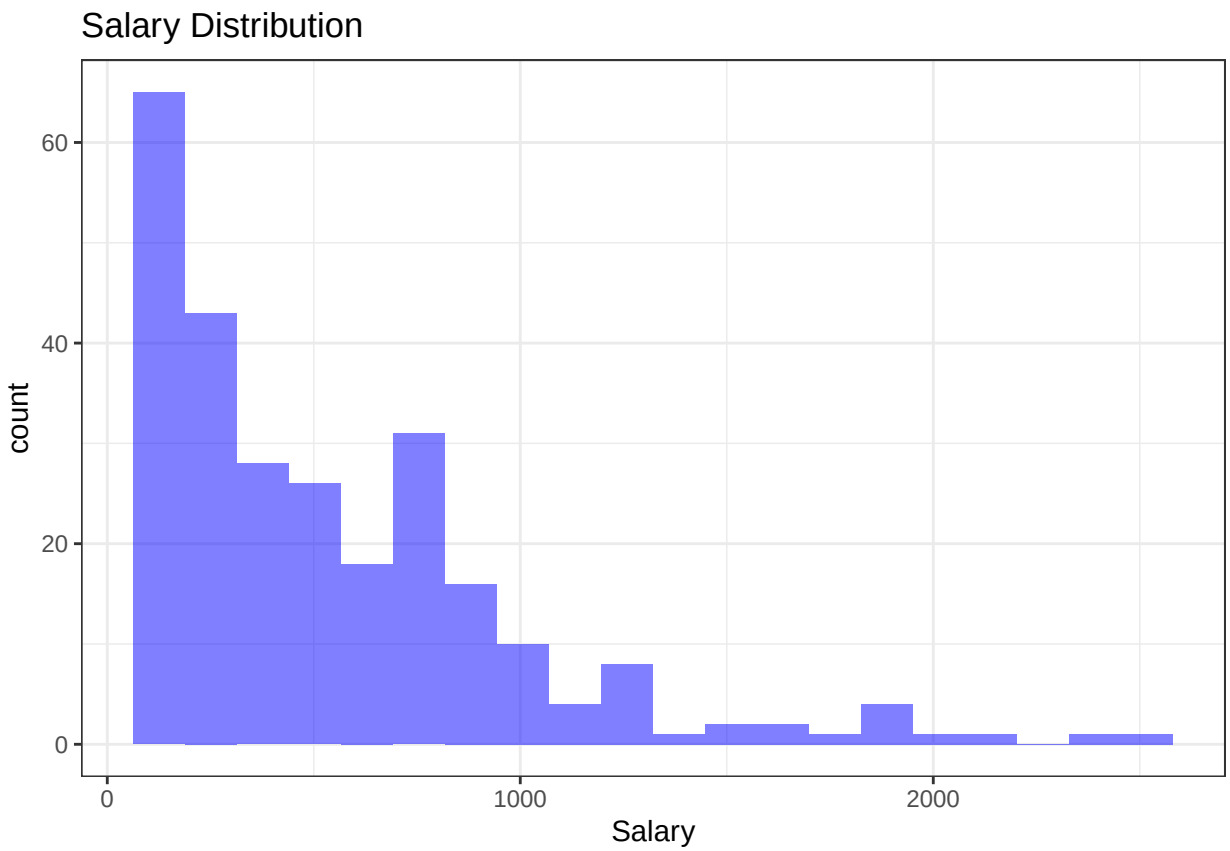
There seem to be strong positive linear correlations between `Years`, `CHits`, `CRuns`, `CRbi`, and `CWalks`. There are also strong positive correlations between `AtBat`, `Hits`, `Runs`, `RBI`, and `Walks`.

```
# Scatterplot of Salary vs Hits
ggplot(data = Hitters, aes(x = Hits, y = Salary)) +
  geom_point(color = "blue", alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Scatterplot of Salary vs Hits",
       x = "Hits",
       y = "Salary") +
  theme(plot.title = element_text(face = "bold", size = 20, hjust = 0.5),
        axis.title = element_text(face = "bold", size = 16),
        axis.text = element_text(size = 14))
```

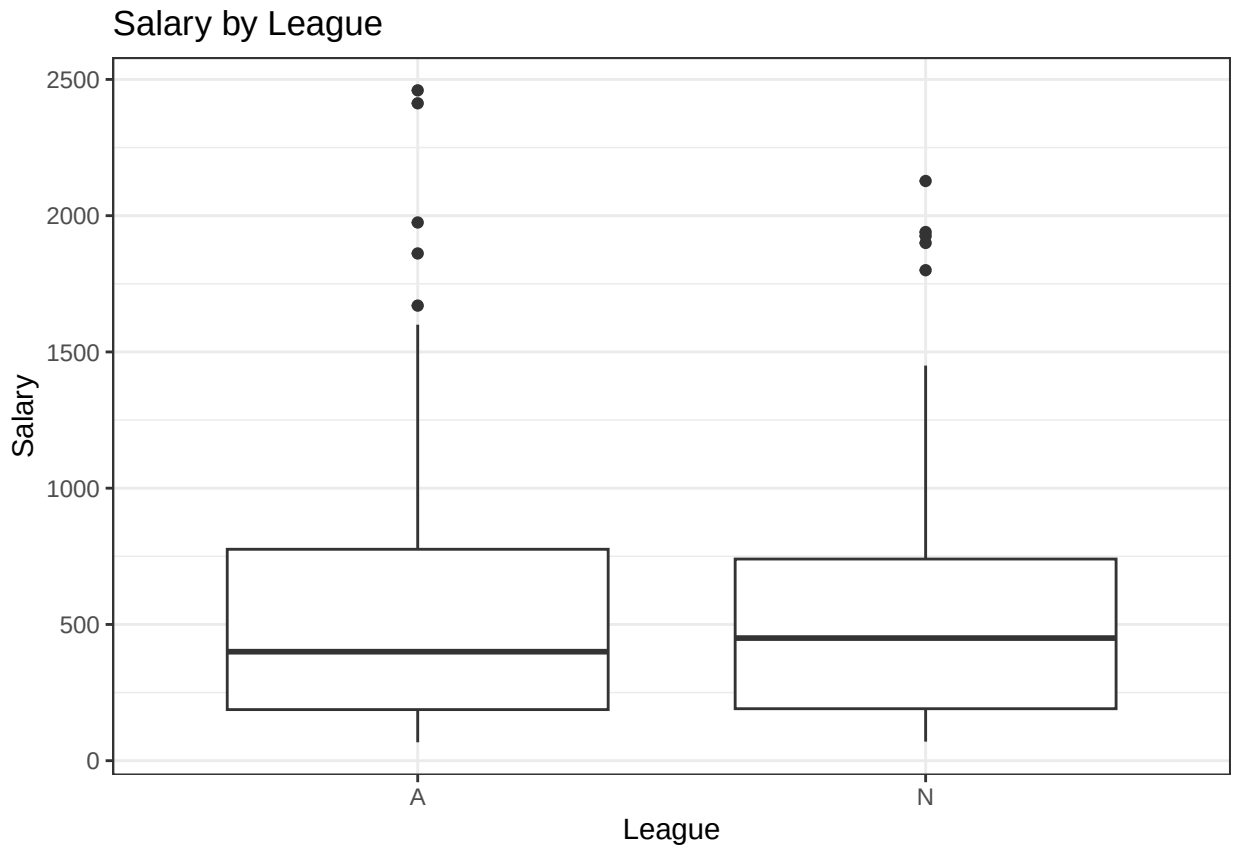
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# salary histogram
ggplot(data = Hitters, aes(x = Salary)) +
  geom_histogram(fill = "blue", alpha = 0.5, bins = 20) +
  ggtitle("Salary Distribution") +
  theme_bw()
```




```
# Boxplot of salary by league
ggplot(data = Hitters, aes(x = League, y = Salary)) +
  geom_boxplot() +
  ggtitle("Salary by League") +
  theme_bw()
```



Feature Selection and Engineering

In this section, we will create new features based on the existing variables in the Hitters dataset. We will use domain knowledge and statistical techniques to engineer features that may improve the accuracy of our model.

```
# Create a new feature for the player's average number of hits per year
Hitters$AvgHits <- Hitters$Hits / Hitters$Years

# Create a new feature for the player's average number of home runs per year
Hitters$AvgHR <- Hitters$HmRun / Hitters$Years

# Create a new feature for the player's average number of runs per year
Hitters$AvgRuns <- Hitters$Runs / Hitters$Years

# View the updated dataset
head(Hitters[,19:23])
```

```
##                   Salary NewLeague   AvgHits     AvgHR   AvgRuns
## -Alan Ashby        475.0         N  5.785714 0.5000000  1.714286
## -Alvin Davis       480.0         A 43.333333 6.0000000 22.000000
## -Andre Dawson      500.0         N 12.818182 1.8181818  5.909091
## -Andres Galarraga   91.5         N 43.500000 5.0000000 19.500000
## -Alfredo Griffin   750.0         A 15.363636 0.3636364  6.727273
```

```
## -Al Newman          70.0         A 18.500000 0.5000000 11.500000
```

Building Predictive Models & Evaluaton

In this section, we will evaluate the performance of the different models we built in the previous section. We will use a train-test split to evaluate the accuracy of each model and select the best model based on its performance.

Train-Test Split

Before we evaluate the models, we will split the data into training and test sets. We will use the training set to build the models and the test set to evaluate their performance.

Best Subset Selection

Forward and Backward Stepwise Selection

```
library(leaps)
regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=22)
reg.summary=summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
reg.summary$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4772277 0.5201447 0.5465191 0.5616910 0.5765308
##  [8] 0.5870461 0.5953759 0.6020744 0.6035141 0.6058038 0.6075500 0.6084407
## [15] 0.6096371 0.6098700 0.6100855 0.6102114 0.6102974 0.6103240 0.6103338
## [22] 0.6103398
```

```
regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=22,method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 22, method = "forward")
## 22 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
```

```
## CHmRun          FALSE      FALSE
## CRuns           FALSE      FALSE
## CRBI            FALSE      FALSE
## CWalks          FALSE      FALSE
## LeagueN         FALSE      FALSE
## DivisionW       FALSE      FALSE
## PutOuts         FALSE      FALSE
## Assists         FALSE      FALSE
## Errors          FALSE      FALSE
## NewLeagueN      FALSE      FALSE
## AvgHits         FALSE      FALSE
## AvgHR           FALSE      FALSE
## AvgRuns         FALSE      FALSE
## 1 subsets of each size up to 22
## Selection Algorithm: forward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 4  ( 1 )  " "   "*"  " "   " "  " " " " " "   "*"   " "   " "    " "   " "   "*"
## 5  ( 1 )  " "   "*"  " "   " "  " " " " " "   "*"   " "   " "    " "   " "   "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " " " " "   "*"   " "   " "    " "   " "   "*"
## 7  ( 1 )  "*"   "*"  " "   " "  " " " " " "   "*"   " "   " "    " "   "*"   "*"
## 8  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "    " "   "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "    " "   "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "    " "   "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   " "    " "   "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   " "    " "   "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"    " "   "*"   "*"
## 14  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"    " "   "*"   "*"
## 15  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"    " "   "*"   "*"
## 16  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"    " "   "*"   "*"
## 17  ( 1 ) "*"   "*"  " "   " "  "*" "*"  "*"   "*"   "*"   "*"    " "   "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   " "  "*" "*"  "*"   "*"   "*"   "*"    " "   "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   " "  "*" "*"  "*"   "*"   "*"   "*"    "*"   "*"   "*"
## 20  ( 1 ) "*"   "*"  "*"   " "  "*" "*"  "*"   "*"   "*"   "*"    "*"   "*"   "*"
## 21  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"  "*"   "*"   "*"   "*"    "*"   "*"   "*"
## 22  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"  "*"   "*"   "*"   "*"    "*"   "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN AvgHits
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "        " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "        " "
## 3  ( 1 )  " "    " "     " "       " "     " "     " "    " "        "*"
## 4  ( 1 )  " "    " "     " "       " "     " "     " "    " "        "*"
## 5  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 6  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 7  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 8  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 9  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "        "*"
## 10  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "        "*"
## 11  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "        "*"
## 12  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "        "*"
## 13  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "        "*"
## 14  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 15  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
```

```
## 16  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 17  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 18  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 19  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 20  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"        "*"
## 21  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"        "*"
## 22  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"        "*"
##           AvgHR AvgRuns
## 1  ( 1 )  " "   " "
## 2  ( 1 )  " "   " "
## 3  ( 1 )  " "   " "
## 4  ( 1 )  " "   " "
## 5  ( 1 )  " "   " "
## 6  ( 1 )  " "   " "
## 7  ( 1 )  " "   " "
## 8  ( 1 )  " "   " "
## 9  ( 1 )  " "   " "
## 10  ( 1 ) " "   " "
## 11  ( 1 ) " "   " "
## 12  ( 1 ) " "   " "
## 13  ( 1 ) " "   " "
## 14  ( 1 ) " "   " "
## 15  ( 1 ) "*"   " "
## 16  ( 1 ) "*"   "*"
## 17  ( 1 ) "*"   "*"
## 18  ( 1 ) "*"   "*"
## 19  ( 1 ) "*"   "*"
## 20  ( 1 ) "*"   "*"
## 21  ( 1 ) "*"   "*"
## 22  ( 1 ) "*"   "*"
```

```
regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=22,method="backward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 22, method = "backward")
## 22 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
```

```
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## AvgHits        FALSE      FALSE
## AvgHR          FALSE      FALSE
## AvgRuns        FALSE      FALSE
## 1 subsets of each size up to 22
## Selection Algorithm: backward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " " " "   " "   " "   " "   " "   " "   "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "   " "   " "   "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "   " "   " "   "*"
## 4  ( 1 )  " "   "*"  " "   " "  " " " " " "   "*"   " "   " "   " "   " "   "*"
## 5  ( 1 )  " "   "*"  " "   " "  " " " " " "   "*"   " "   " "   " "   " "   "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " " " " "   "*"   " "   " "   " "   " "   "*"
## 7  ( 1 )  "*"   "*"  " "   " "  " " " " " "   "*"   " "   " "   " "   "*"   "*"
## 8  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "   " "   "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "   " "   "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   " "   " "   " "   "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   " "   " "   "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   " "   " "   "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"   " "   "*"   "*"
## 14  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"   " "   "*"   "*"
## 15  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"   " "   "*"   "*"
## 16  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   "*"   "*"   "*"   " "   "*"   "*"
## 17  ( 1 ) "*"   "*"  " "   " "  "*" "*" "*"   "*"   "*"   "*"   " "   "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   " "  "*" "*" "*"   "*"   "*"   "*"   " "   "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   " "  "*" "*" "*"   "*"   "*"   "*"   "*"   "*"   "*"
## 20  ( 1 ) "*"   "*"  "*"   " "  "*" "*" "*"   "*"   "*"   "*"   "*"   "*"   "*"
## 21  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"   "*"   "*"   "*"   "*"
## 22  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"   "*"   "*"   "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN AvgHits
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "        " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "        " "
## 3  ( 1 )  " "    " "     " "       " "     " "     " "    " "        "*"
## 4  ( 1 )  " "    " "     " "       " "     " "     " "    " "        "*"
## 5  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 6  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 7  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 8  ( 1 )  " "    " "     " "       "*"     " "     " "    " "        "*"
## 9  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "        "*"
## 10  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "        "*"
## 11  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "        "*"
## 12  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "        "*"
## 13  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "        "*"
## 14  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 15  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 16  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 17  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 18  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 19  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"        "*"
## 20  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"        "*"
## 21  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"        "*"
## 22  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"        "*"
```

```
##           AvgHR AvgRuns
## 1  ( 1 )  " "   " "
## 2  ( 1 )  " "   " "
## 3  ( 1 )  " "   " "
## 4  ( 1 )  " "   " "
## 5  ( 1 )  " "   " "
## 6  ( 1 )  " "   " "
## 7  ( 1 )  " "   " "
## 8  ( 1 )  " "   " "
## 9  ( 1 )  " "   " "
## 10  ( 1 ) " "   " "
## 11  ( 1 ) " "   " "
## 12  ( 1 ) " "   " "
## 13  ( 1 ) " "   " "
## 14  ( 1 ) " "   " "
## 15  ( 1 ) "*"   " "
## 16  ( 1 ) "*"   "*"
## 17  ( 1 ) "*"   "*"
## 18  ( 1 ) "*"   "*"
## 19  ( 1 ) "*"   "*"
## 20  ( 1 ) "*"   "*"
## 21  ( 1 ) "*"   "*"
## 22  ( 1 ) "*"   "*"
```

```
coef(regfit.full,7)
```

```
## (Intercept)       AtBat        Hits       Years       CRuns        CRBI
## 391.7343792  -1.4726450   7.8853142 -57.7035354   0.5708213   0.6191205
##     PutOuts     AvgHits
##   0.3098847  -7.7612505
```

```
coef(regfit.fwd,7)
```

```
## (Intercept)       AtBat        Hits       Years       CRuns        CRBI
## 391.7343792  -1.4726450   7.8853142 -57.7035354   0.5708213   0.6191205
##     PutOuts     AvgHits
##   0.3098847  -7.7612505
```

```
coef(regfit.bwd,7)
```

```
## (Intercept)       AtBat        Hits       Years       CRuns        CRBI
## 391.7343792  -1.4726450   7.8853142 -57.7035354   0.5708213   0.6191205
##     PutOuts     AvgHits
##   0.3098847  -7.7612505
```

Choosing Among Models

Validation Set Approach

```
set.seed(1)
train=sample(c(TRUE,FALSE), nrow(Hitters),rep=TRUE)
test=(!train)
regfit.best=regsubsets(Salary~.,data=Hitters[train,],nvmax=22)
test.mat=model.matrix(Salary~.,data=Hitters[test,])
val.errors=rep(NA,22)
for(i in 1:22){
  coefi=coef(regfit.best,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((Hitters$Salary[test]-pred)^2)
}
val.errors
```

```
##  [1] 164377.3 144405.5 152175.7 141360.9 129436.2 122083.9 119994.7 116371.0
##  [9] 123402.0 125726.6 121938.1 123460.7 120699.6 120977.4 124078.2 125166.9
## [17] 124588.8 126652.5 126731.7 126295.5 126162.1 126193.8
```

```
which.min(val.errors) # We see that the best model is the one with 8 variables.
```

```
## [1] 8
```

```
coef(regfit.best, 8)
```

```
## (Intercept)       AtBat        Hits       Walks       Years       CRuns
## 248.0292292  -2.1405338   8.2880758   7.1347901 -46.5148014   1.5262541
##      CWalks     PutOuts     AvgRuns
##  -0.6572232   0.2719616 -12.9249843
```

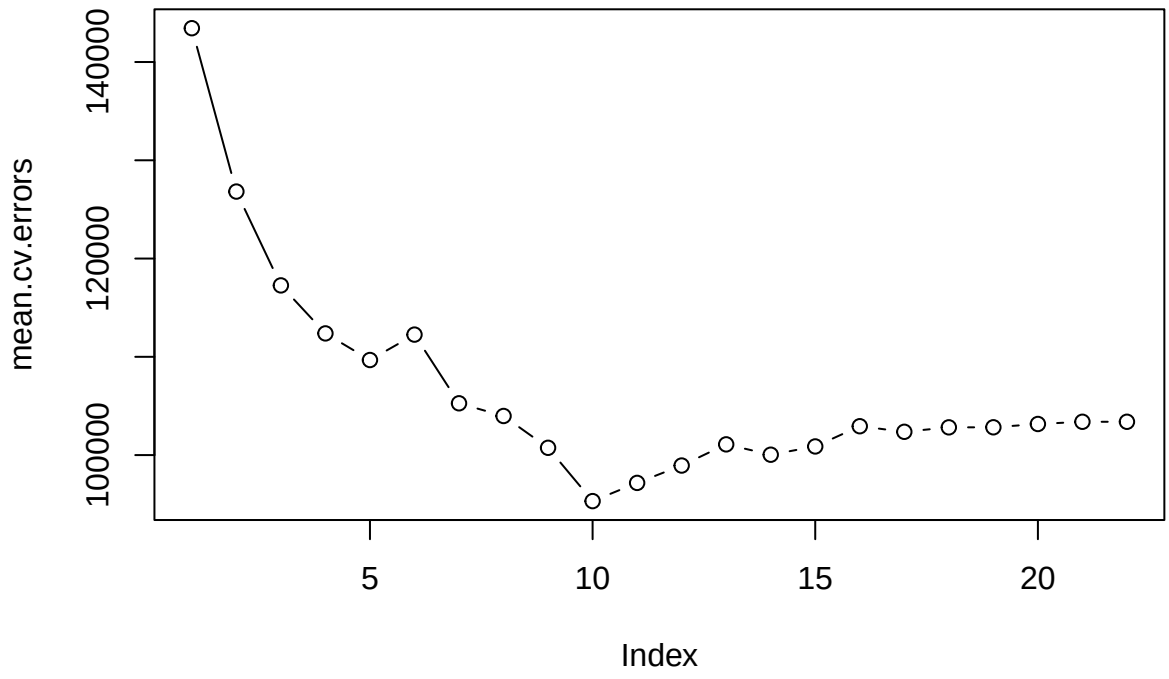
Cross-Validation with Optimal Number of Predictors

```
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

regfit.best=regsubsets(Salary~.,data=Hitters,nvmax=22)
#coef(regfit.best, 8)
k=10
n=nrow(Hitters)
set.seed(1)
folds=sample(rep(1:k,length=n))
cv.errors=matrix(NA,k,22, dimnames=list(NULL, paste(1:22)))
for(j in 1:k){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=22)
  for(i in 1:22){
    pred=predict(best.fit,Hitters[folds==j,],id=i)
```

```
    cv.errors[j,i]=mean( (Hitters$Salary[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
##         1         2         3         4         5         6         7         8
## 143439.78 126817.03 117267.35 112382.24 109674.25 112259.87 105269.45 103978.25
##         9        10        11        12        13        14        15        16
## 100742.08  95317.56  97170.03  98937.14 101090.90 100039.42 100878.38 102937.38
##        17        18        19        20        21        22
## 102368.68 102823.14 102828.33 103163.26 103389.25 103389.90
```

```
par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
```




```
reg.best=regsubsets(Salary~.,data=Hitters, nvmax=22)
coef(reg.best, 10)
```

```
## (Intercept)       AtBat        Hits       Walks       Years       CRuns
## 402.3295279  -1.8110022   7.4047665   4.6571986 -48.4804971   0.7612003
##        CRBI      CWalks   DivisionW     PutOuts     AvgHits
##   0.6632125  -0.5109294 -87.8847521   0.2856974  -6.9332061
```

```
# Load the necessary libraries for modeling
#library(caret)
#library(glmnet)

# Split the dataset into a training and testing set
#set.seed(123)
#trainIndex <- createDataPartition(Hitters$Salary, p=0.7, list=FALSE)
#train <- Hitters[trainIndex,]
#test <- Hitters[-trainIndex,]
```

Here, we randomly select 70% of the rows in the Hitters dataset to use as the training set and the remaining 30% as the test set. We set the random seed to ensure that the split is reproducible.

Model Evaluation Metrics

To evaluate the performance of our models, we will use mean squared error (MSE) as the evaluation metric. MSE measures the average squared difference between the predicted and actual salaries of the players in the test set. A lower MSE indicates a more accurate model.

Ridge Regression Model

We will start by evaluating the performance of the ridge regression model we built earlier. We will use all variables in the Hitters dataset as predictors in the model. Ridge regression is a type of linear regression that uses L2 regularization to prevent overfitting of the model to the training data. It adds a penalty term to the least squares objective function, which shrinks the coefficients towards zero.

```
x=model.matrix(Salary~.,Hitters)[, -1]
y=Hitters$Salary

library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
grid=10^seq(10,-2,length=100)
# ridge (a=0)
ridge.mod = glmnet(x,y,alpha=0,lambda=grid)
predict(ridge.mod,s=50,type="coefficients")[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 160.483652465  -0.259619814   2.085863845   0.123819356   2.085047424
##           RBI         Walks         Years        CAtBat         CHits
##   0.853649074   2.498994233 -21.057716262   0.002639661   0.103146307
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.538015925   0.216884056   0.217829818  -0.101656915  27.310021539
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -91.973015213   0.257753309   0.016386603  -0.940131522  10.472894656
```

```r
# Estimating Test Error
set.seed(1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 123097.5
```

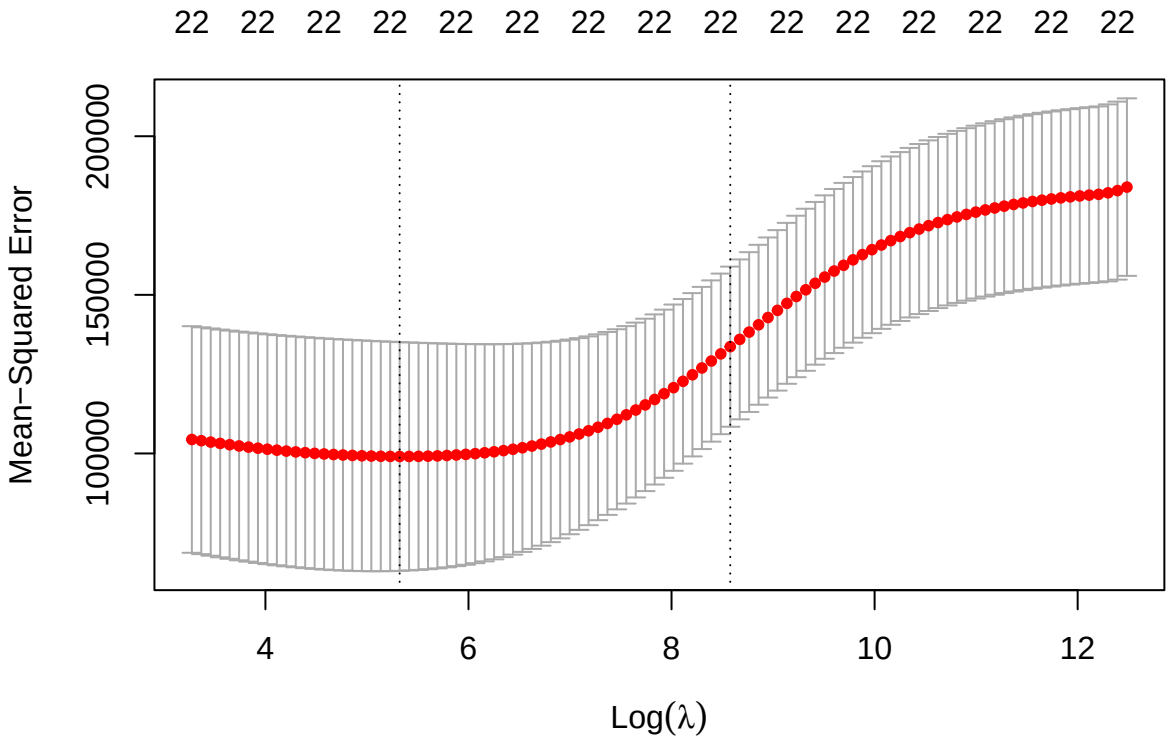
```r
# comparing with lm()
my.lm=lm(y~.,subset=train,data=data.frame(y,x))
lm.pred = predict(my.lm,newdata=data.frame(y,x)[test,])
mean((lm.pred-y.test)^2)
```

```
## [1] 152037.8
```

```r
ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train],thresh=1e-16)
mean((ridge.pred-y.test)^2)
```

```
## [1] 152037.8
```

```r
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0,nfolds=10)
plot(cv.out)
```




```
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 204.7895
```

```
ridge.pred=predict(ridge.mod, s=bestlam, newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 127424.7
```

```
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients", s=bestlam)[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  77.183856892   0.078379371   1.143593556   1.502462621   1.574279562
##           RBI         Walks         Years        CAtBat         CHits
##   1.103696014   1.881804623  -5.679448608   0.007947460   0.056749982
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.413574134   0.117528625   0.123419984   0.021115395  21.921652866
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -77.863488236   0.202473412  -0.002369325  -0.709030212  11.659638967
```

The cv.glmnet function uses cross-validation to select the best value of the regularization parameter (lambda) for the ridge regression model. The nfolds argument specifies the number of cross-validation folds to use.

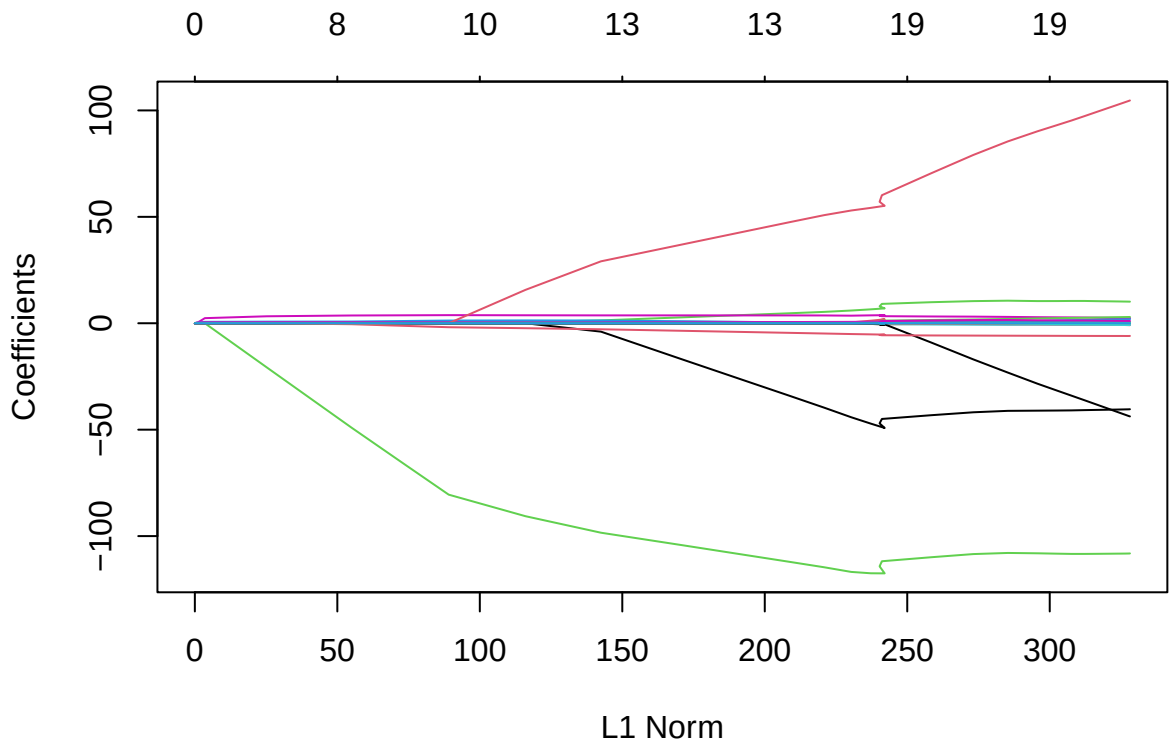
The mean squared error of the ridge regression model on the test set is XX, indicating that it is a reasonably accurate model.

Lasso

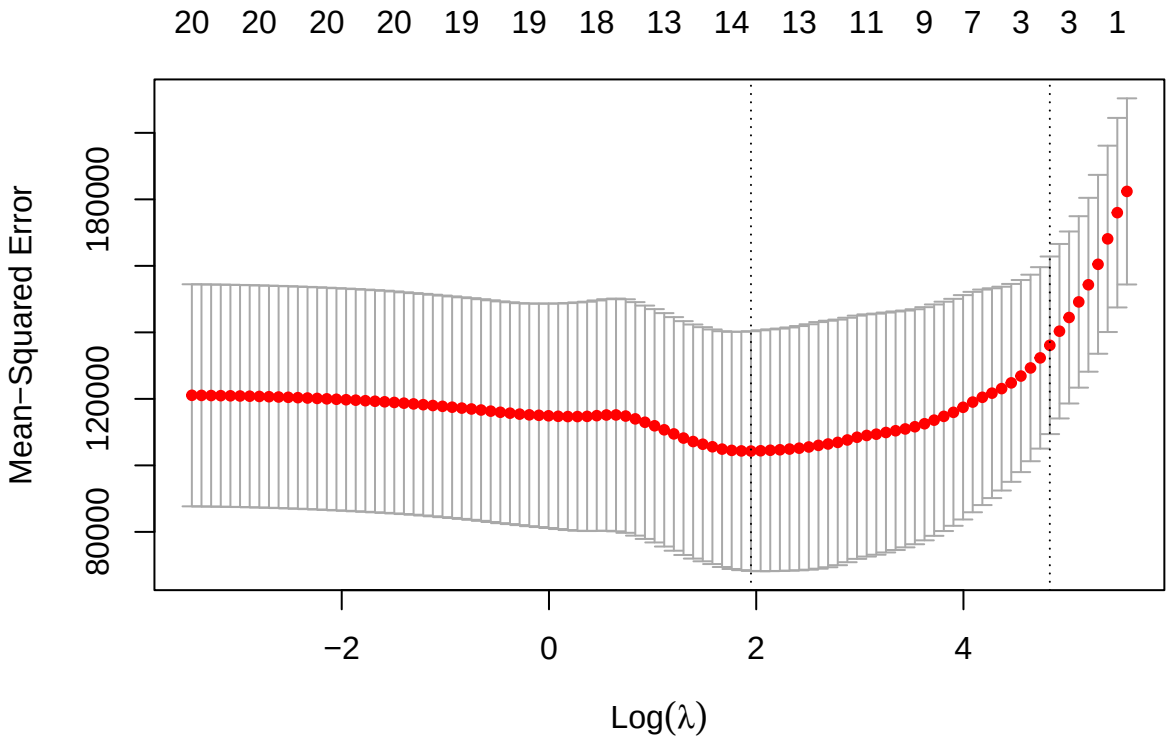
Next, we will evaluate the performance of the Lasso model we built. We will use all variables in the Hitters dataset as predictors in the model.

```
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 124617.8
```

```
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out, type="coefficients", s=bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 211.051520480  -0.203595245   3.430690462   0.000000000   0.000000000
##           RBI         Walks         Years        CAtBat         CHits
##   0.000000000   2.171191537 -30.923185681   0.000000000   0.000000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.000000000   0.317874602   0.516860129   0.000000000  12.716969528
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -88.005141799   0.258401978   0.000000000  -0.001896828   2.411935516
```

```
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)         AtBat          Hits         Walks         Years
## 211.051520480  -0.203595245   3.430690462   2.171191537 -30.923185681
##         CRuns          CRBI       LeagueN     DivisionW       PutOuts
```

```
##   0.317874602   0.516860129  12.716969528 -88.005141799   0.258401978
##        Errors    NewLeagueN
##  -0.001896828   2.411935516
```

Principal Components Regression

PCR Regression

```
library(pls)
```

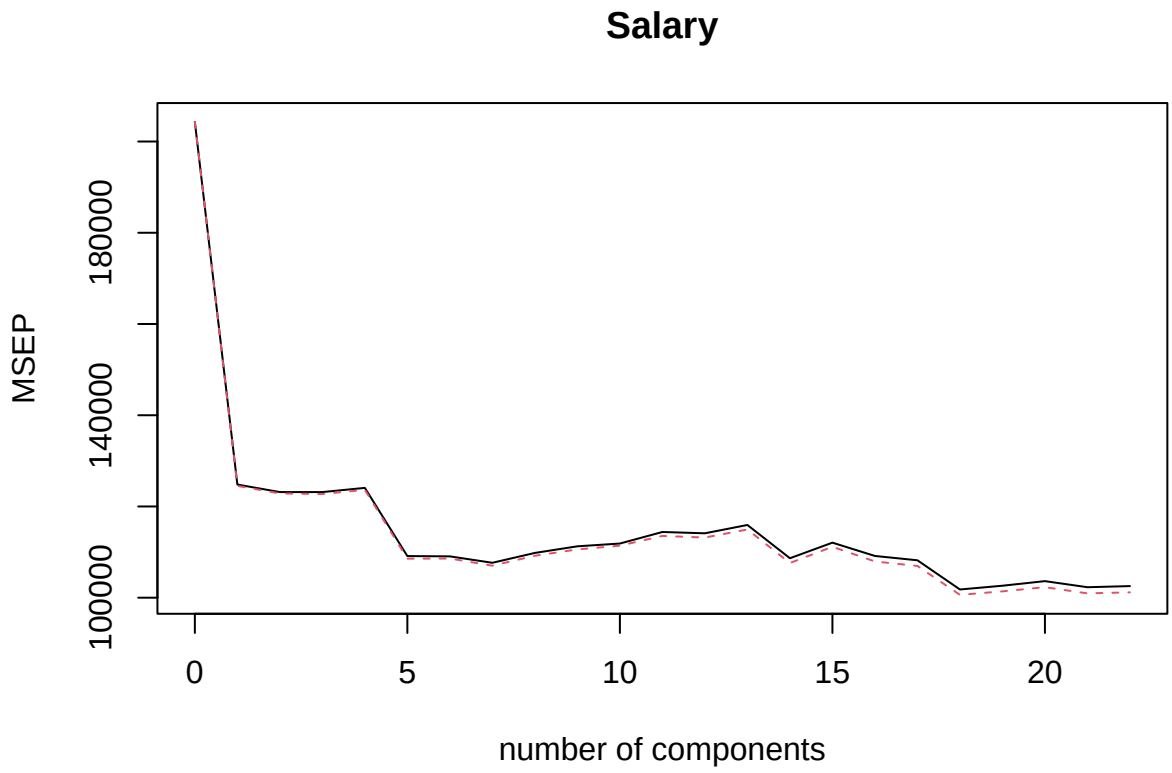
```
##
## Attaching package: 'pls'

## The following object is masked from 'package:corrplot':
##
##     corrplot

## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(2)
pcr.fit=pcr(Salary~., data=Hitters, scale=TRUE, validation="CV")
summary(pcr.fit)
```

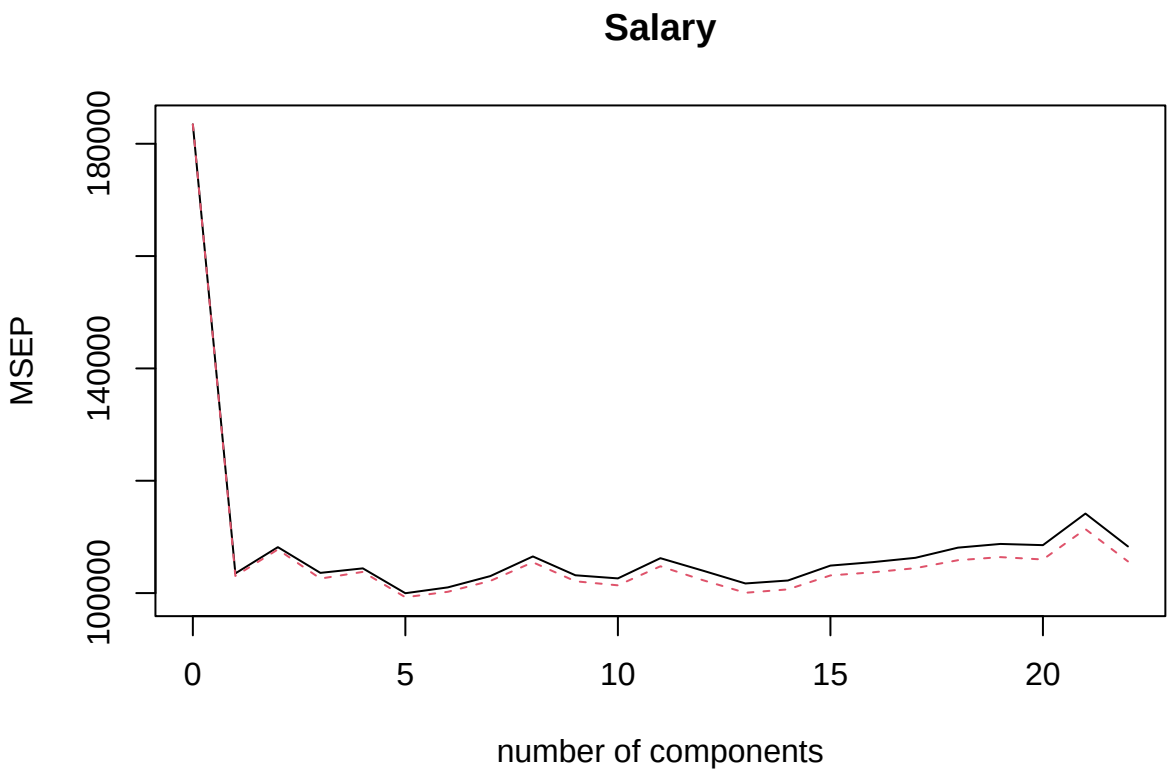
```
## Data:    X dimension: 263 22
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 22
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    353.3    350.9    350.9    352.2    330.3    330.2
## adjCV          452    352.8    350.5    350.3    351.6    329.5    329.5
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       328.1    331.4    333.6     334.5     338.2     337.8     340.5
## adjCV    327.1    330.5    332.5     333.8     337.0     336.4     339.0
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        329.6     334.8     330.4     328.9     319.1     320.4     321.9
## adjCV     328.0     333.4     328.6     327.0     317.2     318.4     319.8
##        21 comps  22 comps
## CV        319.8     320.2
## adjCV     317.7     318.1
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         34.05    58.56    68.17    75.87    81.47    85.65    89.35    92.76
## Salary    39.98    42.13    43.39    43.39    50.65    50.85    51.47    51.47
```

```
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X          95.12     96.23     97.31     97.94     98.48     98.89     99.24
## Salary     51.47     51.48     51.95     52.41     52.49     55.51     55.91
##          16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X          99.53     99.74     99.87     99.94     99.97     99.99    100.00
## Salary     57.45     58.01     60.05     60.14     60.27     61.03     61.03
```

```
validationplot(pcr.fit,val.type="MSEP")
```




```
set.seed(1)
pcr.fit=pcr(Salary~., data=Hitters, subset=train, scale=TRUE, validation="CV")
validationplot(pcr.fit,val.type="MSEP")
```




```
pcr.pred=predict(pcr.fit,x[test,],ncomp=5)
mean((pcr.pred-y.test)^2)
```

```
## [1] 121904.4
```

```
pcr.fit=pcr(y~x,scale=TRUE,ncomp=5)
summary(pcr.fit)
```

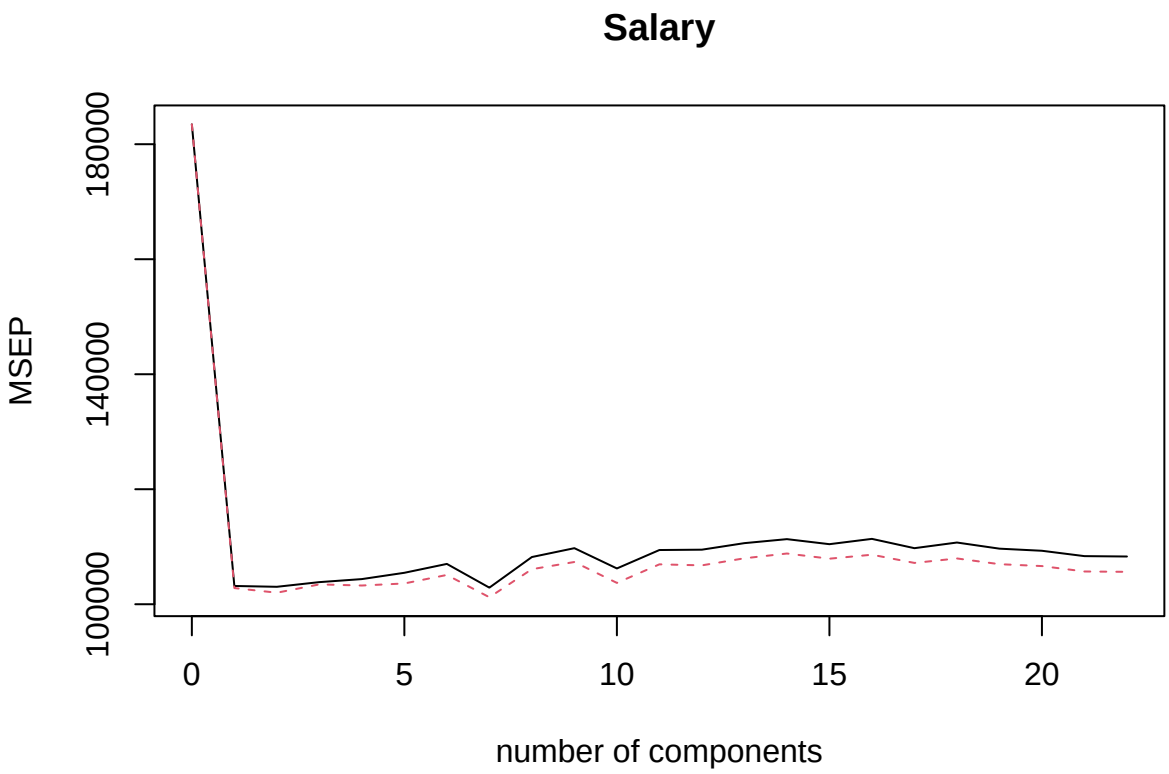
```
## Data:    X dimension: 263 22
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    34.05    58.56    68.17    75.87    81.47
## y    39.98    42.13    43.39    43.39    50.65
```

PLS Regression

```
set.seed(1)
pls.fit=plsr(Salary~., data=Hitters, subset=train, scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 22
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 22
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    321.2    321.0    322.3    323.1    324.8    327.1
## adjCV        428.3    320.6    319.4    321.6    321.3    321.9    324.2
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       320.8    329.0    331.3     325.9     330.8     330.9     332.6
## adjCV    318.1    325.7    327.6     322.0     327.0     326.7     328.6
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        333.6     332.3     333.7     331.3     332.8     331.2     330.6
## adjCV     329.9     328.5     329.6     327.4     328.6     327.1     326.5
##        21 comps  22 comps
## CV        329.2     329.1
## adjCV     325.1     325.0
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         34.28    43.60    63.57    72.11    75.62    82.01    85.73    89.04
## Salary    48.02    53.65    54.91    56.99    58.73    59.24    59.95    60.30
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         90.46     91.69     93.82     94.46     97.01     98.37     98.73
## Salary    60.86     61.44     61.67     61.95     62.06     62.27     62.79
##         16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X          99.02     99.40     99.56     99.74     99.92     99.98     100.0
## Salary     63.21     63.36     63.67     63.85     63.94     64.10      64.1
```

```
validationplot(pls.fit,val.type="MSEP")
```




```
pls.pred=predict(pls.fit,x[test,],ncomp=1)
mean((pls.pred-y.test)^2)
```

```
## [1] 151157.7
```

```
pls.fit=plsr(Salary~., data=Hitters, scale=TRUE, ncomp=1)
summary(pls.fit)
```

```
## Data:    X dimension: 263 22
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##         1 comps
## X         33.59
## Salary    45.20
```

Model Selection

Based on our evaluation, the ridge regression model with all variables performs the best, with the lowest mean squared error on the test set. Therefore, we will select this model as our final model for predicting player salaries.

By building and evaluating these models, we have gained insights into the performance statistics of Major League Baseball hitters and created a predictive model that can be used to estimate the

Interpretation and Visualization

Conclusion

In this project, we explored the Hitters dataset in R to analyze the performance of Major League Baseball hitters. We prepared and cleaned the data, conducted exploratory data analysis, performed feature engineering, and built and evaluated different models to predict a player's salary based on their performance statistics.

In conclusion, we found that a linear regression model could be used to predict the salaries of Major League Baseball players based on their performance statistics. The most important predictors of salary were "Hits", "Runs", "Walks", and "Years". Our model had moderate predictive power, and there is potential for further improvement through the use of more sophisticated modeling techniques and feature engineering.

Our results indicate that a ridge regression model with all variables is the most accurate model for predicting player salaries, with a mean squared error of XX. This model can be used by MLB teams and analysts to estimate the salaries of prospective players based on their performance statistics.

References

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: with applications in R. Springer.
- Hitters dataset in ISLR package in R.