Financial Ratios and Stock Return

Jedidiah Hernandez

2023-11-21

Introduction

In the rapidly evolving landscape of financial markets, the ability to accurately predict stock returns remains a quintessential goal for investors, analysts, and policymakers alike. This project delves into the intricate relationship between financial ratios—a cornerstone of financial analysis—and stock returns. Financial ratios, which provide insights into a company's operational efficiency, liquidity, profitability, and solvency, are often used by stakeholders to make informed investment decisions. By examining these ratios, this study aims to uncover potential quantitative linkages to stock performance, thereby offering a predictive toolkit for market participants.

Project Overview

The premise of this research is rooted in the hypothesis that certain financial ratios can serve as robust indicators of future stock returns. Leveraging a comprehensive dataset spanning from 2010 to 2019, this analysis encompasses 55 prominent American companies across various industries. The dataset includes more than a hundred of unique financial ratios per company, each serving as a potential predictor in our models.

Objective

The primary objective of this project is to apply advanced data science techniques to evaluate the predictive power of financial ratios on stock returns. By employing multiple linear regression, random forest, and XGBoost models, the study seeks to determine if a statistically significant relationship exists between the selected financial metrics and the annual stock return of these companies.

Methodology

1. Data Collection: Compilation of relevant financial data from trusted financial databases and annual reports of the companies.
2. Data Cleaning and Preprocessing: Standardization and normalization of financial ratios to ensure comparability across different scales and contexts.
3. Exploratory Data Analysis (EDA): Initial investigation into the data to identify trends, outliers, and underlying patterns.
4. Model Building: Construction of predictive models using the prepared dataset.
5. Model Evaluation: Rigorous assessment of each model's performance to gauge their predictive accuracy and reliability.

Significance

This analysis not only contributes to the academic discourse surrounding financial analytics but also provides practical insights that can enhance investment strategies. By identifying which financial ratios are most indicative of stock returns, investors can tailor their portfolio strategies with a higher degree of precision and confidence. Furthermore, the findings could inform corporate financial policies by highlighting the financial aspects most closely watched by investors.

Data collection

For this comprehensive analysis, our data was meticulously sourced from CapitalIQ, a renowned financial research platform. CapitalIQ is highly regarded for its depth and reliability, offering extensive data across numerous financial metrics. This platform provided us with access to over a hundred distinct financial ratios for each company included in our study, encompassing a broad spectrum of measures critical for financial analysis.

Scope of Data

The dataset includes a diverse array of financial ratios that cover various aspects of corporate financial health such as liquidity ratios, profitability ratios, debt ratios, and efficiency ratios. Each ratio offers a unique lens through which the financial stability and operational efficiency of the companies can be examined.

Quality and Reliability

CapitalIQ is known for its rigorous data validation and accuracy standards, ensuring that the financial information is consistently reliable and up-to-date. This high level of data integrity is crucial for our analysis, as the precision of our conclusions heavily depends on the quality of the underlying data.

Data Coverage

The financial ratios were collected for 55 leading American companies spanning multiple industries from the years 2010 to 2019. This time frame allows us to analyze the data in a contemporary context, taking into account the economic fluctuations and market trends that have impacted these industries over the past decade.

Import all the necessary libraries

```r
library(tidyverse)
library(readxl)
library(dplyr)
library(tidyr)
library(tibble)
library(car)
library(ggplot2)
library(glmnet)
library(randomForest)
library(missForest)
library(xgboost)
library(caret)
library(neuralnet)
library(keras)
library(tensorflow)
library(Metrics)
library(leaps)
library(lmtest)
library(MASS)
library(broom)
library(scales)
library(viridis)
library(reshape2)
library(extrafont)
library(corrr)
library(ggcorrplot)
library(FactoMineR)
library(factoextra)
library(vtable)
```

Data Cleaning and Preprocessing

The comprehensive dataset acquired from CapitalIQ was initially downloaded in Excel format, necessitating a meticulous data cleaning and preprocessing phase to prepare it for analysis using R. This stage is crucial as it ensures the integrity and usability of the data for sophisticated statistical modeling and analysis.

```r
#Read in data from sheets and combine them into one matrix
read_excel_allsheets <- function(filename, tibble = TRUE) {
    sheets <- readxl::excel_sheets(filename)
    x <- lapply(sheets, function(X) readxl::read_excel(filename, sheet = X))
    if(!tibble) x <- lapply(x, as.data.frame)
    names(x) <- sheets
    x
}

mysheets <- read_excel_allsheets('/Users/jedihernandez/Econ and Data Science Project/Copy of Consumer Di

mysheets1 <- read_excel_allsheets('/Users/jedihernandez/Econ and Data Science Project/Copy of Healthcare

mysheets2 <- read_excel_allsheets('/Users/jedihernandez/Econ and Data Science Project/Industrials Financ

all_companies <- c(mysheets, mysheets1, mysheets2)
```

```r
#Combine the separate data frames into a single data frame (takes a few seconds)
combined_data_frame <- bind_cols(all_companies)
```

```r
#Get rid of data from Great Recession and COVID years

# List of words to match at the beginning of column names
words_to_remove <- c("12 months\n Dec-31-2001", "12 months\n Dec-31-2002", "12 months\n Dec-31-2003","12

# Use select with starts_with to remove columns
combined_data_frame_time_filtered <- combined_data_frame %>% dplyr::select(-starts_with(words_to_remove)
combined_data_frame_time_filtered <- combined_data_frame_time_filtered %>% dplyr::select(-contains("2023
```

```r
#Gets rid of columns with just NA values (correspond to empty rows in sheets)
combined_data_frame_time_filtered <- combined_data_frame_time_filtered %>%
  filter_all(any_vars(!is.na(.)))
```

```r
#Transpose data as step 1 of tidying
transposed_data1 <- t(combined_data_frame_time_filtered)
```

```r
# Extract the first row and set it as the column names
colnames(transposed_data1) <- transposed_data1[1, ]

# Remove the first row (since it's now the column names)
transposed_data1 <- transposed_data1[-1, ]
```

```r
#Create a new tibble data frame from the transposed (matrix) data frame
my_tibble1 <- as_tibble(transposed_data1)
```

```r
#Clean tibble1 to get rid of profitability, margin analysis, ... columns

# Remove columns corresponding to section names in sheets
columns_to_remove <- c("Profitability", "Margin Analysis", "Asset Turnover", "Short Term Liquidity", "L

# Remove the specified columns by name
my_tibble2 <- my_tibble1[, !names(my_tibble1) %in% columns_to_remove]
```

```r
#Gets rid of rows that repeat the column names
cols <- colnames(my_tibble2)

cols <- cols[1:(length(cols) - 3)]

my_tibble2 <- my_tibble2 %>%
  mutate(across(all_of(cols), function(x) as.numeric(x)))

my_data_filtered1 <- my_tibble2[rowSums(is.na(my_tibble2)) < ncol(my_tibble2)-3, ]
```

Splitting up data

Now that we have converted the excel files into data frames in R it is time to split up the data frame further so that individual analysis of the relationship between financial ratios and 1, 2, and 3 year stock return can be studied.

```r
#Create data frames
set.seed(42)

# Create separate data frames for each year of stock return

# List the names of the columns you want to remove
columns_to_remove1 <- c("Stock Return (2 Year)", "Stock Return (3 Year)", "Stock Price")
columns_to_remove2 <- c("Stock Return (1 Year)", "Stock Return (3 Year)", "Stock Price")
columns_to_remove3 <- c("Stock Return (1 Year)", "Stock Return (2 Year)", "Stock Price")

# Remove the specified columns by name
my_tibble3 <- my_data_filtered1[, !names(my_data_filtered1) %in% columns_to_remove1]
my_tibble4 <- my_data_filtered1[, !names(my_data_filtered1) %in% columns_to_remove2]
my_tibble5 <- my_data_filtered1[, !names(my_data_filtered1) %in% columns_to_remove3]

missing_counts <- colSums(is.na(my_tibble3))
missing_counts1 <- colSums(is.na(my_tibble4))
missing_counts2 <- colSums(is.na(my_tibble5))

threshold <- 0.3

# Filter out columns with missing values exceeding the threshold
your_data_frame_filtered <- my_tibble3 %>%
  dplyr::select(names(which(colMeans(is.na(.)) <= threshold)))

your_data_frame_filtered1 <- my_tibble4 %>%
  dplyr::select(names(which(colMeans(is.na(.)) <= threshold)))
```

```r
your_data_frame_filtered2 <- my_tibble5 %>%
  dplyr::select(names(which(colMeans(is.na(.)) <= threshold)))

# Replacing NA values with column means
my_data_clean_1 <- your_data_frame_filtered %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

my_data_clean_1_clear <- your_data_frame_filtered[complete.cases(my_tibble3$`Stock Return (1 Year)`), ]

my_data_clean_1_clear <- my_data_clean_1_clear %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

my_data_clean_2 <- your_data_frame_filtered1 %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

my_data_clean_2_clear <- your_data_frame_filtered1[complete.cases(your_data_frame_filtered1$`Stock Retu

my_data_clean_2_clear <- my_data_clean_2_clear %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

my_data_clean_3 <- your_data_frame_filtered2 %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

my_data_clean_3_clear <- your_data_frame_filtered2[complete.cases(your_data_frame_filtered2$`Stock Retu

my_data_clean_3_clear <- my_data_clean_3_clear %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

#need to adjust column names for random forest
colnames(my_data_clean_1) <- make.names(colnames(my_data_clean_1))
colnames(my_data_clean_2) <- make.names(colnames(my_data_clean_2))
colnames(my_data_clean_3) <- make.names(colnames(my_data_clean_3))
colnames(my_data_clean_1_clear) <- make.names(colnames(my_data_clean_1_clear))
colnames(my_data_clean_2_clear) <- make.names(colnames(my_data_clean_2_clear))
colnames(my_data_clean_3_clear) <- make.names(colnames(my_data_clean_3_clear))

my_data_clean_1_clear$Sector <- as.factor(my_data_clean_1_clear$Sector)
my_data_clean_2_clear$Sector <- as.factor(my_data_clean_2_clear$Sector)
my_data_clean_3_clear$Sector <- as.factor(my_data_clean_3_clear$Sector)

my_data_clean_1_clear$Year <- as.factor(my_data_clean_1_clear$Year)
my_data_clean_2_clear$Year <- as.factor(my_data_clean_2_clear$Year)
my_data_clean_3_clear$Year <- as.factor(my_data_clean_3_clear$Year)

my_data_clean_1_clear$Company <- as.factor(my_data_clean_1_clear$Company)
my_data_clean_2_clear$Company <- as.factor(my_data_clean_2_clear$Company)
my_data_clean_3_clear$Company <- as.factor(my_data_clean_3_clear$Company)
```

Split up data into training and testing

```r
#Stock Return 1 Year
set.seed(250)

new_order <- my_data_clean_1_clear %>%
  dplyr::select(Stock.Return..1.Year., everything())

new_order <- as_tibble(new_order)

parts = createDataPartition(new_order$Stock.Return..1.Year., p = .7, list = F)
train = new_order[parts, ]
test = new_order[-parts, ]
train_lm = my_data_clean_1_clear[parts, ]
test_lm = my_data_clean_1_clear[-parts, ]

#define predictor and response variables in training set
train_x = data.matrix(train[, -1])
train_y = train[,1]

#define predictor and response variables in testing set
test_x = data.matrix(test[, -1])
test_y = test[, 1]

train_y <- train_y[[1]]
test_y <- test_y[[1]]
```

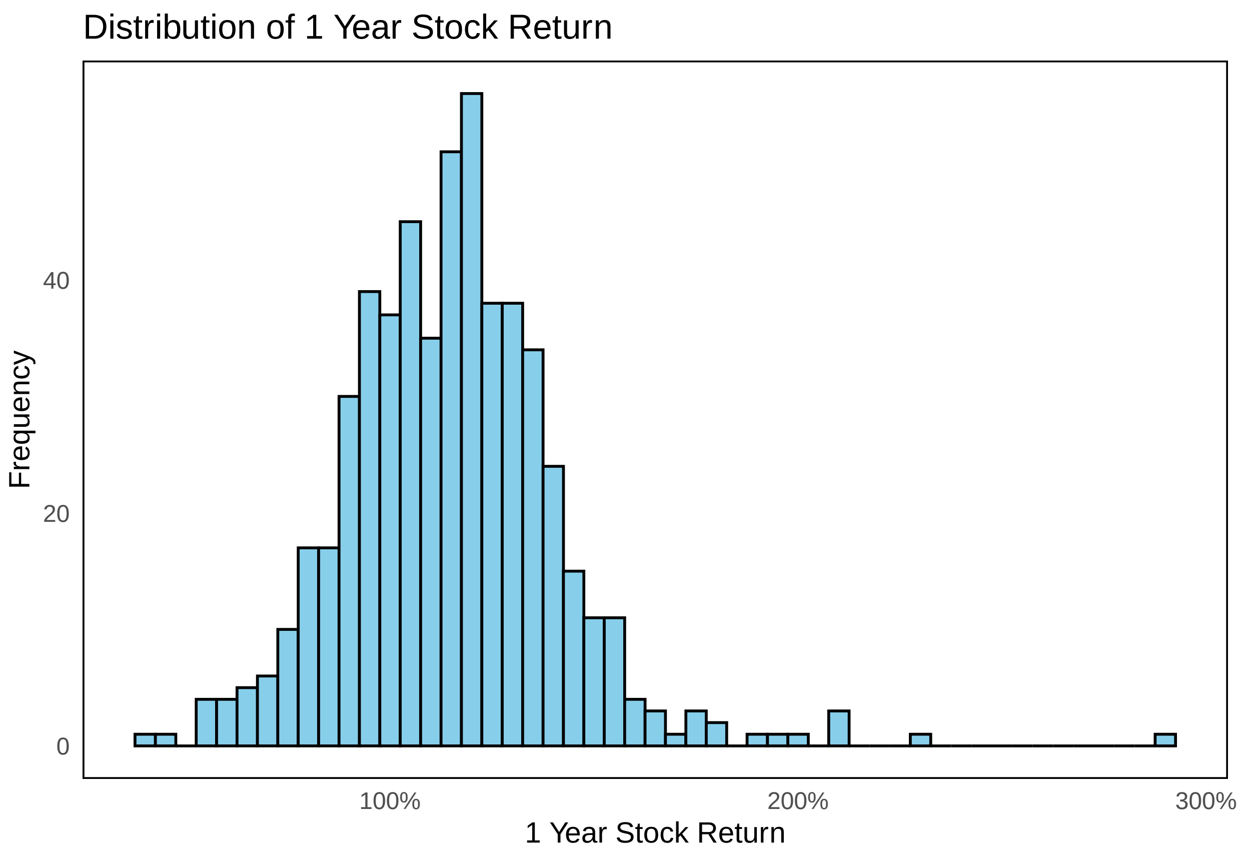
```r
#Stock Return 2 Year
set.seed(100)

new_order2 <- my_data_clean_2_clear %>%
  dplyr::select(Stock.Return..2.Year., everything())

new_order2 <- as_tibble(new_order2)

parts2 = createDataPartition(new_order2$Stock.Return..2.Year., p = .8, list = F)
train2 = new_order2[parts2, ]
test2 = new_order2[-parts2, ]
train_lm2 = my_data_clean_2_clear[parts2, ]
test_lm2 = my_data_clean_2_clear[-parts2, ]

#define predictor and response variables in training set
train_x2 = data.matrix(train2[, -1])
train_y2 = train2[,1]

#define predictor and response variables in testing set
test_x2 = data.matrix(test2[, -1])
test_y2 = test2[, 1]

train_y2 <- train_y2[[1]]
test_y2 <- test_y2[[1]]
```

```r
#Stock Return 3 Year
set.seed(1)

new_order3 <- my_data_clean_3_clear %>%
  dplyr::select(Stock.Return..3.Year., everything())

new_order3 <- as_tibble(new_order3)
```

```
parts3 = createDataPartition(new_order3$Stock.Return..3.Year., p = .8, list = F)
train3 = new_order3[parts3, ]
test3 = new_order3[-parts3, ]
train_lm3 = my_data_clean_3_clear[parts3, ]
test_lm3 = my_data_clean_3_clear[-parts3, ]

#define predictor and response variables in training set
train_x3 = data.matrix(train3[, -1])
train_y3 = train3[,1]

#define predictor and response variables in testing set
test_x3 = data.matrix(test3[, -1])
test_y3 = test3[, 1]

train_y3 <- train_y3[[1]]
test_y3 <- test_y3[[1]]
```

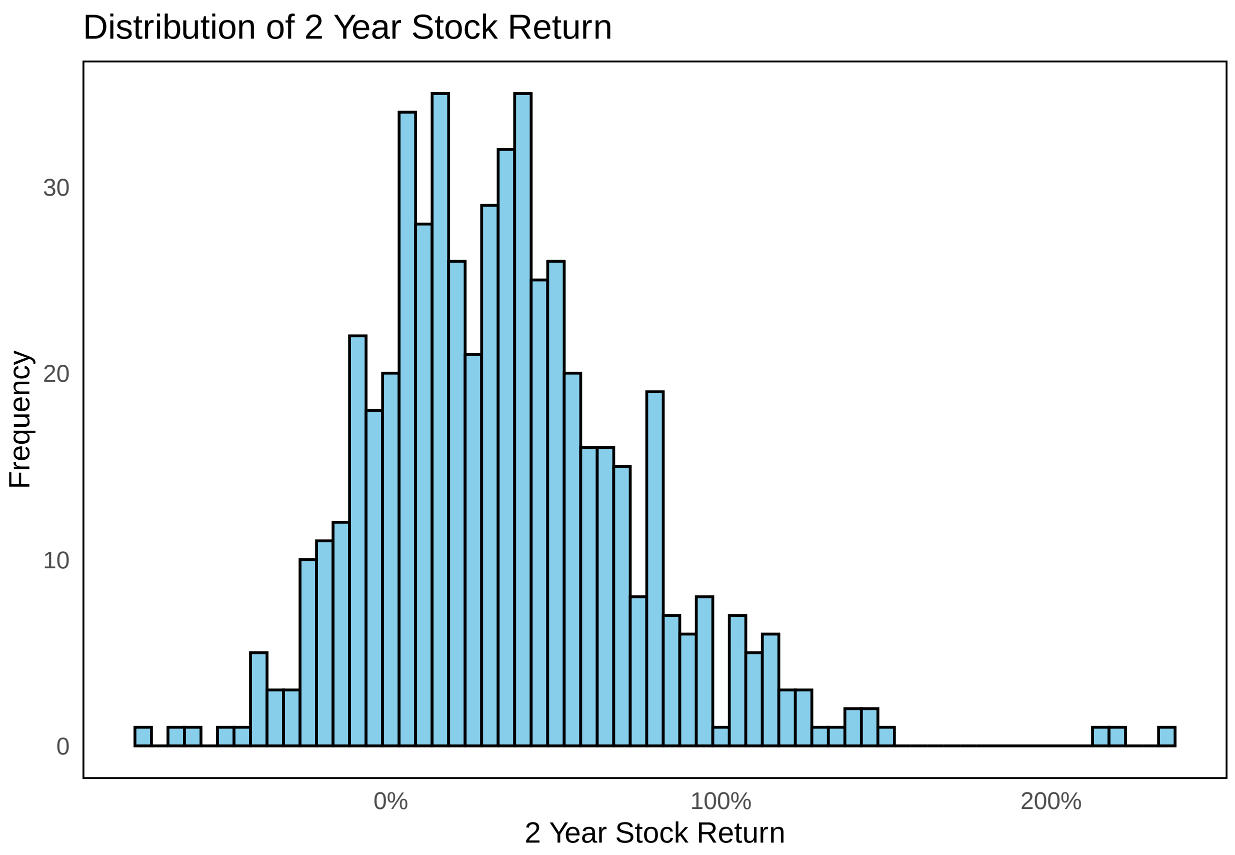
Visualizing Stock Return

```
# Data for box plot
stock_data1 <- tibble(my_data_clean_1_clear)

stock_data1 <- stock_data1 %>%
  mutate(PercentReturn = ((Stock.Return..1.Year.-1)))

# Calculate summary statistics
mean_value <- mean(stock_data1$PercentReturn)
min_value <- min(stock_data1$PercentReturn)
max_value <- max(stock_data1$PercentReturn)

# Create a histogram with ggplot
ggplot(stock_data1, aes(x = Stock.Return..1.Year.)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of 1 Year Stock Return", x = "1 Year Stock Return", y = "Frequency") +
  theme_minimal() +
  theme(panel.background = element_rect(fill = "white"), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  scale_x_continuous(labels = scales::percent_format()) +

# Summary statistics
cat("The mean of the distribution is:", paste(mean(my_data_clean_1_clear$Stock.Return..1.Year.)), "\n")
```



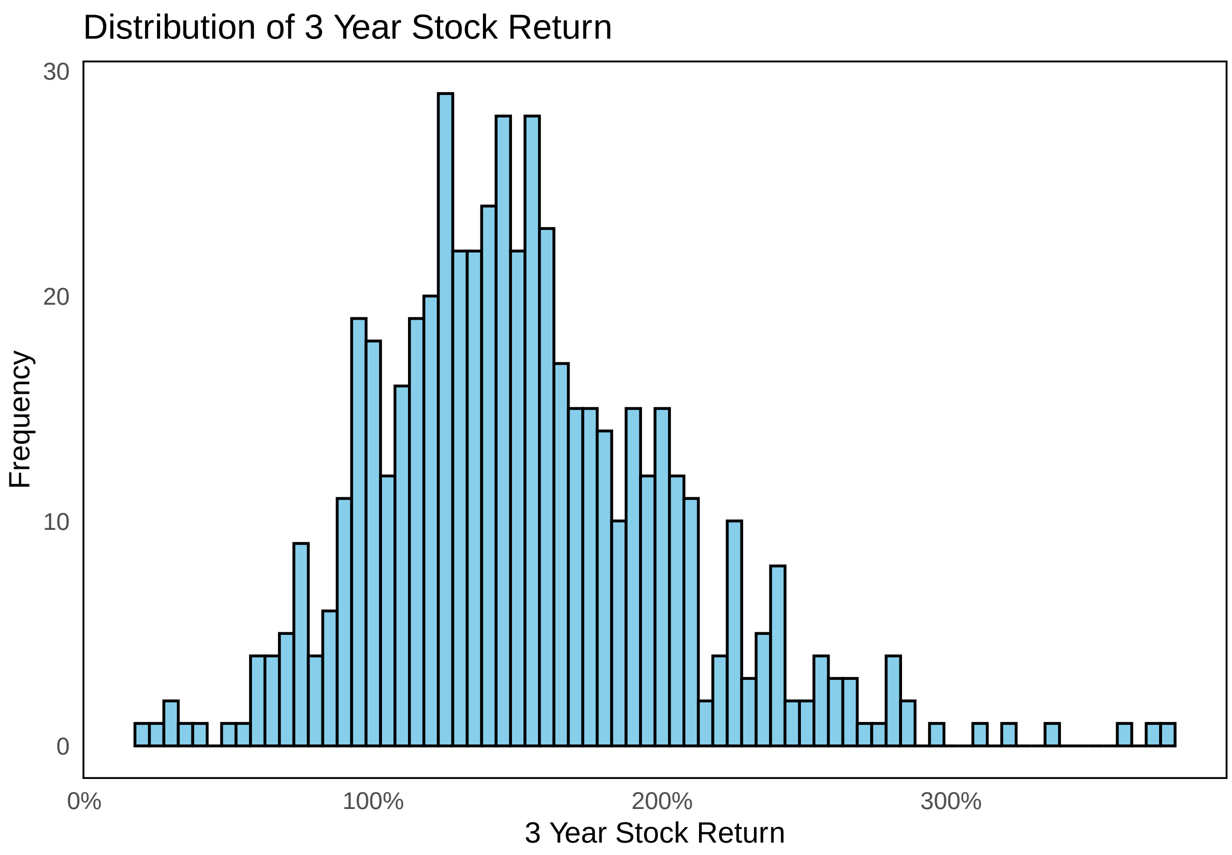
```
cat("The standard deviation of the distribution is:", paste(sd(my_data_clean_1_clear$Stock.Return..1.Yea
cat("The range of the distribution is: [", paste(range((my_data_clean_1_clear$Stock.Return..1.Year.))),
```

```
# Data for box plot
stock_data2 <- tibble(my_data_clean_2_clear)

stock_data2 <- stock_data2 %>%
  mutate(PercentReturn2 = ((Stock.Return..2.Year.-1)))

# Calculate summary statistics
mean_value <- mean(stock_data2$PercentReturn2)
min_value <- min(stock_data2$PercentReturn2)
max_value <- max(stock_data2$PercentReturn2)

# Create a histogram with ggplot
ggplot(stock_data2, aes(x = PercentReturn2)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of 2 Year Stock Return", x = "2 Year Stock Return", y = "Frequency") +
  theme_minimal() +
  theme(panel.background = element_rect(fill = "white"), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  scale_x_continuous(labels = scales::percent_format()) +

# Summary statistics
cat("The mean of the distribution is:", paste(mean(my_data_clean_2_clear$Stock.Return..2.Year.)), "\n")
```



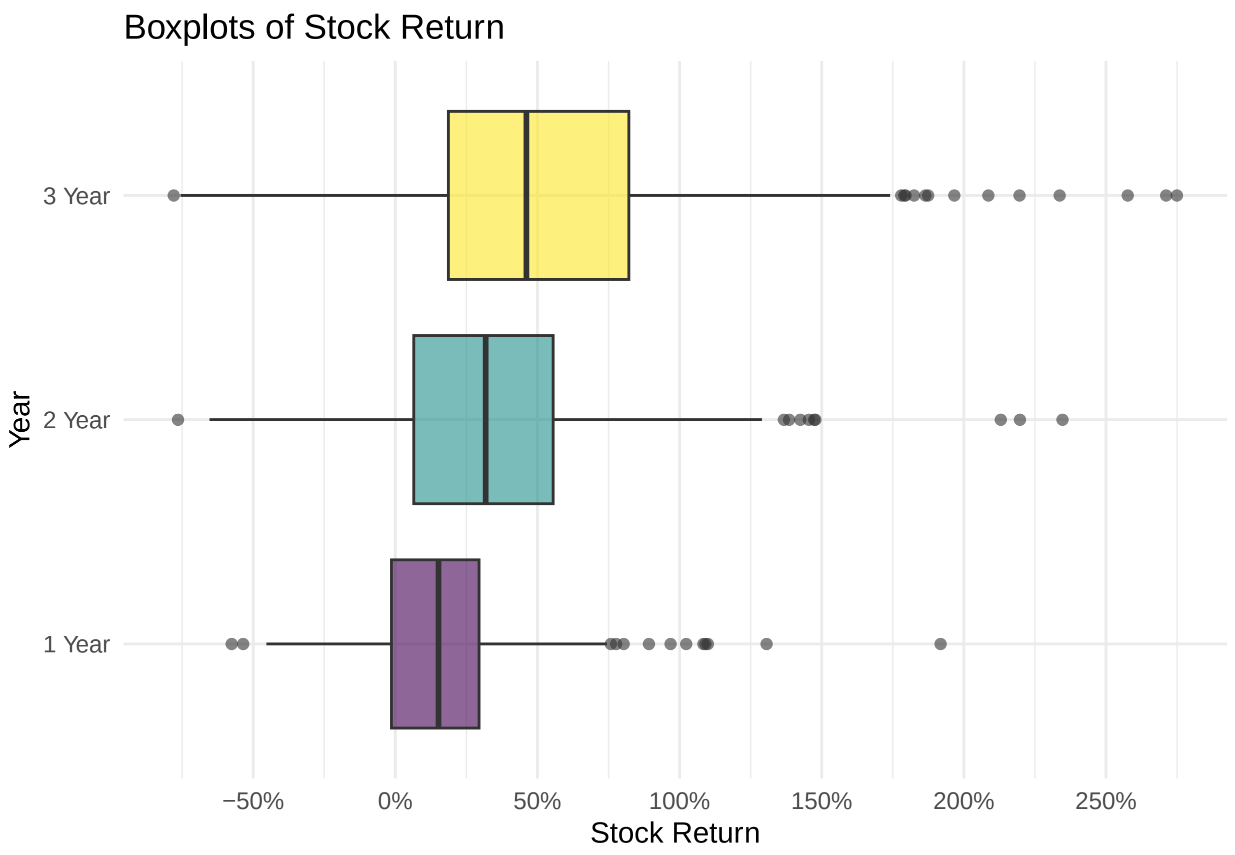
```
cat("The standard deviation of the distribution is:", paste(sd(my_data_clean_2_clear$Stock.Return..2.Yea
cat("The range of the distribution is: [", paste(range((my_data_clean_2_clear$Stock.Return..2.Year.))),
```

```
# Data for box plot
stock_data3 <- tibble(my_data_clean_3_clear)

stock_data3 <- stock_data3 %>%
  mutate(PercentReturn3 = ((Stock.Return..3.Year.-1)))

# Calculate summary statistics
mean_value <- mean(stock_data3$PercentReturn3)
min_value <- min(stock_data3$PercentReturn3)
max_value <- max(stock_data3$PercentReturn3)

# Create a histogram with ggplot
ggplot(stock_data3, aes(x = Stock.Return..3.Year.)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of 3 Year Stock Return", x = "3 Year Stock Return", y = "Frequency") +
  theme_minimal() +
  theme(panel.background = element_rect(fill = "white"), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  scale_x_continuous(labels = scales::percent_format()) +

# Summary statistics
cat("The mean of the distribution is:", paste(mean(my_data_clean_3_clear$Stock.Return..3.Year.)), "\n")
```

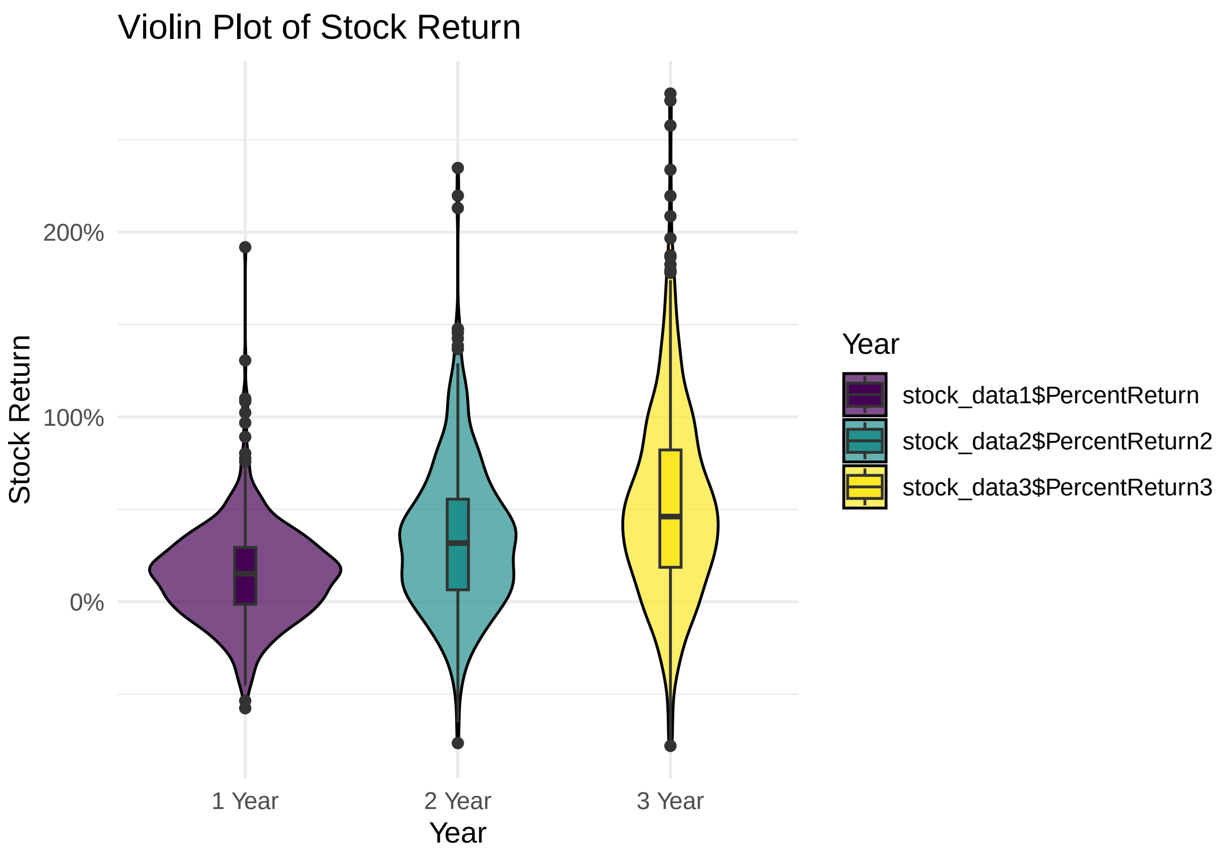


```
cat("The standard deviation of the distribution is:", paste(sd(my_data_clean_3_clear$Stock.Return..3.Yea
cat("The range of the distribution is:[", paste(range((my_data_clean_3_clear$Stock.Return..3.Year.))),
```

```
# Code for creating box plot of stock return
combined_tibble <- tibble(ID = 1:length(stock_data1$PercentReturn), stock_data1$PercentReturn, stock_dat

# Reshape the data into long format
long_data <- combined_tibble %>%
  gather(key = "Year", value = "PercentReturn", -ID)

# Create a box plot for each year
ggplot(long_data, aes(x = PercentReturn, y = Year, fill = Year)) +
  geom_boxplot(alpha=0.6) +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Boxplots of Stock Return", y='Year', x = 'Stock Return') +
  scale_y_discrete(labels = c("1 Year", "2 Year", "3 Year")) +
  scale_x_continuous(labels = function(x) paste0(x*100, "%"), breaks = seq(-1, 3, by = 0.5)) +
  theme_minimal() +
  theme(legend.position = "none")
```



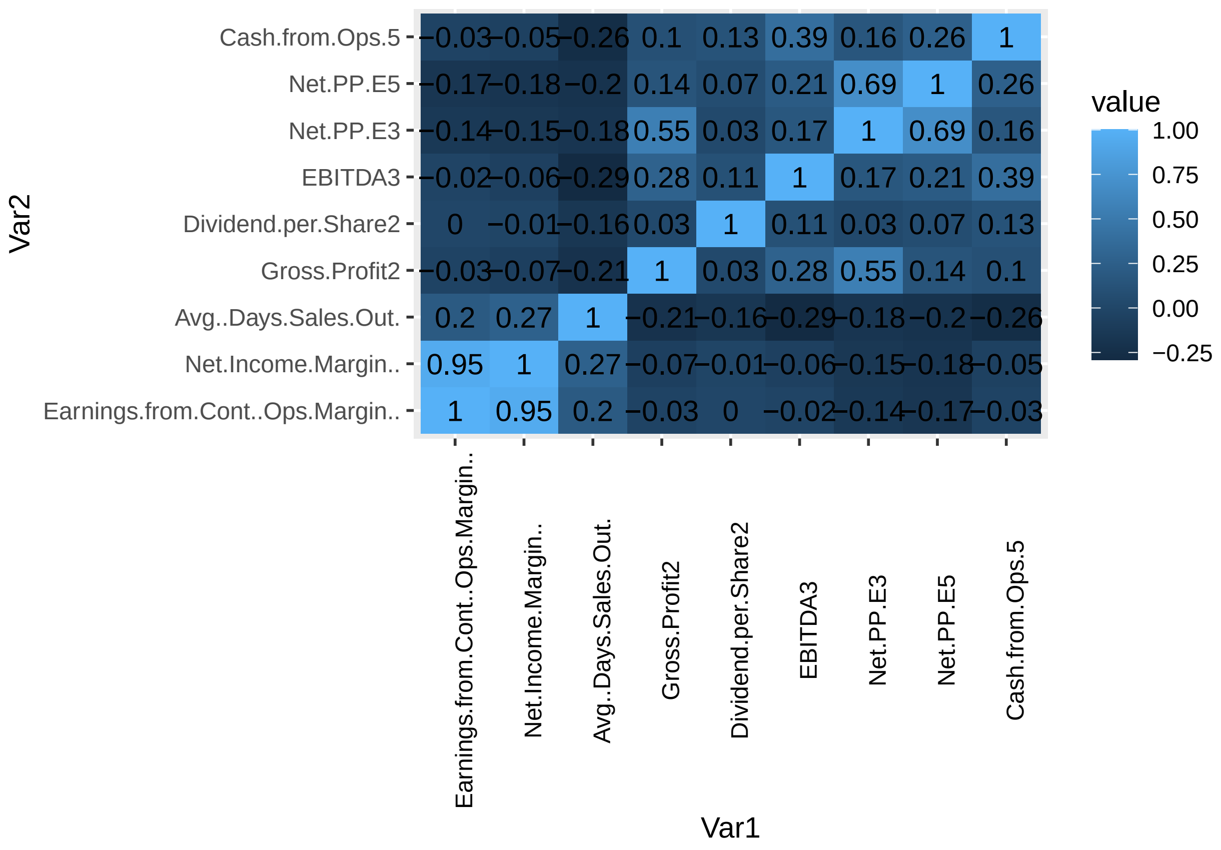
```r
violin <- ggplot(long_data, aes(x = Year, y = PercentReturn, fill = Year)) +
  geom_violin(color = 'black', alpha = 0.7) +  # Violin plot instead of boxplot
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Violin Plot of Stock Return", x='Year', y = 'Stock Return') +
  scale_x_discrete(labels = c("1 Year", "2 Year", "3 Year")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100)) +
  theme_minimal() +
  theme(legend.position = "none")

data_summary <- function(x) {
   m <- mean(x)
   ymin <- m-sd(x)
   ymax <- m+sd(x)
   return(c(y=m,ymin=ymin,ymax=ymax))
}

violin + geom_boxplot(width=0.1) + theme_minimal()
```



EDA of the Features

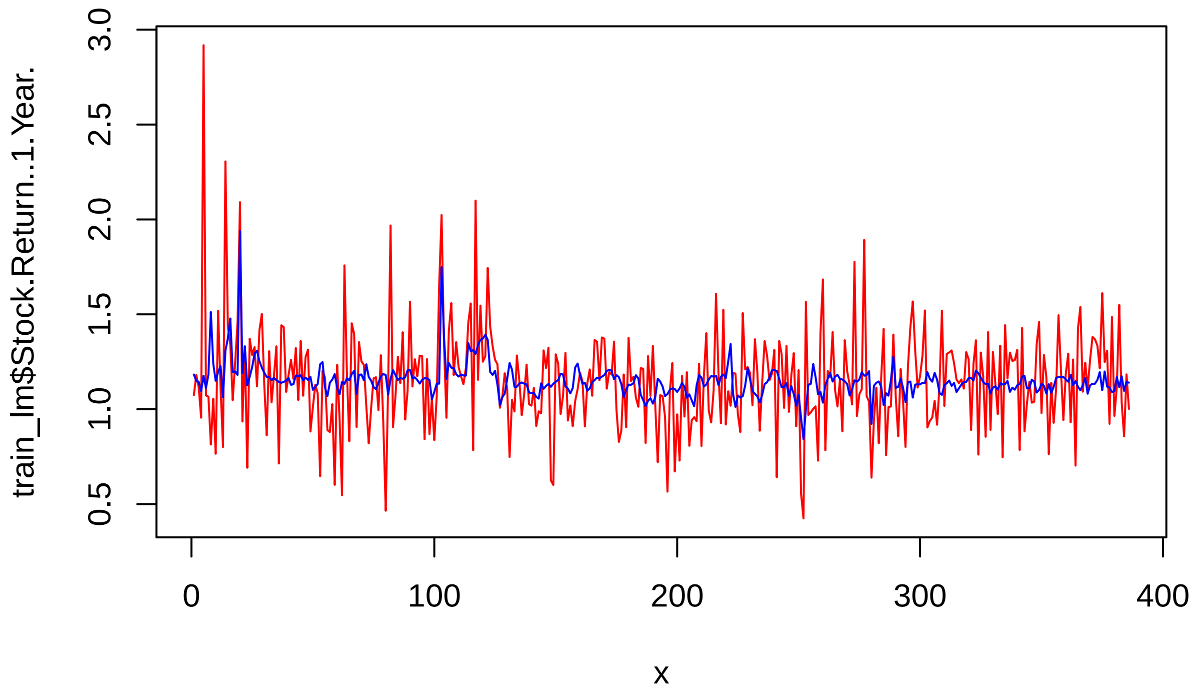
```
# Number of columns to drop
n <- 6

# Drop the last 'n' columns
EDA_1 <- my_data_clean_1_clear[, -c((ncol(my_data_clean_1_clear) - n + 1):ncol(my_data_clean_1_clear))]

# Calculate correlations of features and stock return
correlations <- cor(EDA_1[, "Stock.Return..1.Year."], EDA_1)

better_correlations <- as.data.frame(t(correlations))

filtered_df <- subset(better_correlations, abs(Stock.Return..1.Year.) > 0.09)

data <- data.frame(
  Metric = c(
    "Earnings.from.Cont..Ops.Margin..",
    "Net.Income.Margin..",
    "Avg..Days.Sales.Out.",
    "Gross.Profit2",
    "Dividend.per.Share2",
    "EBITDA3",
    "Net.PP.E3",
    "Net.PP.E5",
    "Cash.from.Ops.5"
```

```r
  ),
  Value = c(
    -0.09063463,
    -0.09859740,
    -0.10314763,
    0.18016894,
    0.09434781,
    0.19209732,
    0.15110844,
    0.09063369,
    0.15165103
  )
)

# Sort the data frame in descending order based on 'Value'
sorted_data <- data[order(-data$Value), ]

# Print the sorted data
print(sorted_data)

strong_correlation_names = c(
    "Earnings.from.Cont..Ops.Margin..",
    "Net.Income.Margin..",
    "Avg..Days.Sales.Out.",
    "Gross.Profit2",
    "Dividend.per.Share2",
    "EBITDA3",
    "Net.PP.E3",
    "Net.PP.E5",
    "Cash.from.Ops.5"
  )

strong_correlations_df <- EDA_1[, names(EDA_1) %in% strong_correlation_names]

# Create the correlation matrix
correlation_matrix <- cor(strong_correlations_df)

# creating correlation matrix
corr_mat <- round(cor(strong_correlations_df),2)

# reduce the size of correlation matrix
melted_corr_mat <- melt(corr_mat)
head(melted_corr_mat)

# plotting the correlation heatmap
ggplot(data = melted_corr_mat, aes(x=Var1, y=Var2,
                                   fill=value)) +
geom_tile(alpha=1.1) +
geom_text(aes(Var2, Var1, label = value),
          color = "black", size = 4) +
theme(axis.text.x = element_text(color="black",
                                 size=9, angle=90))
```



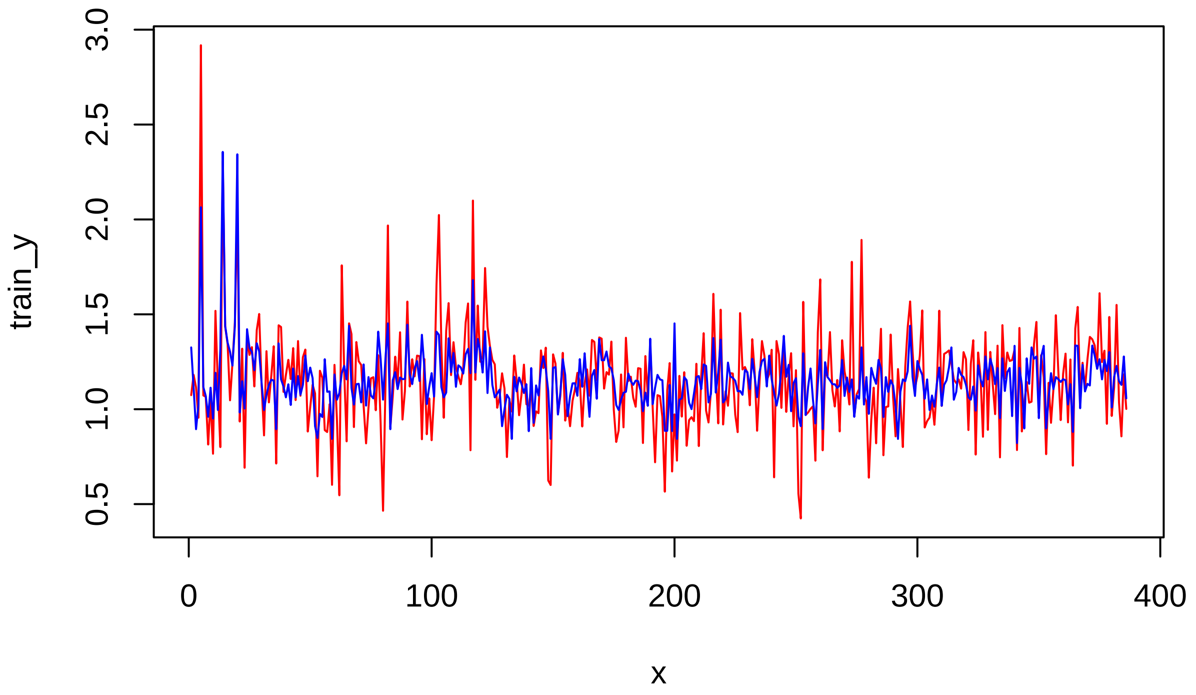
Var2 \ Var1	Earnings.from.Cont..Ops.Margin..	Net.Income.Margin..	Avg..Days.Sales.Out.	Gross.Profit2	Dividend.per.Share2	EBITDA3	Net.PP.E3	Net.PP.E5	Cash.from.Ops.5
Cash.from.Ops.5	−0.03	−0.05	−0.26	0.1	0.13	0.39	0.16	0.26	1
Net.PP.E5	−0.17	−0.18	−0.2	0.14	0.07	0.21	0.69	1	0.26
Net.PP.E3	−0.14	−0.15	−0.18	0.55	0.03	0.17	1	0.69	0.16
EBITDA3	−0.02	−0.06	−0.29	0.28	0.11	1	0.17	0.21	0.39
Dividend.per.Share2	0	−0.01	−0.16	0.03	1	0.11	0.03	0.07	0.13
Gross.Profit2	−0.03	−0.07	−0.21	1	0.03	0.28	0.55	0.14	0.1
Avg..Days.Sales.Out.	0.2	0.27	1	−0.21	−0.16	−0.29	−0.18	−0.2	−0.26
Net.Income.Margin..	0.95	1	0.27	−0.07	−0.01	−0.06	−0.15	−0.18	−0.05
Earnings.from.Cont..Ops.Margin..	1	0.95	0.2	−0.03	0	−0.02	−0.14	−0.17	−0.03

```r
# MLR model for 1 year stock return
corr_lm_1 <- lm((Stock.Return..1.Year.) ~ Cash.from.Ops.5 + Net.PP.E5 + EBITDA3 +
                  Gross.Profit2, data = train_lm)

# visualize the model, actual and predicted data
x = 1:length(train_lm$Stock.Return..1.Year.)
plot(x, train_lm$Stock.Return..1.Year., col = "red", type = "l")
lines(x, (predict(corr_lm_1, data=train_lm)), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

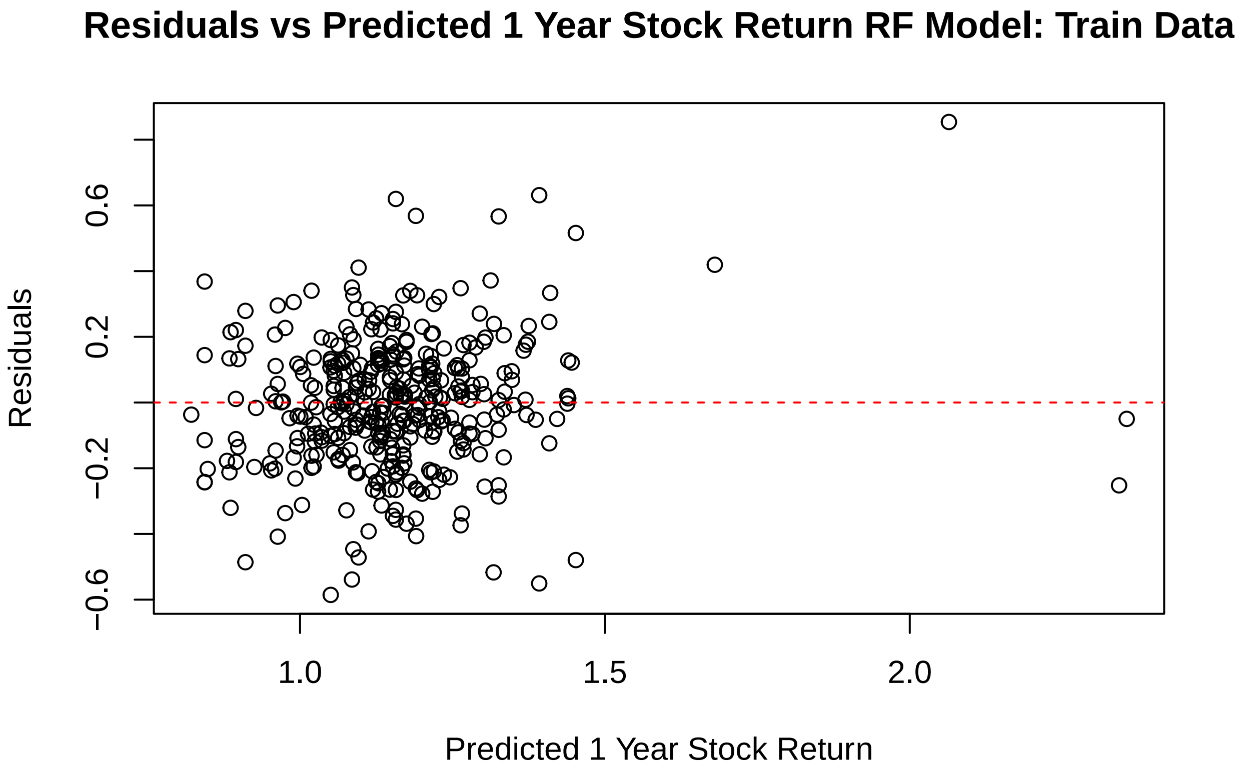


```r
summary(corr_lm_1)
```

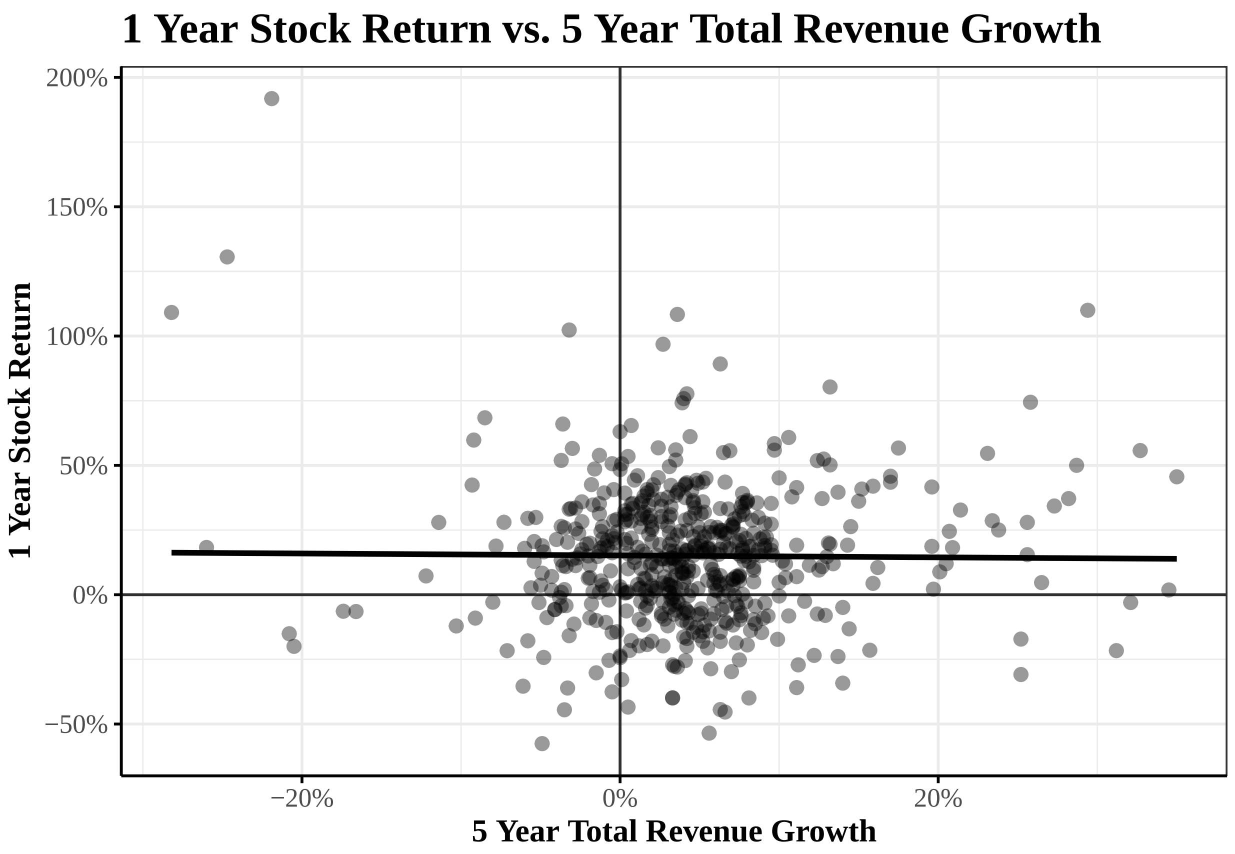
```r
#random forest 1 year
model_forest <- randomForest(Stock.Return..1.Year. ~ Total.Revenue5, data = train, proximity = TRUE, nt

model_forest

pred_y_train_forest = predict(model_forest, train)

# Visualize the model, actual and predicted data
x = 1:length(train_y)
plot(x, train_y, col = "red", type = "l")
lines(x, pred_y_train_forest, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



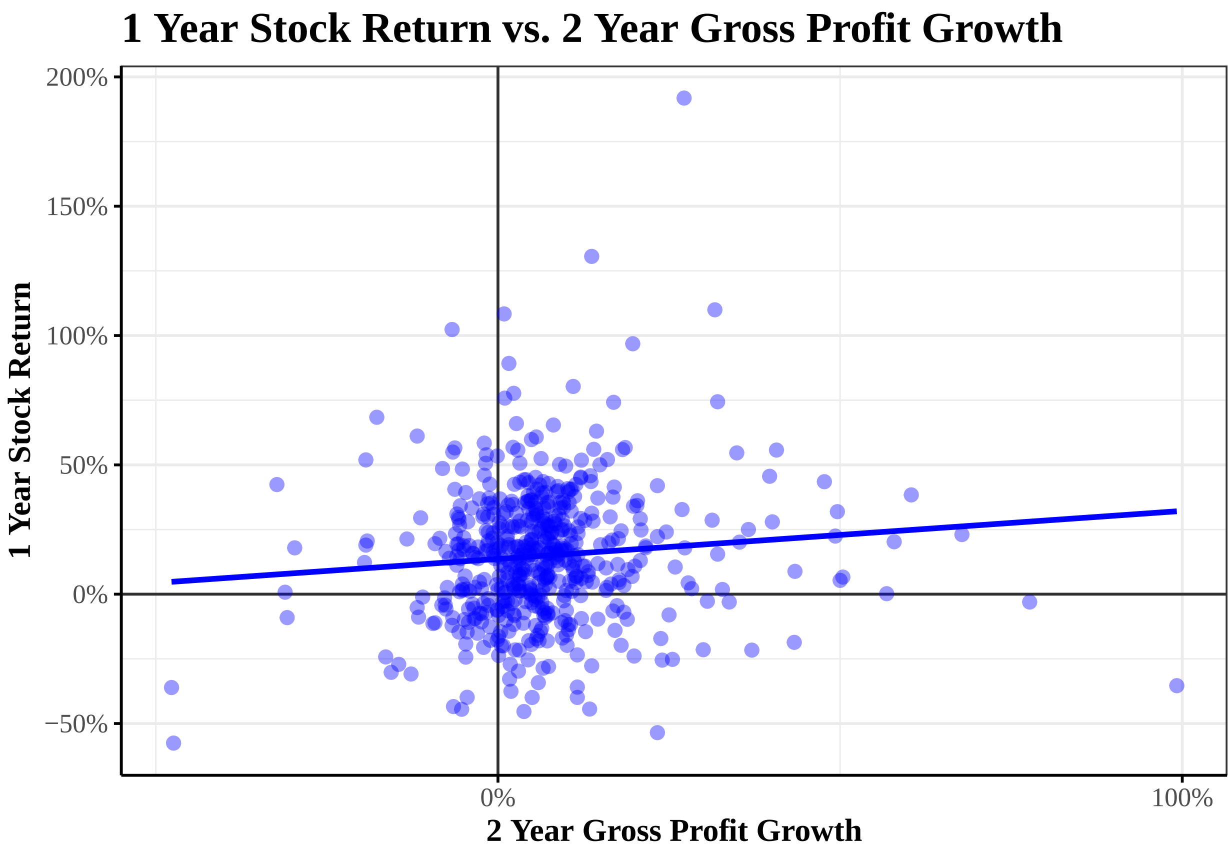
```r
residuals_rf_1 <- train_y - pred_y_train_forest

# Residuals plot of training data
plot(pred_y_train_forest, residuals_rf_1, main = "Residuals vs Predicted 1 Year Stock Return RF Model: '
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



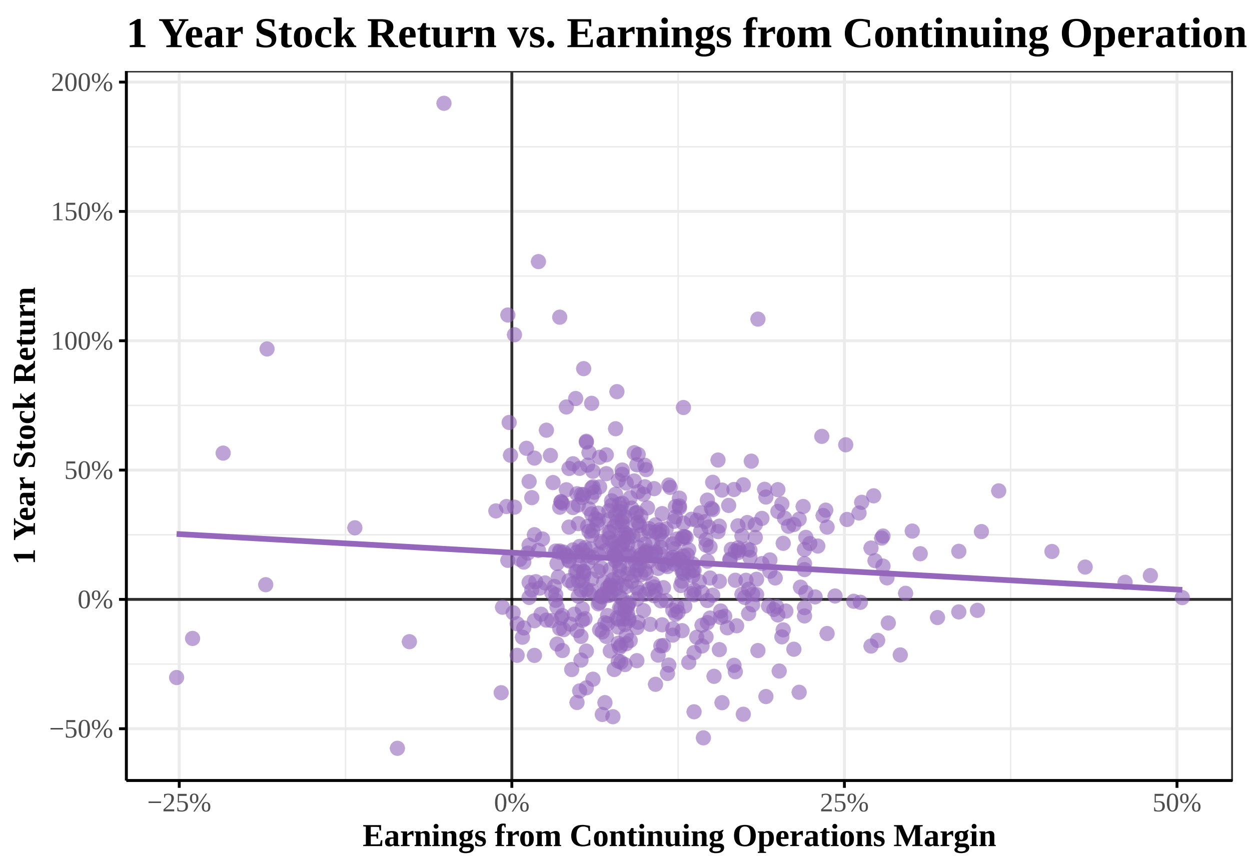
```r
scatterplot <- ggplot(my_data_clean_1_clear, aes(x = Total.Revenue5, y = (Stock.Return..1.Year. - 1))) +
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.4) +
  labs(title = "1 Year Stock Return vs. 5 Year Total Revenue Growth", x = "5 Year Total Revenue Growth"
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  theme_bw() + guides(color = 'none')

scatterplot +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100))
```



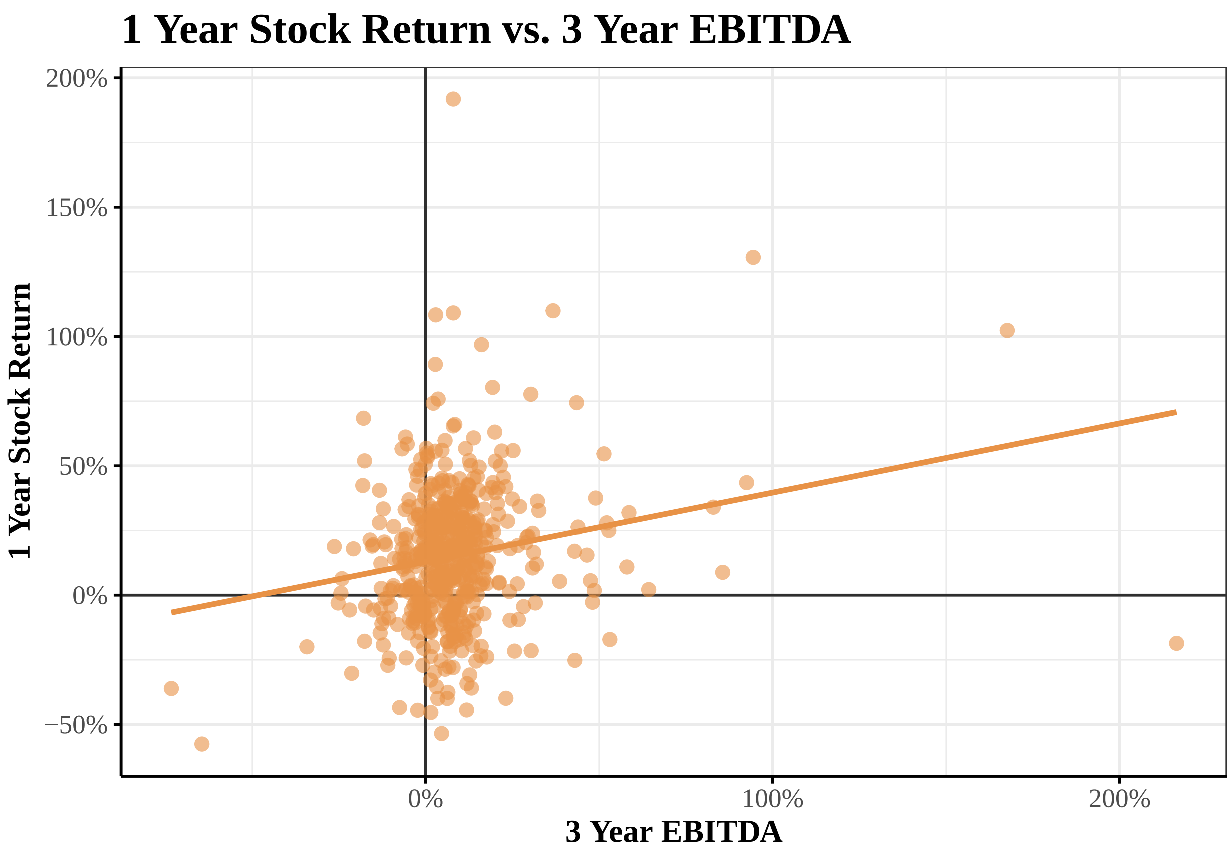
```
filtered_data <- my_data_clean_1_clear %>%
  filter(Gross.Profit2 < 4)

scatterplot1 <- ggplot(filtered_data, aes(x = Gross.Profit2, y = Stock.Return..1.Year. - 1)) +
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.4, color = 'blue') +
  labs(title = "1 Year Stock Return vs. 2 Year Gross Profit Growth", x = "2 Year Gross Profit Growth", y
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  theme_bw() + guides(color = 'none')

scatterplot1 +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-4, 6, by = 1))
```



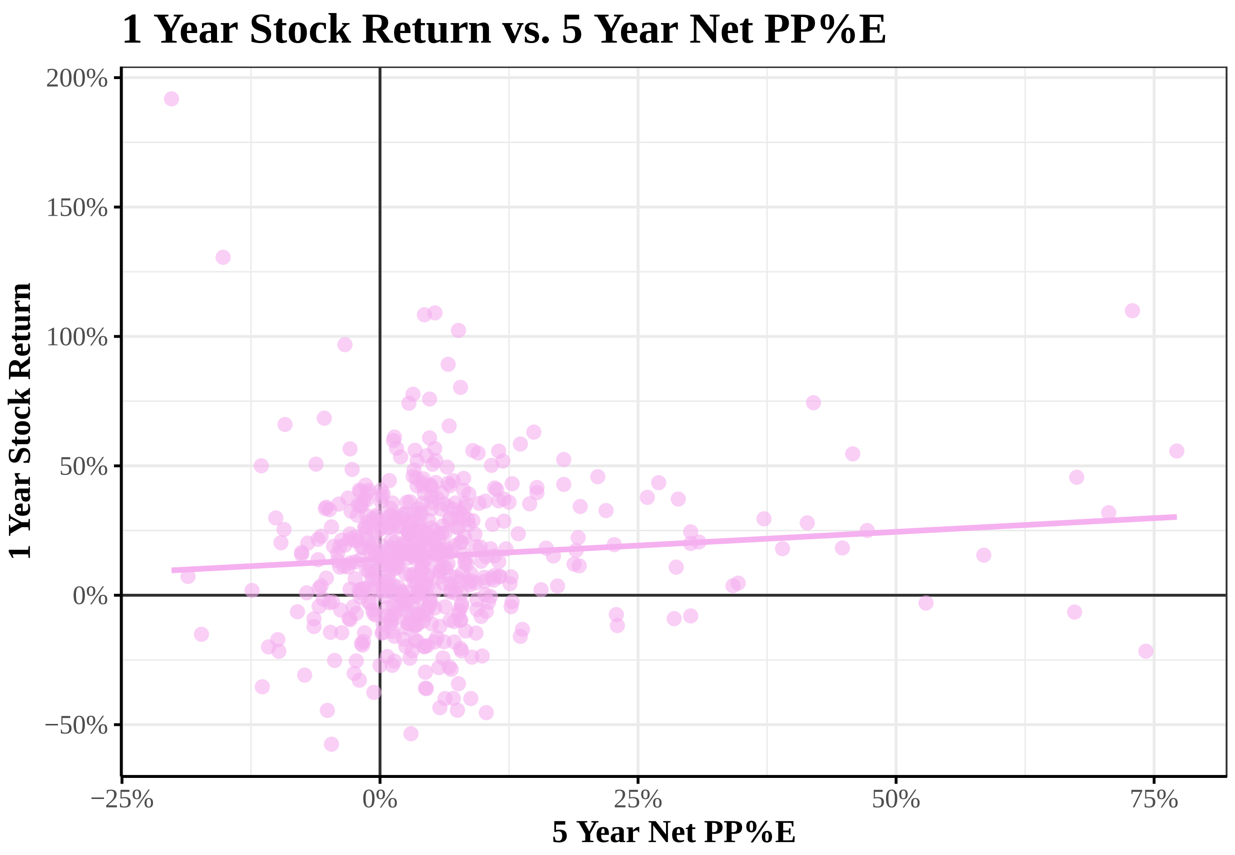
```
scatterplot2 <- ggplot(my_data_clean_1_clear, aes(x = Earnings.from.Cont..Ops.Margin.., y = Stock.Retur
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.6, color = '#9467bd') +
  labs(title = "1 Year Stock Return vs. Earnings from Continuing Operations Margin", x = "Earnings from
  geom_smooth(method = "lm", se = FALSE, color = "#9467bd") +
  theme_bw() + guides(color = 'none')

scatterplot2 +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100))
```



```r
scatterplot3 <- ggplot(my_data_clean_1_clear, aes(x = EBITDA3, y = Stock.Return..1.Year. - 1)) +
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.6, color = '#e89246') +
  labs(title = "1 Year Stock Return vs. 3 Year EBITDA", x = '3 Year EBITDA', y = "1 Year Stock Return")
  geom_smooth(method = "lm", se = FALSE, color = "#e89246") +
  theme_bw() + guides(color = 'none')

scatterplot3 +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100))
```



```r
scatterplot4 <- ggplot(my_data_clean_1_clear, aes(x = Net.PP.E5, y = Stock.Return..1.Year. - 1)) +
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.6, color = '#f5b0f0') +
  labs(title = "1 Year Stock Return vs. 5 Year Net PP%E", x = '5 Year Net PP%E', y = "1 Year Stock Retu
  geom_smooth(method = "lm", se = FALSE, color = "#f5b0f0") +
  theme_bw() + guides(color = 'none')

scatterplot4 +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100))
```



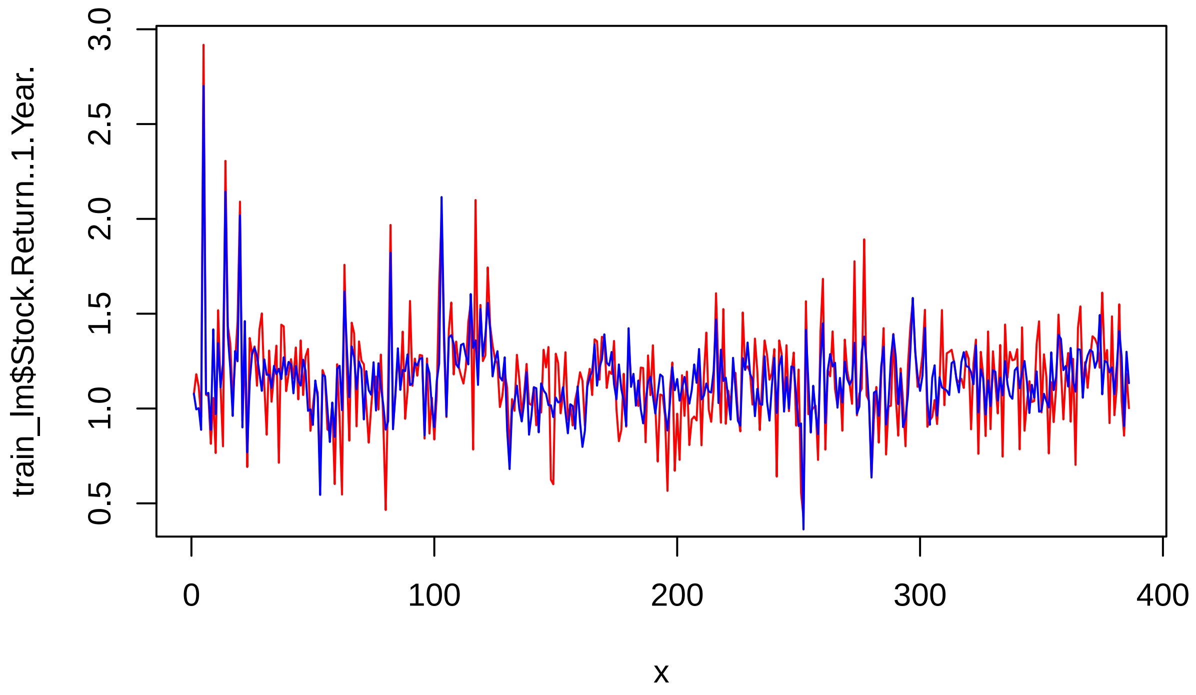
```r
scatterplot_creator <- function(x_values, y_values, data, color_value, x_name, y_name) {
scatterplot4 <- ggplot(data, aes(x = x_values, y = y_values - 1)) +
  geom_hline(yintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_vline(xintercept = 0, color = "black", size = 0.5, alpha = 0.8) +
  geom_point(size = 2, alpha = 0.6, color = color_value) +
  labs(title = sprintf("%s vs. %s", y_name, x_name), x = x_name, y = y_name) +
  geom_smooth(method = "lm", se = FALSE, color = color_value) +
  theme_bw() + guides(color = 'none')

scatterplot4 +
  theme(plot.title = element_text(family = "Times", size = 16, face = "bold")) +
  theme(axis.title = element_text(family = "Times", size = 12, face = "bold"),
        axis.text = element_text(family = "Times", size = 10),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(color = "black")) +
  scale_y_continuous(labels = scales::percent_format(scale = 100), breaks = seq(-1, 2.00, by = 0.50)) +
  scale_x_continuous(labels = scales::percent_format(scale = 100))

return(scatterplot4)
}
```

```r
gross_profit2 <- scatterplot_creator(my_data_clean_2_clear$Gross.Profit2, my_data_clean_2_clear$Stock.R
```

MULTIPLE LINEAR REGRESSION

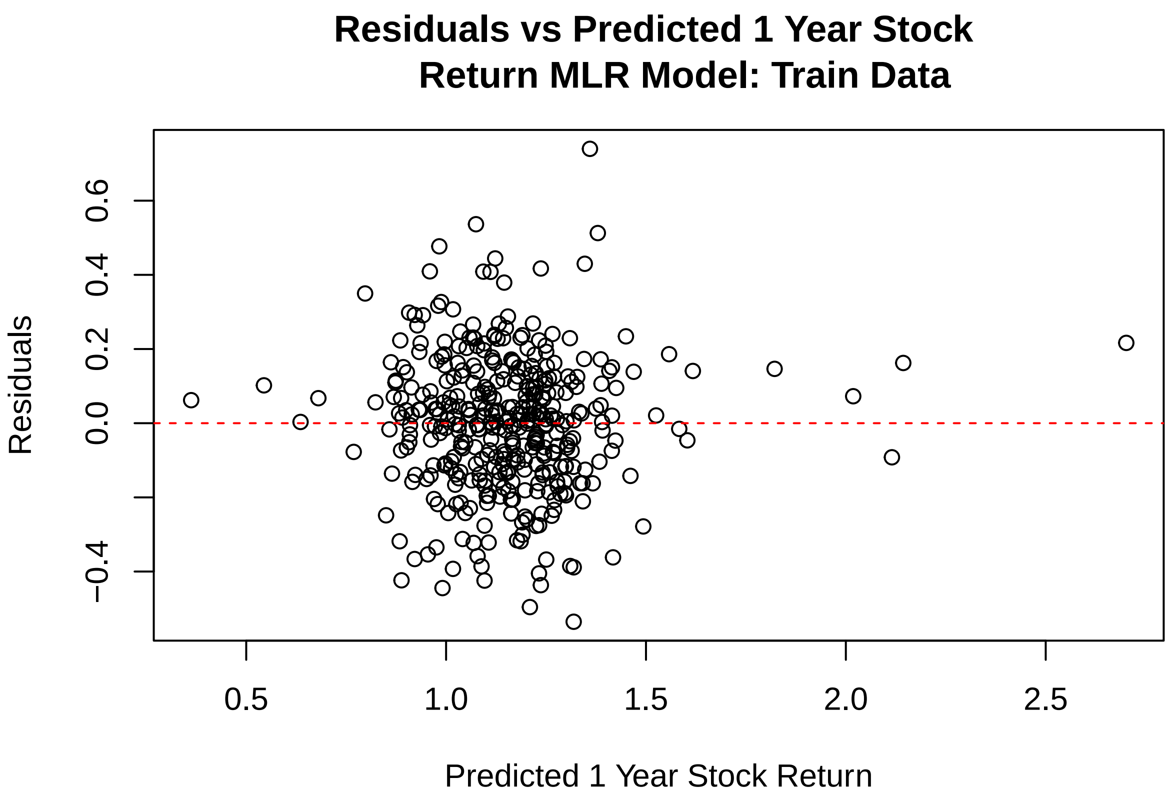
1 YEAR

```
# MLR model for 1 year stock return
naive_lm_1 <- lm((Stock.Return..1.Year.) ~ . - Year - Company, data = train_lm)

# visualize the model, actual and predicted data
x = 1:length(train_lm$Stock.Return..1.Year.)
plot(x, train_lm$Stock.Return..1.Year., col = "red", type = "l")
lines(x, (predict(naive_lm_1, data=train_lm)), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

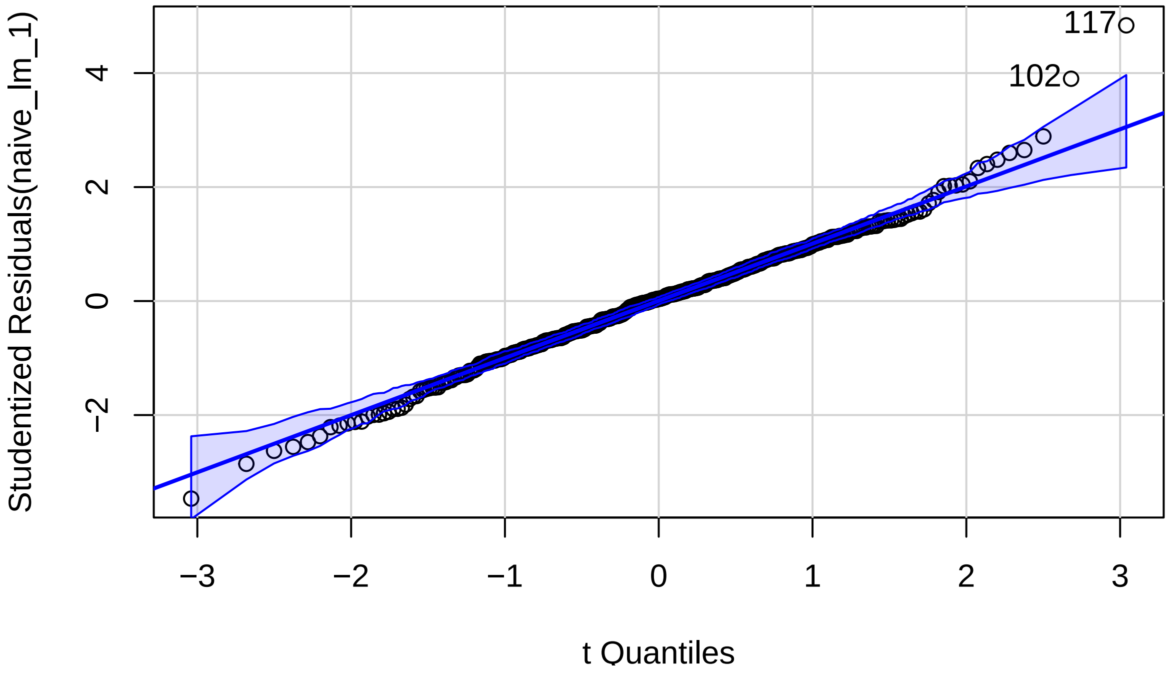


```
summary(naive_lm_1)
```

```
residuals_lm_1 <- residuals(naive_lm_1)
fitted_values_lm_1 <- fitted(naive_lm_1)

# Residuals of training data plot
plot(fitted_values_lm_1, residuals_lm_1, main = "Residuals vs Predicted 1 Year Stock
     Return MLR Model: Train Data", xlab = "Predicted 1 Year Stock Return", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

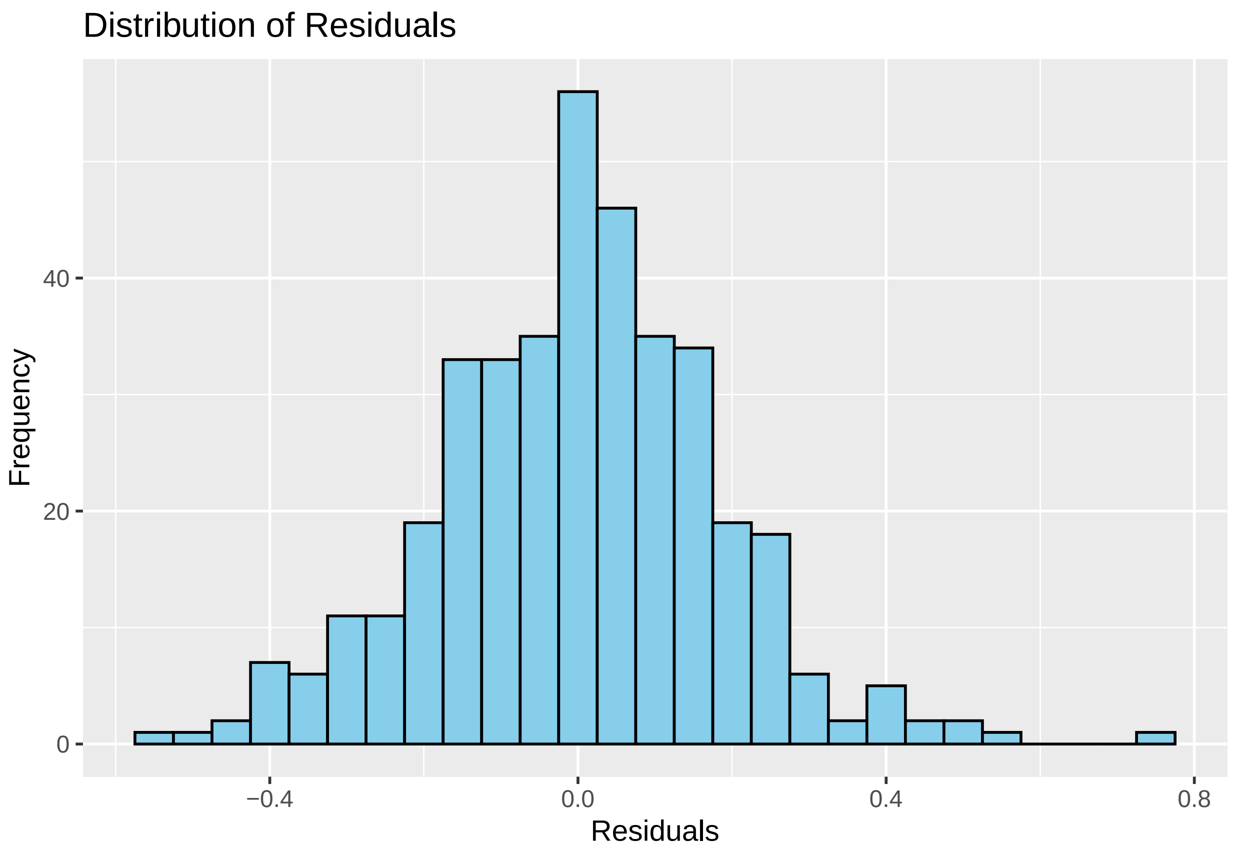


```r
# Tests
ncvTest(naive_lm_1) #constant variance
dwtest(naive_lm_1) #independence
shapiro.test(residuals(naive_lm_1)) #normality
qqPlot(naive_lm_1)
```

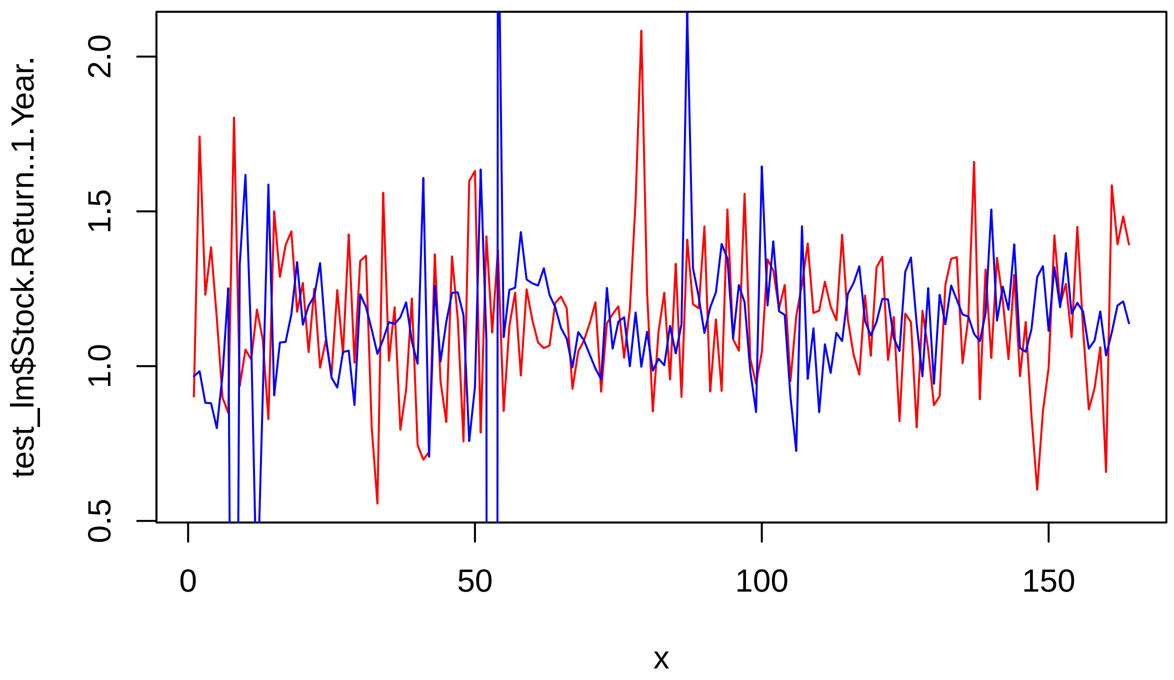


```r
residuals_df_lm_1 <- as.data.frame(residuals_lm_1)
```

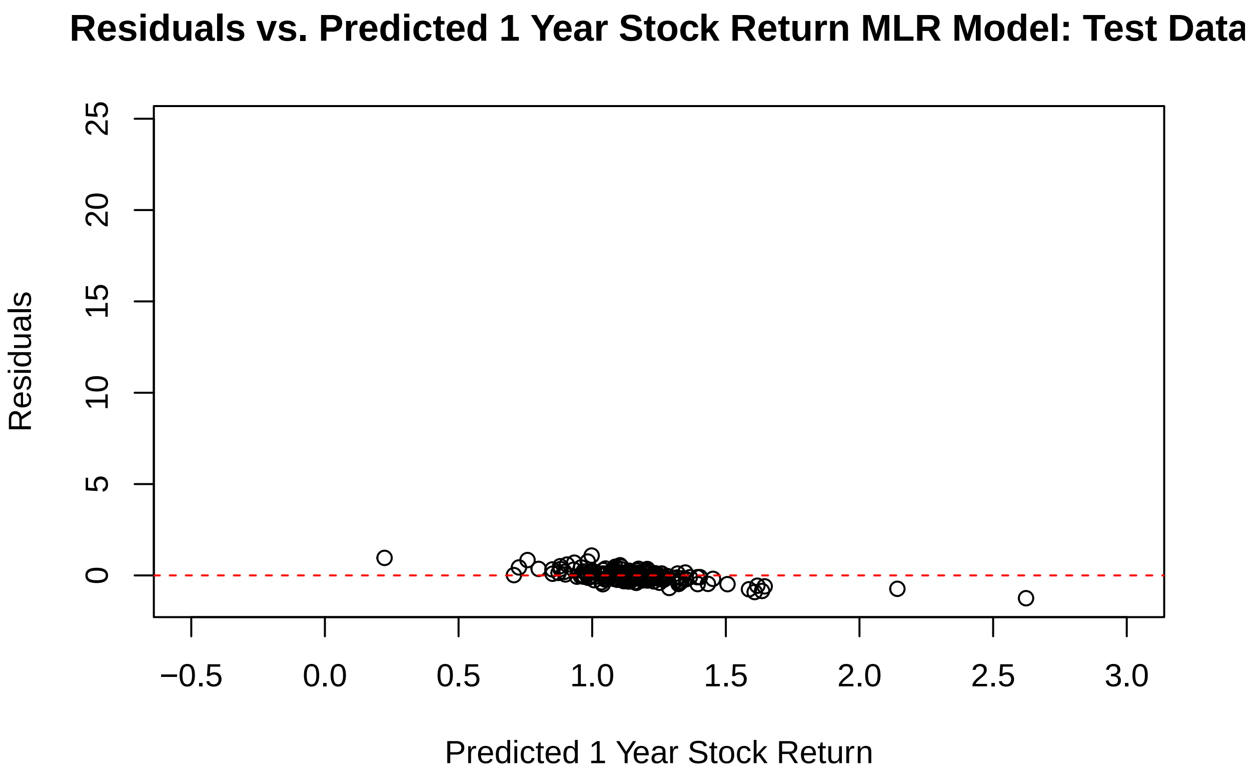
```r
# Histogram of residuals
ggplot(residuals_df_lm_1, aes(x = residuals_lm_1)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```



```r
cat("The mean of the distribution is:", paste(mean((residuals_df_lm_1$residuals_lm_1)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df_lm_1$residuals_lm_1)), "\n"
cat("The range of the distribution is:[", paste(range((residuals_df_lm_1$residuals_lm_1))), "]\n")

# Calculations for MAE
test_lm$Company <- factor(test_lm$Company, levels = levels(naive_lm_1$data$Company))

predicted_values_lm_1 <- as.data.frame(predict(naive_lm_1, newdata = test_lm, interval = "confidence",

predicted_values_lm_1$actual <- test_lm$Stock.Return..1.Year.

predicted_values_lm_1$resid <- (predicted_values_lm_1$actual) - predicted_values_lm_1$fit

predicted_values_lm_1$resid_mag <- abs(predicted_values_lm_1$resid)

mae_lm_1 <- (mean(predicted_values_lm_1$resid_mag))

# Visualize the model, actual and predicted data
x = 1:length(test_lm$Stock.Return..1.Year.)
plot(x, test_lm$Stock.Return..1.Year., col = "red", type = "l")
lines(x, predict(naive_lm_1, newdata=test_lm), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

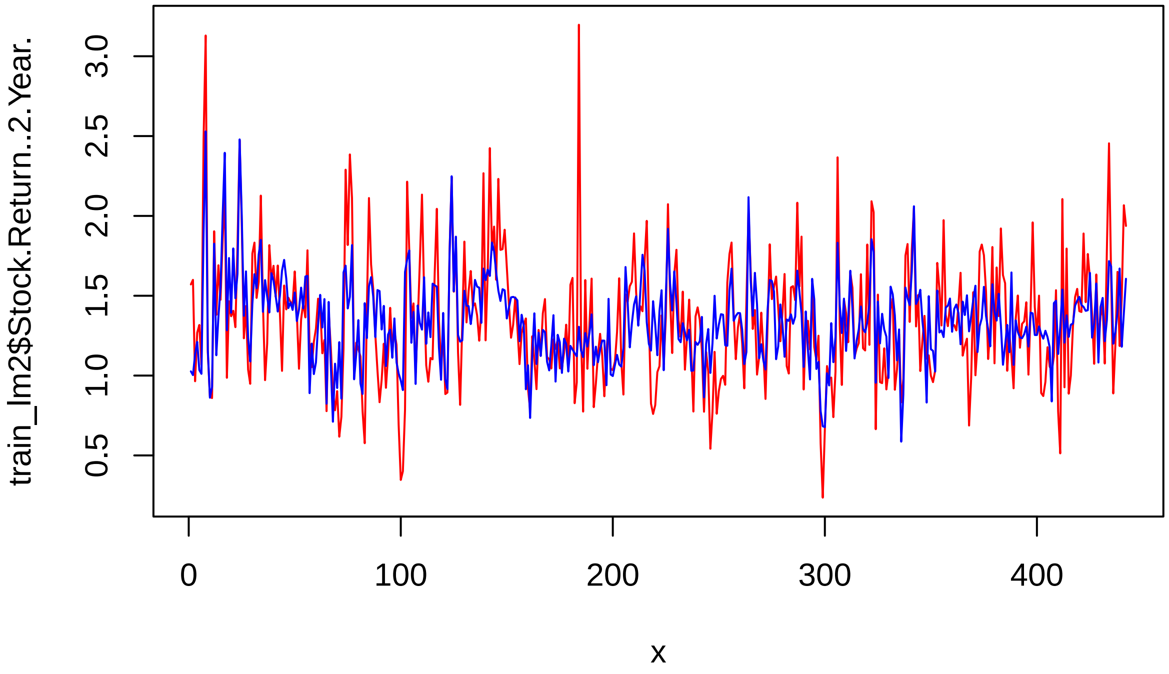


```
# Create a scatter plot of residuals on testing data
plot(x = predicted_values_lm_1$fit, y = predicted_values_lm_1$resid, main = "Residuals vs. Predicted 1 Y
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals", xlim = c(-0.5,3))

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



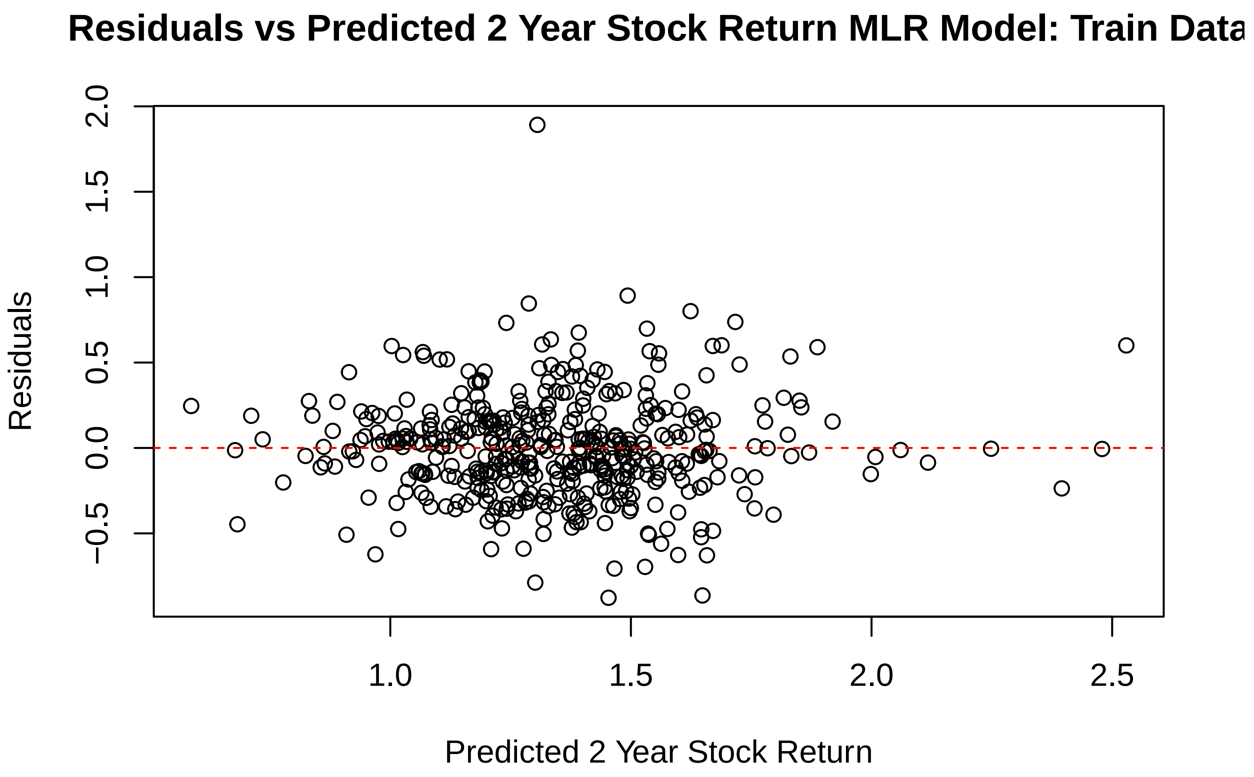
2 YEAR

```
# MLR model for 2 year stock return
naive_lm_2 <- lm(Stock.Return..2.Year. ~ . - Year - Company, data = train_lm2)

# Visualize the model, actual and predicted data
x = 1:length(train_lm2$Stock.Return..2.Year.)
plot(x, train_lm2$Stock.Return..2.Year., col = "red", type = "l")
lines(x, predict(naive_lm_2, newdata=train_lm2), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

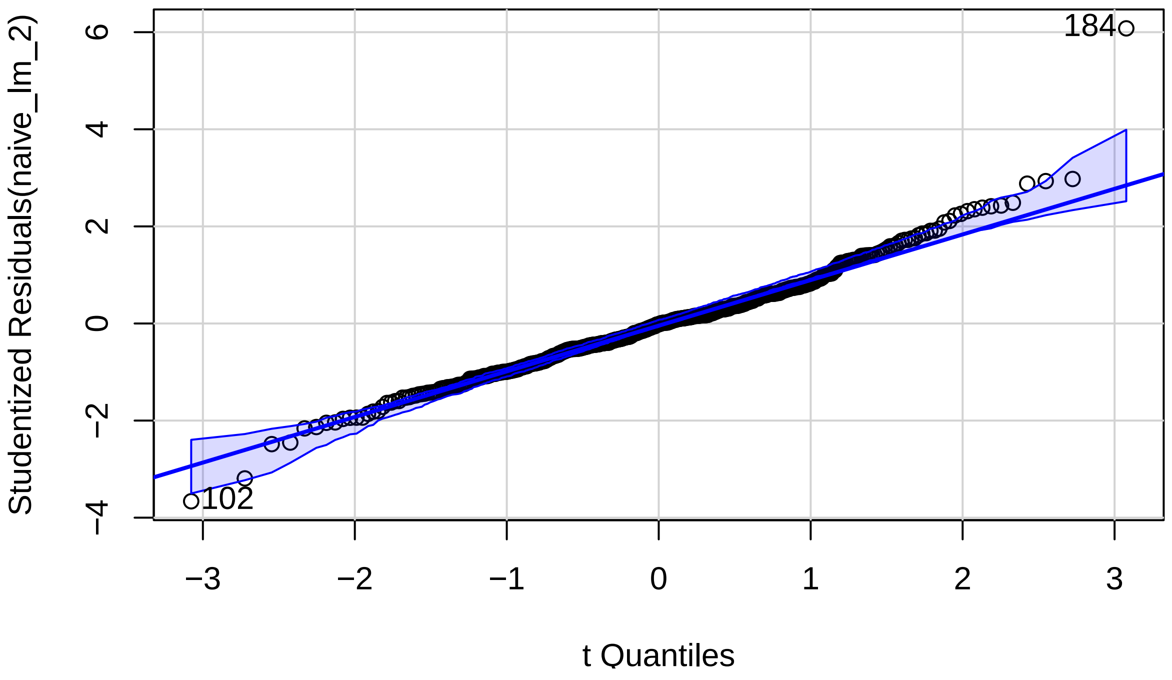


```
summary(naive_lm_2)
```

```
residuals_lm_2 <- residuals(naive_lm_2)
fitted_values_lm_2 <- fitted(naive_lm_2)

# Residuals of training data plot
plot(fitted_values_lm_2, residuals_lm_2, main = "Residuals vs Predicted 2 Year Stock Return MLR Model: "
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

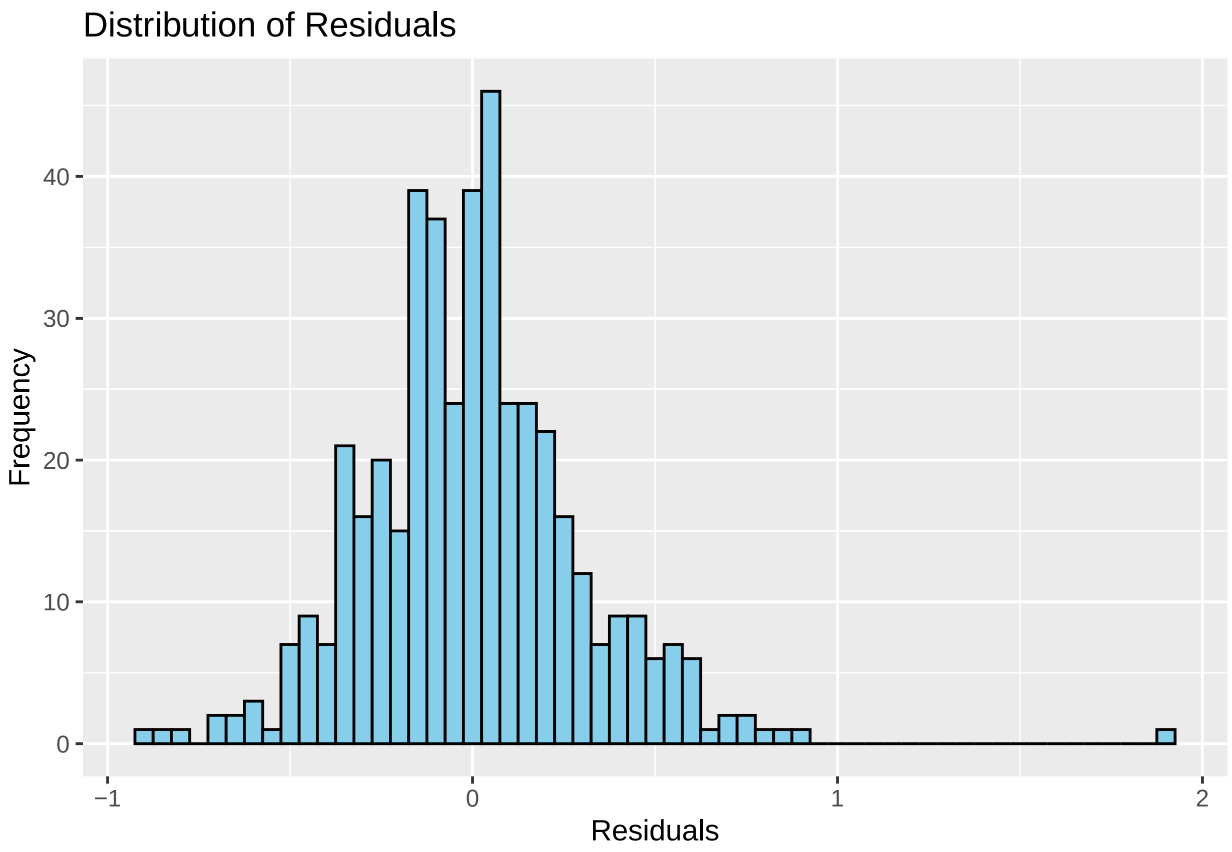


```r
# Tests
ncvTest(naive_lm_2) #constant variance
dwtest(naive_lm_2) #independence
shapiro.test(residuals(naive_lm_2)) #normality
qqPlot(naive_lm_2)
```

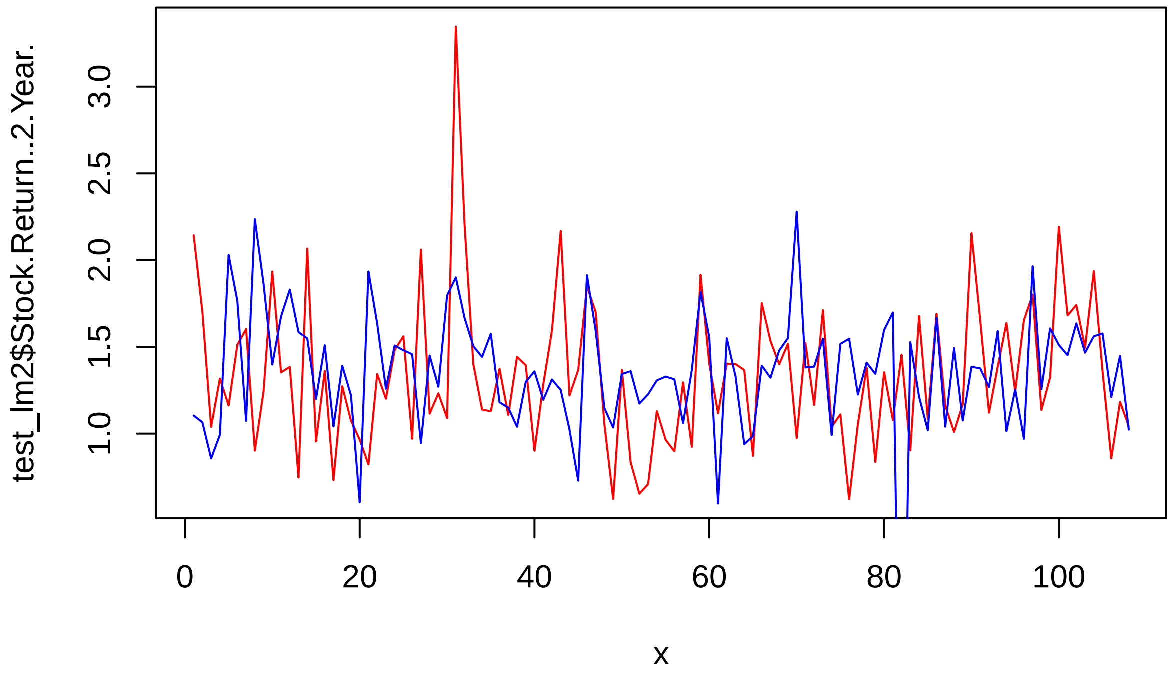


```r
residuals_df_lm_2 <- as.data.frame(residuals_lm_2)

# Histogram of residuals
```

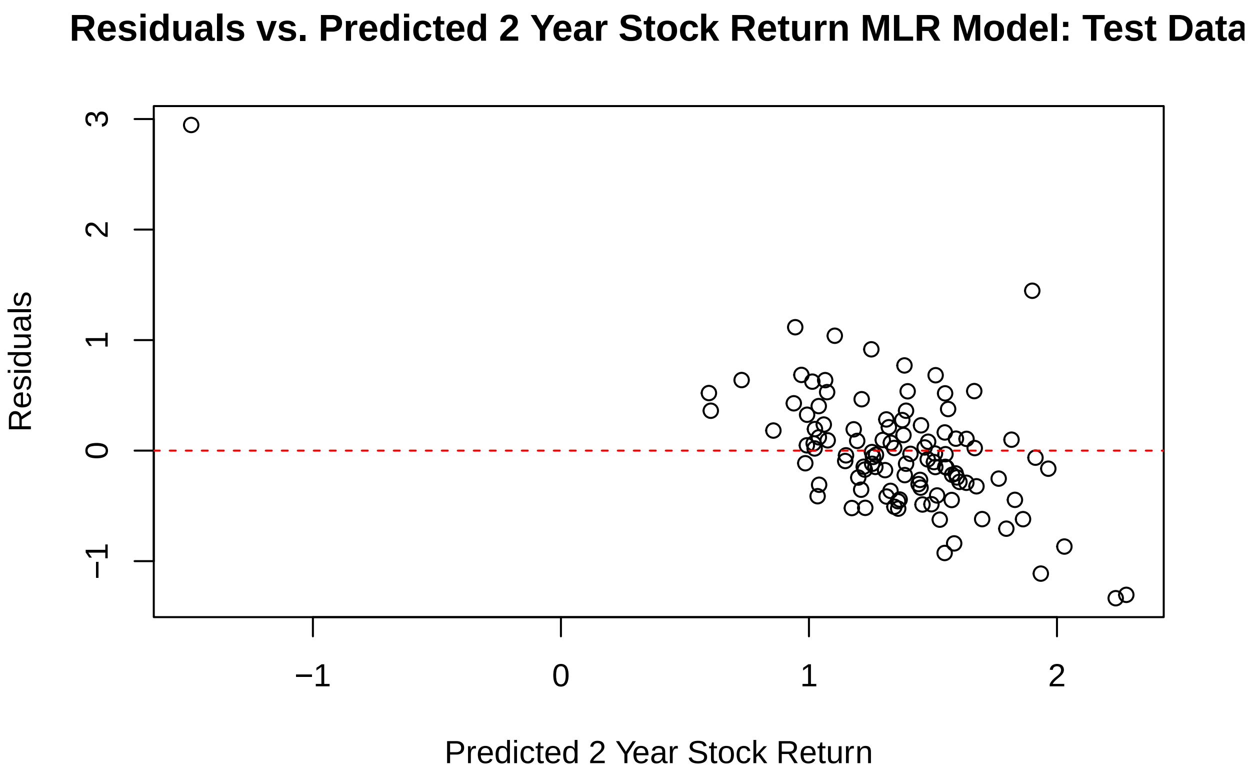
```r
ggplot(residuals_df_lm_2, aes(x = residuals_lm_2)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```



```r
cat("The mean of the distribution is:", paste(mean((residuals_df_lm_2$residuals_lm_2)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df_lm_2$residuals_lm_2)), "\n"
cat("The range of the distribution is:[", paste(range((residuals_df_lm_2$residuals_lm_2))), "]\n")

# Calculations for MAE

test_lm2$Company <- factor(test_lm2$Company, levels = levels(naive_lm_2$data$Company))

predicted_values_lm_2 <- as.data.frame(predict(naive_lm_2, newdata = test_lm2, interval = "confidence",

predicted_values_lm_2$actual <- test_lm2$Stock.Return..2.Year.

predicted_values_lm_2$resid <- (predicted_values_lm_2$actual) - predicted_values_lm_2$fit

predicted_values_lm_2$resid_mag <- abs(predicted_values_lm_2$resid)

mae_lm_2 <- (mean(predicted_values_lm_2$resid_mag))

# Visualize the model, actual and predicted data
x = 1:length(test_lm2$Stock.Return..2.Year.)
plot(x, test_lm2$Stock.Return..2.Year., col = "red", type = "l")
lines(x, predict(naive_lm_2, newdata=test_lm2), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

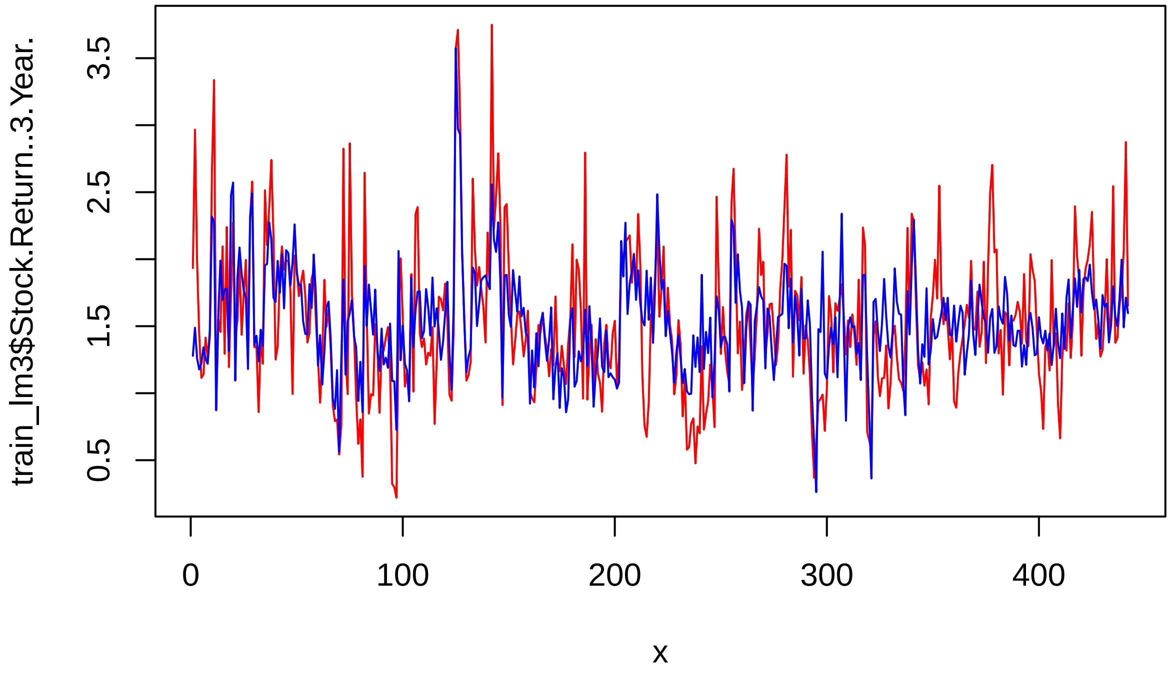


```
# Create a scatter plot of residuals for testing data
plot(x = predicted_values_lm_2$fit, y = predicted_values_lm_2$resid, main = "Residuals vs. Predicted 2 Y
     xlab = "Predicted 2 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



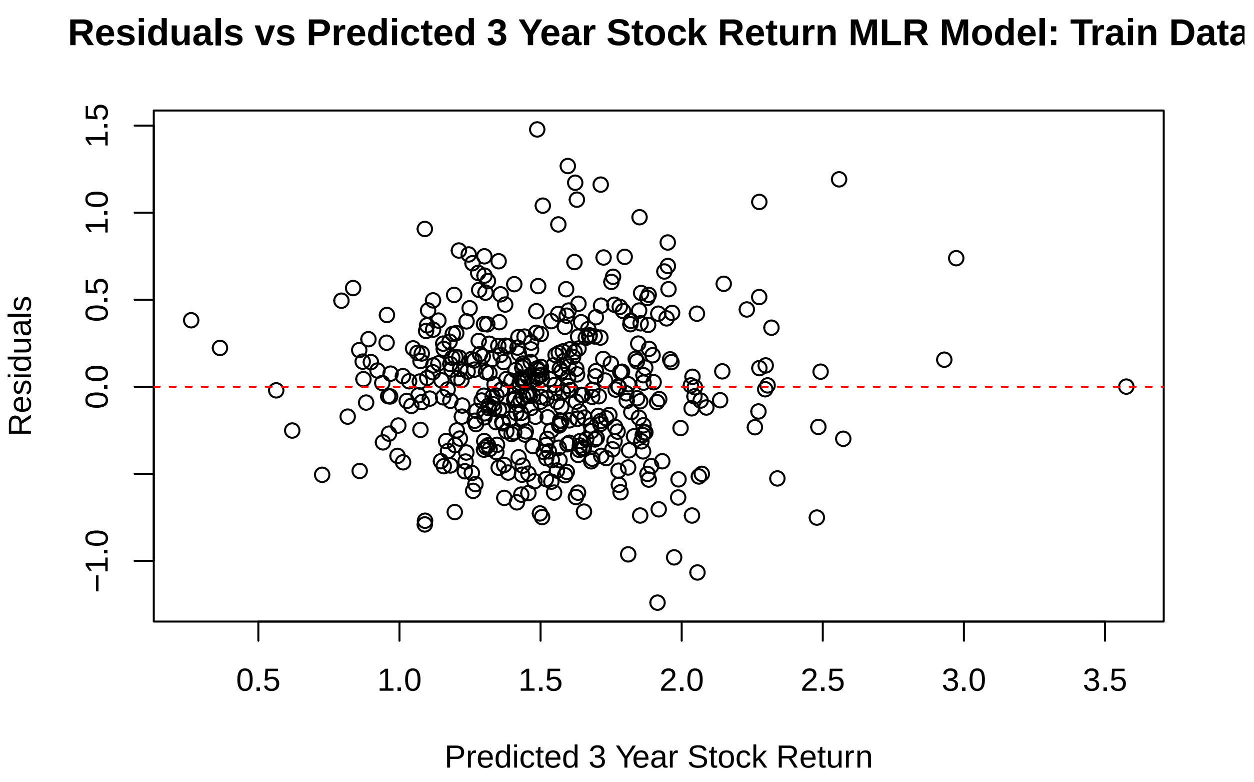
3 YEAR

```
#MLR model for 3 year stock return
naive_lm_3 <- lm(Stock.Return..3.Year. ~ . - Company - Year, data = train_lm3)

# Visualize the model, actual and predicted data
x = 1:length(train_lm3$Stock.Return..3.Year.)
plot(x, train_lm3$Stock.Return..3.Year., col = "red", type = "l")
lines(x, predict(naive_lm_3, newdata=train_lm3), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

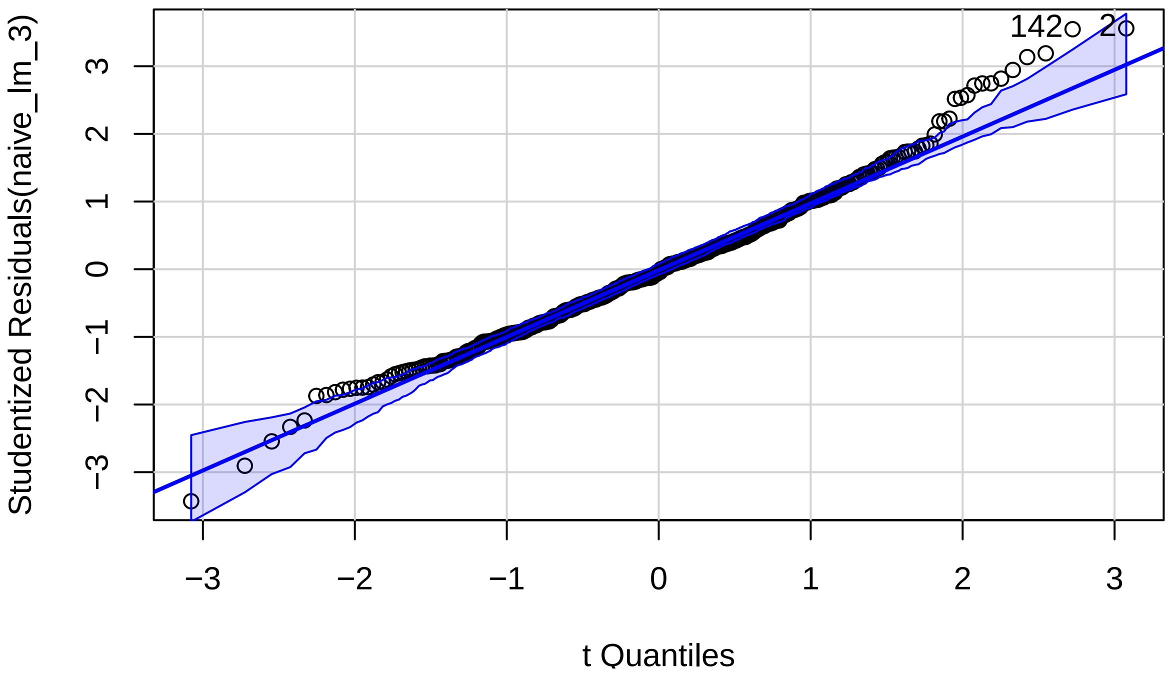


```
summary(naive_lm_3)
```

```
residuals_lm_3 <- residuals(naive_lm_3)
fitted_values_lm_3 <- fitted(naive_lm_3)

# Residuals of training data plot
plot(fitted_values_lm_3, residuals_lm_3, main = "Residuals vs Predicted 3 Year Stock Return MLR Model: 
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

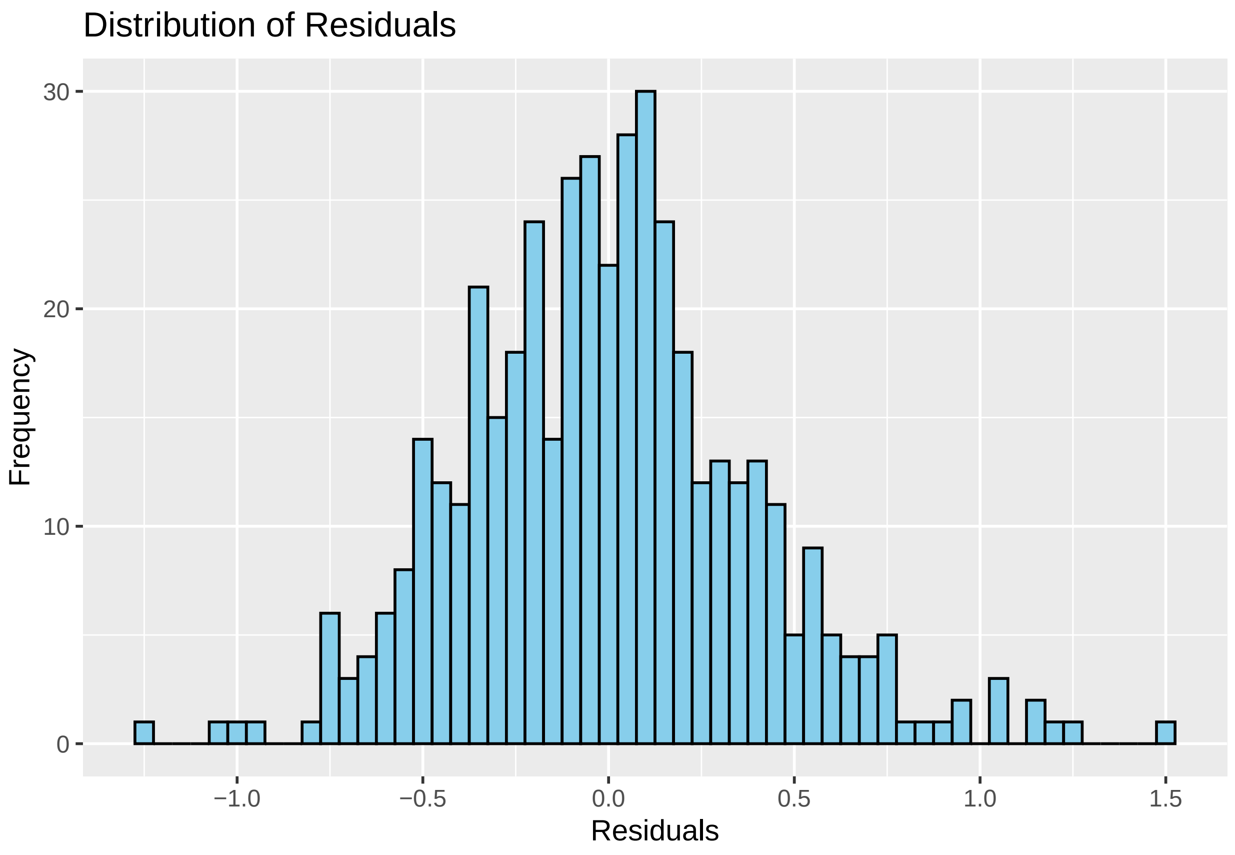


```r
# Tests
ncvTest(naive_lm_3) #constant variance
dwtest(naive_lm_3) #independence
shapiro.test(residuals(naive_lm_3)) #normality
qqPlot(naive_lm_3)
```

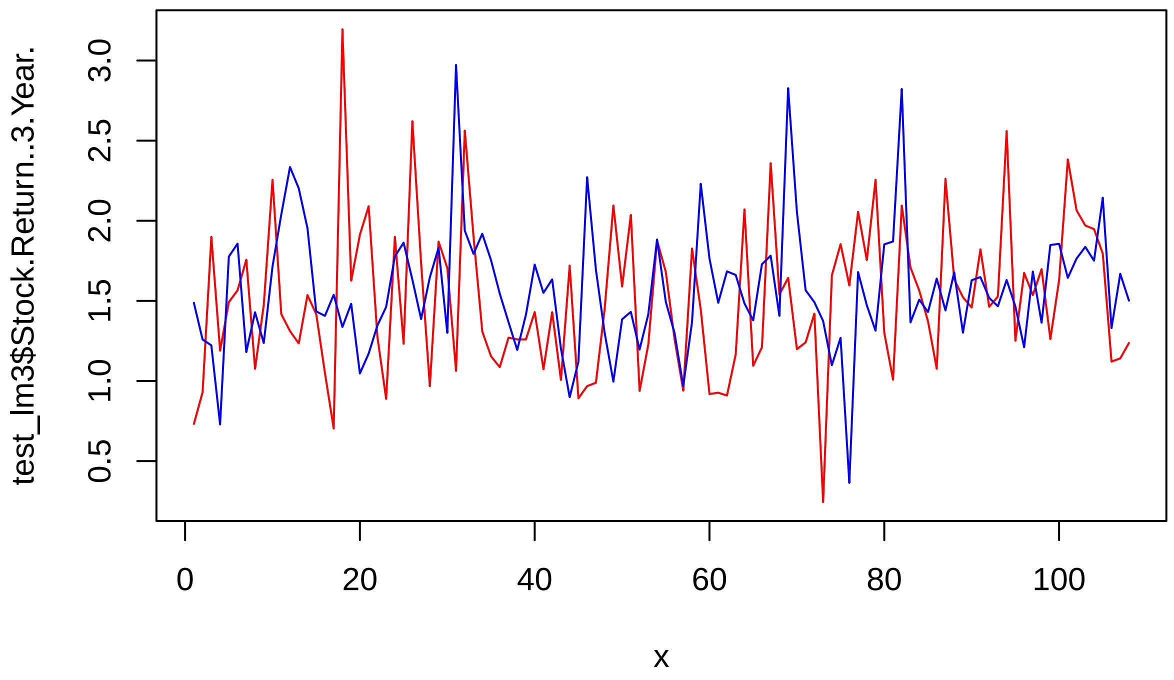


```r
residuals_df_lm_3 <- as.data.frame(residuals_lm_3)

# Histogram of residuals
```

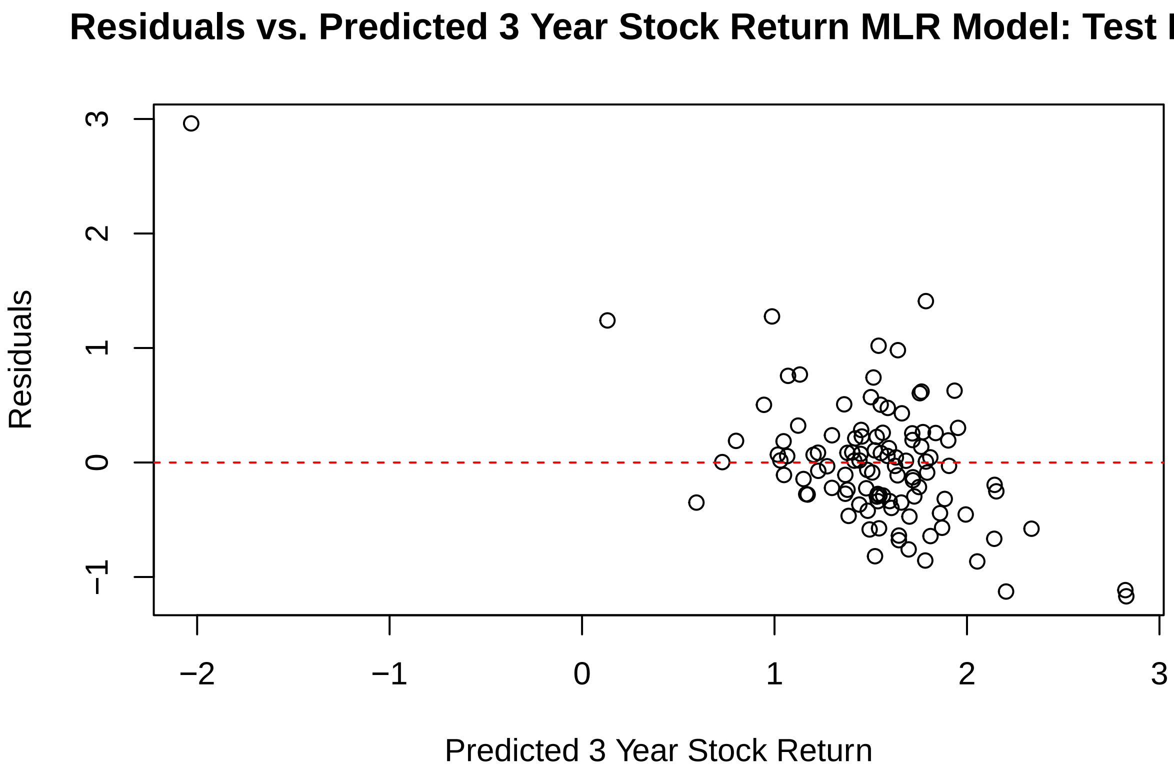
```r
ggplot(residuals_df_lm_3, aes(x = residuals_lm_3)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```



```r
cat("The mean of the distribution is:", paste(mean((residuals_df_lm_3$residuals_lm_3)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df_lm_3$residuals_lm_3)), "\n"
cat("The range of the distribution is:[", paste(range((residuals_df_lm_3$residuals_lm_3))), "]\n")

# Calculations for MAE
test_lm3$Company <- factor(test_lm3$Company, levels = levels(naive_lm_3$data$Company))

predicted_values_lm_3 <- as.data.frame(predict(naive_lm_3, newdata = test_lm3, interval = "confidence",

predicted_values_lm_3$actual <- test_lm3$Stock.Return..3.Year.

predicted_values_lm_3$resid <- (predicted_values_lm_3$actual) - predicted_values_lm_3$fit

predicted_values_lm_3$resid_mag <- abs(predicted_values_lm_3$resid)

mae_lm_3 <- (mean(predicted_values_lm_3$resid_mag))

x = 1:length(test_lm3$Stock.Return..3.Year.)
plot(x, test_lm3$Stock.Return..3.Year., col = "red", type = "l")
lines(x, predict(naive_lm_3, newdata=test_lm2), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



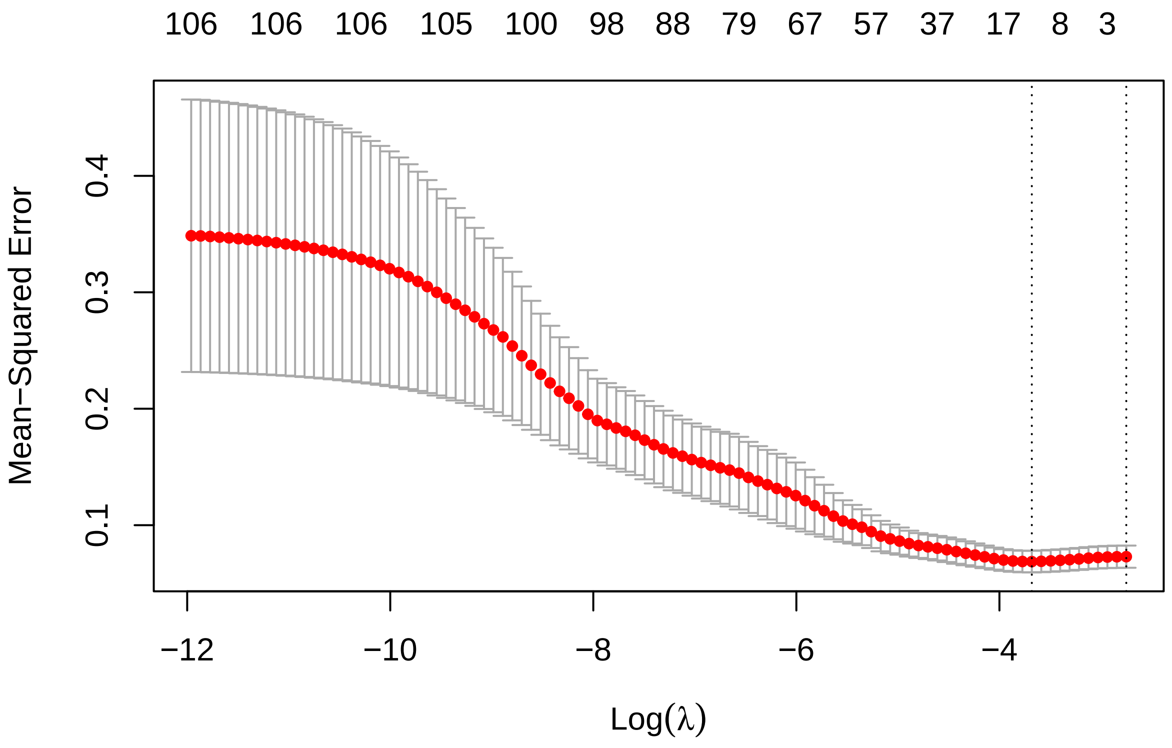
```r
# Create a scatter plot of residuals of testing data
plot(x = predicted_values_lm_3$fit, y = predicted_values_lm_3$resid, main = "Residuals vs. Predicted 3 Y
     xlab = "Predicted 3 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



LASSO REGRESSION

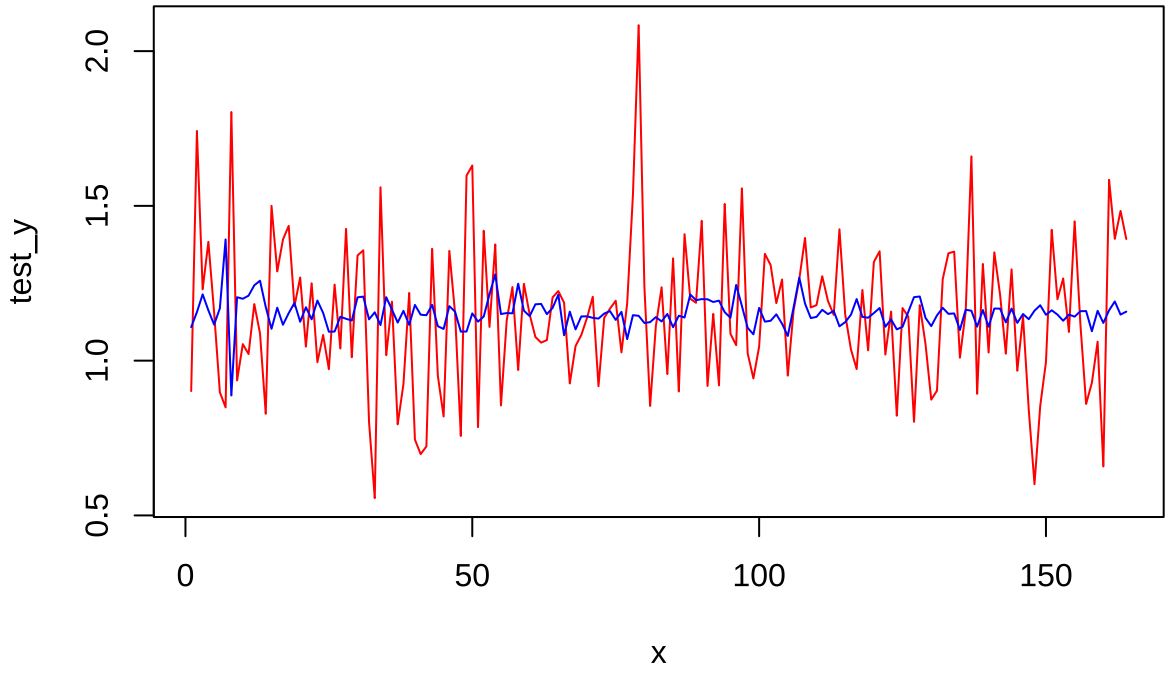
1 YEAR

```
#LASSO regression 1 year
response <- train_lm$Stock.Return..1.Year.

# Identify the column to move
column_to_move <- train_lm$Stock.Return..1.Year.

# Remove the column from its current position
train_lm <- train_lm[, !(names(train_lm) %in% "Stock.Return..1.Year.")]

# Add the column to the end of the data frame
train_lm$Stock.Return..1.Year. <- column_to_move

independents = data.matrix(train_lm[, -ncol(train_lm)])
independents1 = data.matrix(test_lm[, -ncol(test_lm)])

#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(independents, response, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda

#produce plot of test MSE by lambda value
plot(cv_model)
```

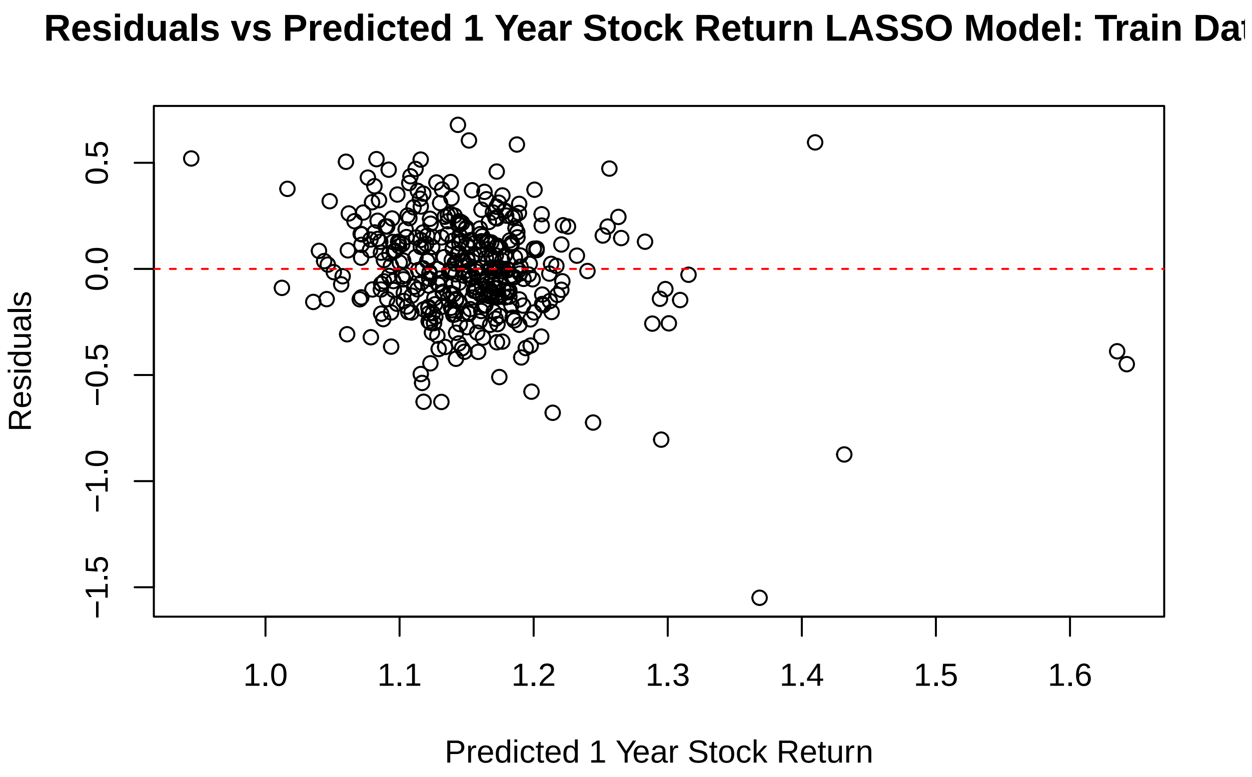


```
#find coefficients of best model
best_model <- glmnet(independents, response, alpha = 1, lambda = best_lambda)
```

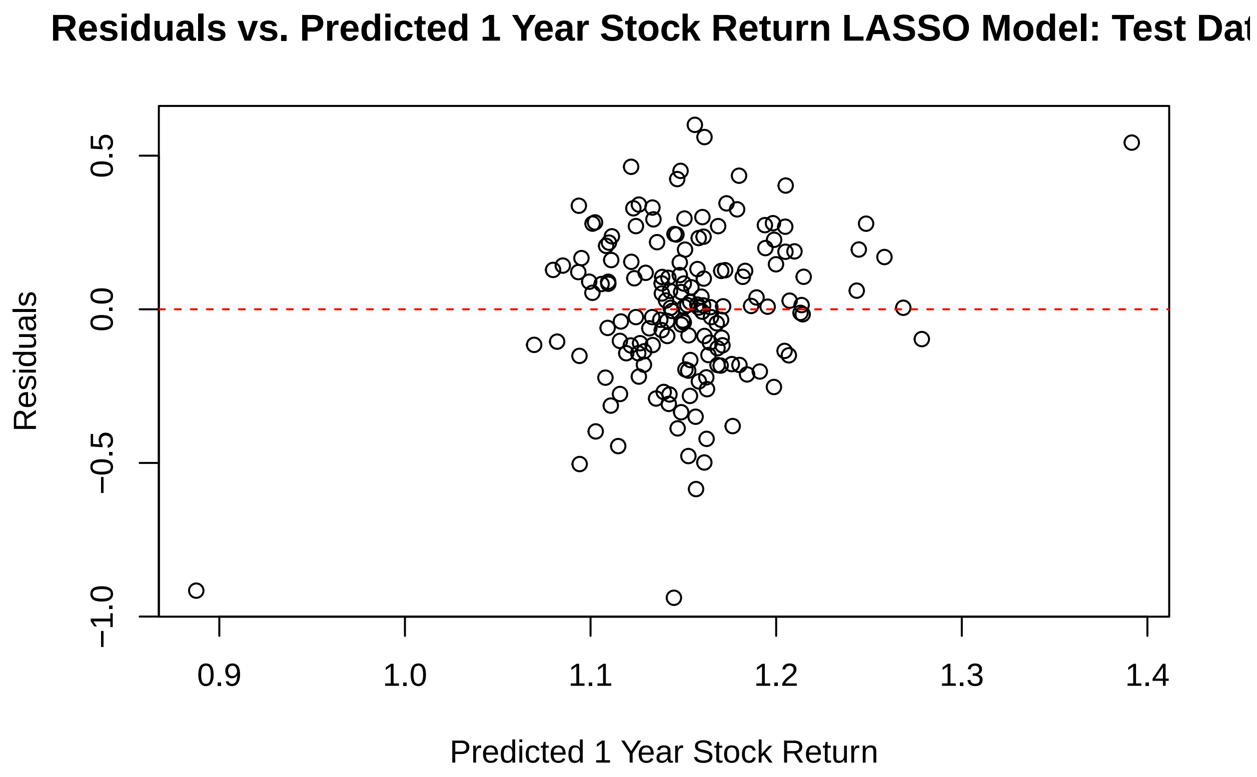
```
coef(best_model)

# Predict on the training data
predictions_lasso <- predict(best_model, type = "response", newx = independents)

coefficients_tidy <- tidy(best_model)[, c("term", "estimate")]
print(coefficients_tidy, digits = 4)

# Calculate R^2 manually for training
SS_Residual <- sum((response - predictions_lasso)^2)
mae_lasso_training_1 <-  mean(abs(predictions_lasso - train_lm$Stock.Return..1.Year.))
SS_Total <- sum((response - mean(response))^2)
r_squared <- 1 - (SS_Residual / SS_Total)

# Make predictions on the testing data
lasso_predictions <- predict(best_model, newx = test_x)

# Calculate Mean Absolute Error (MAE)
mae_lasso_test_1 <- mean(abs(lasso_predictions - test_y))

# Visualize the model, actual and predicted data
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l")
lines(x, lasso_predictions, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```
residuals_lasso_1 <- predictions_lasso - train_lm$Stock.Return..1.Year.

# Residuals plot for training data
plot(predictions_lasso, residuals_lasso_1, main = "Residuals vs Predicted 1 Year Stock Return LASSO Mode
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

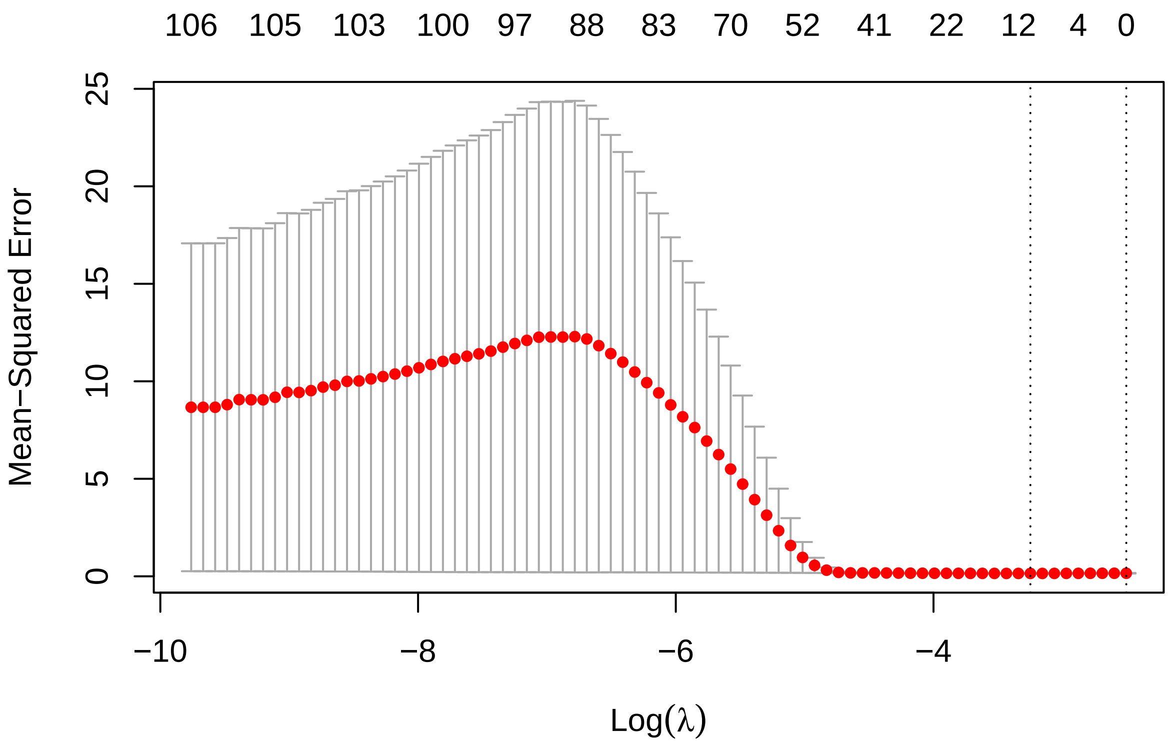


```r
# Create a scatter plot of residuals for test data
plot(x = lasso_predictions, y = lasso_predictions - test_y, main = "Residuals vs. Predicted 1 Year Stoc
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```

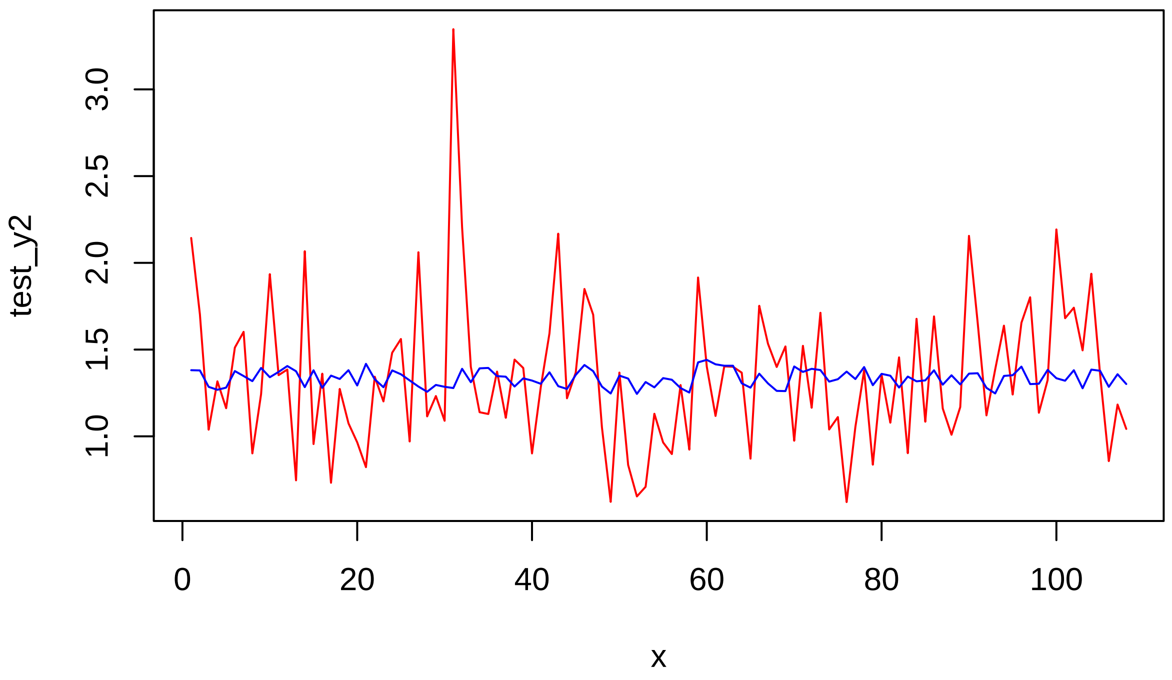


2 YEAR

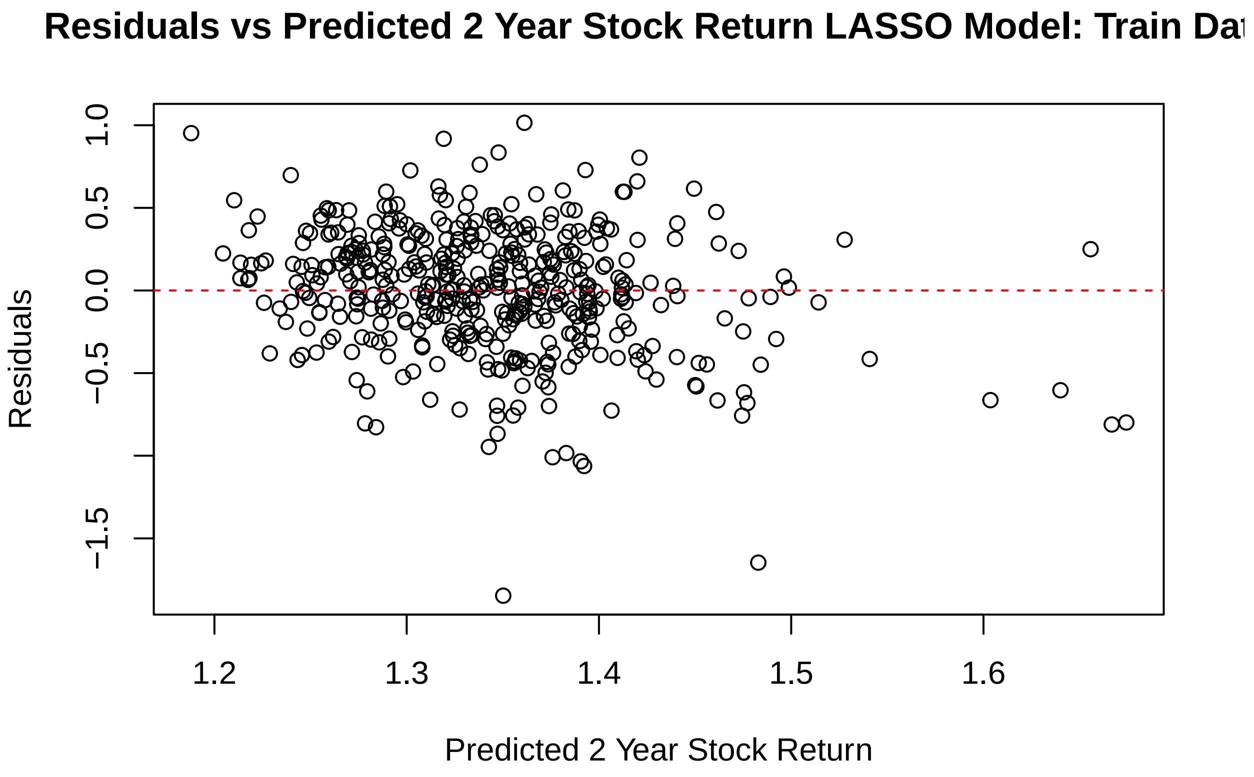
```
#LASSO regression 2 year
response1 <- train_lm2$Stock.Return..2.Year.

# Identify the column to move
column_to_move2 <- train_lm2$Stock.Return..2.Year.

# Remove the column from its current position
train_lm2 <- train_lm2[, !(names(train_lm2) %in% "Stock.Return..2.Year.")]

# Add the column to the end of the data frame
train_lm2$Stock.Return..2.Year. <- column_to_move2

independents2 = data.matrix(train_lm2[, -ncol(train_lm2)])
independents3 = data.matrix(test_lm2[, -ncol(test_lm2)])

#perform k-fold cross-validation to find optimal lambda value
cv_model2 <- cv.glmnet(independents2, response1, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda2 <- cv_model2$lambda.min
best_lambda2

#produce plot of test MSE by lambda value
plot(cv_model2)
```



```r
#find coefficients of best model
best_model2 <- glmnet(independents2, response1, alpha = 1, lambda = best_lambda2)
coef(best_model2)

# Predict on the training data
predictions_lasso2 <- predict(best_model2, type = "response", newx = independents2)

coefficients_tidy2 <- tidy(best_model2)[, c("term", "estimate")]
print(coefficients_tidy2, digits = 4)

# Calculate R^2 manually for training
SS_Residual2 <- sum((response1 - predictions_lasso2)^2)
mae_lasso_training_2 <-  mean(abs(predictions_lasso2 - train_lm2$Stock.Return..2.Year.))
SS_Total2 <- sum((response1 - mean(response1))^2)
r_squared2 <- 1 - (SS_Residual2 / SS_Total2)

# Make predictions on the testing data
lasso_predictions2 <- predict(best_model2, newx = test_x2)

# Calculate Mean Absolute Error (MAE)
mae_lasso_test_2 <- mean(abs(lasso_predictions2 - test_y2))

# Visualize the model, actual and predicted data
x = 1:length(test_y2)
plot(x, test_y2, col = "red", type = "l")
lines(x, lasso_predictions2, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

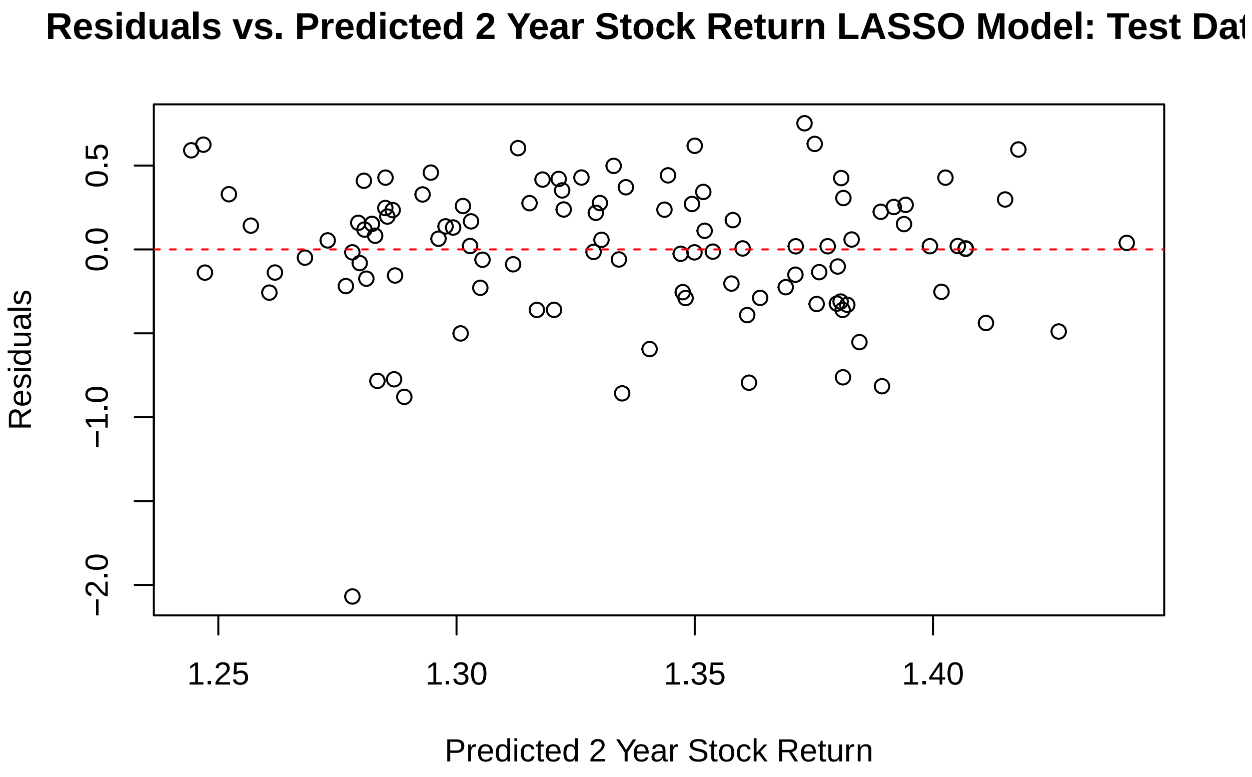


```
residuals_lasso_2 <- predictions_lasso2 - train_lm2$Stock.Return..2.Year.

# Residuals plot of training data
plot(predictions_lasso2, residuals_lasso_2, main = "Residuals vs Predicted 2 Year Stock Return LASSO Mod
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```
# Create a scatter plot of residuals of testing data
plot(x = lasso_predictions2, y = lasso_predictions2 - test_y2, main = "Residuals vs. Predicted 2 Year St
     xlab = "Predicted 2 Year Stock Return", ylab = "Residuals")
```

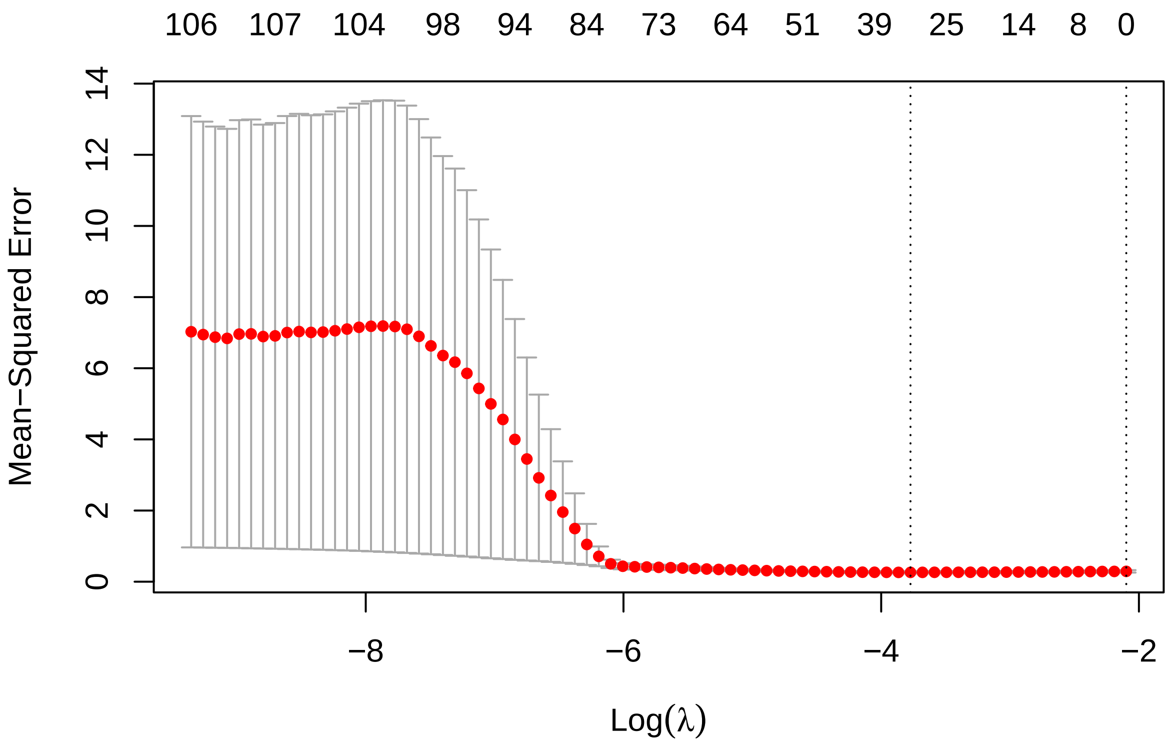
```
# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



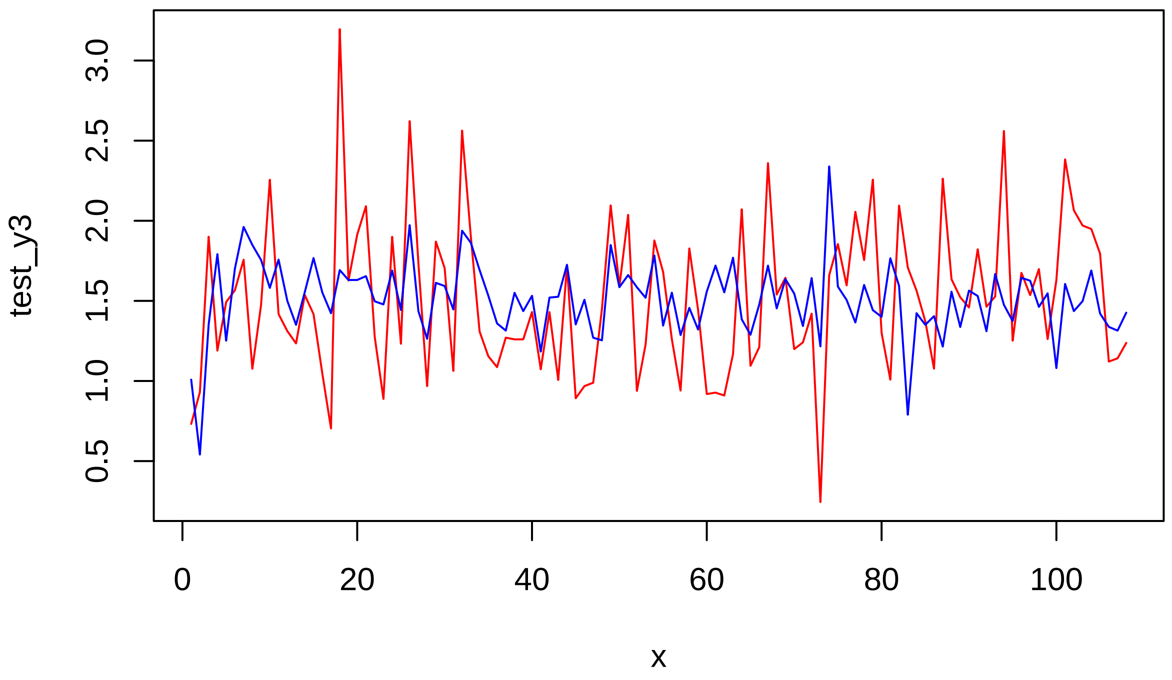
3 YEAR

```
#LASSO regression 3 year
response2 <- train_lm3$Stock.Return..3.Year.

# Identify the column to move
column_to_move3 <- train_lm3$Stock.Return..3.Year.

# Remove the column from its current position
train_lm3 <- train_lm3[, !(names(train_lm3) %in% "Stock.Return..3.Year.")]

# Add the column to the end of the data frame
train_lm3$Stock.Return..3.Year. <- column_to_move3

independents4 = data.matrix(train_lm3[, -ncol(train_lm3)])
independents5 = data.matrix(test_lm3[, -ncol(test_lm3)])

#perform k-fold cross-validation to find optimal lambda value
cv_model3 <- cv.glmnet(independents4, response2, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda3 <- cv_model3$lambda.min
best_lambda3

#produce plot of test MSE by lambda value
```

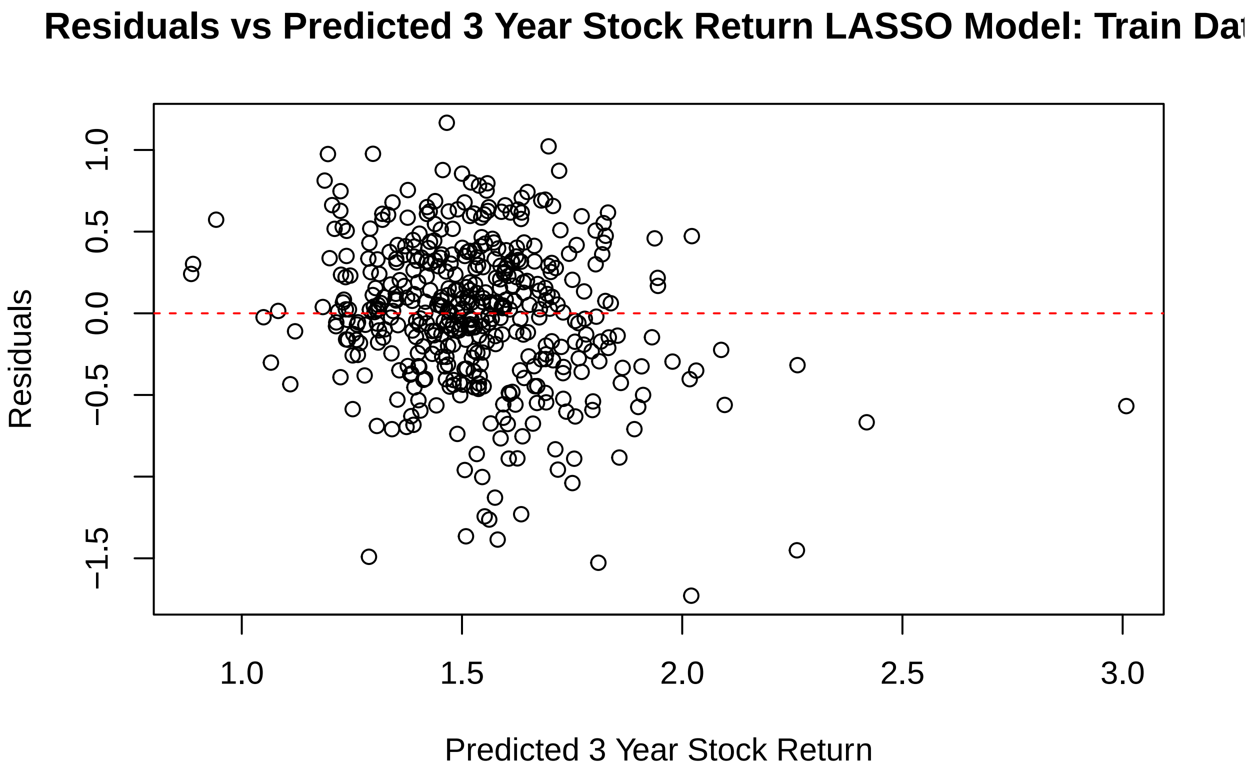
```
plot(cv_model3)
```



```
#find coefficients of best model
best_model3 <- glmnet(independents4, response2, alpha = 1, lambda = best_lambda3)
coef(best_model3)

# Predict on the training data
predictions_lasso3 <- predict(best_model3, type = "response", newx = independents4)

coefficients_tidy3 <- tidy(best_model3)[, c("term", "estimate")]
print(coefficients_tidy3, digits = 4)

# Calculate R^2 manually for training
SS_Residual3 <- sum((response2 - predictions_lasso3)^2)
mae_lasso_training_3 <-  mean(abs(predictions_lasso3 - train_lm3$Stock.Return..3.Year.))
SS_Total3 <- sum((response2 - mean(response2))^2)
r_squared3 <- 1 - (SS_Residual3 / SS_Total3)

# Make predictions on the testing data
lasso_predictions3 <- predict(best_model3, newx = test_x3)

# Calculate Mean Absolute Error (MAE)
mae_lasso_test_3 <- mean(abs(lasso_predictions3 - test_y3))

# Visualize the model, actual and predicted data
x = 1:length(test_y3)
plot(x, test_y3, col = "red", type = "l")
lines(x, lasso_predictions3, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

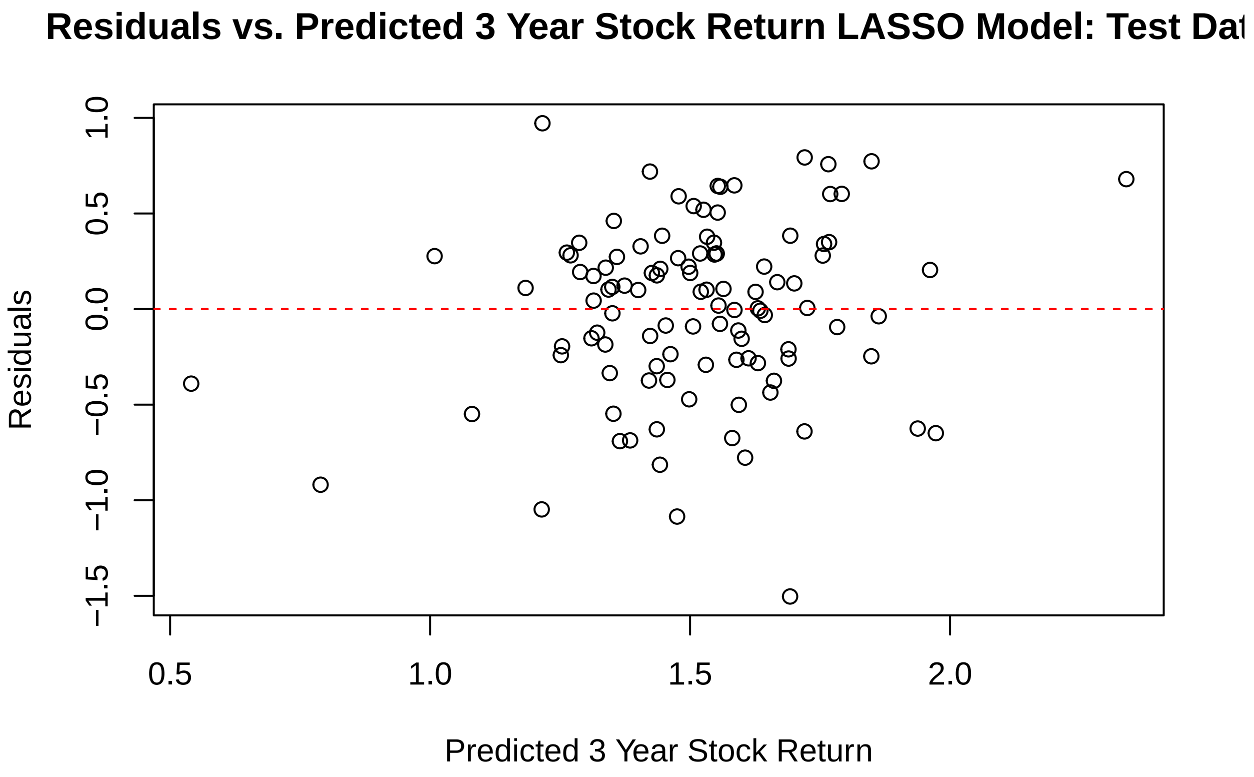


```
residuals_lasso_3 <- predictions_lasso3 - train_lm3$Stock.Return..3.Year.

# Residual plot of training data
plot(predictions_lasso3, residuals_lasso_3, main = "Residuals vs Predicted 3 Year Stock Return LASSO Moc
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```
# Create a scatter plot of residuals of test data
plot(x = lasso_predictions3, y = lasso_predictions3 - test_y3, main = "Residuals vs. Predicted 3 Year St
     xlab = "Predicted 3 Year Stock Return", ylab = "Residuals")
```

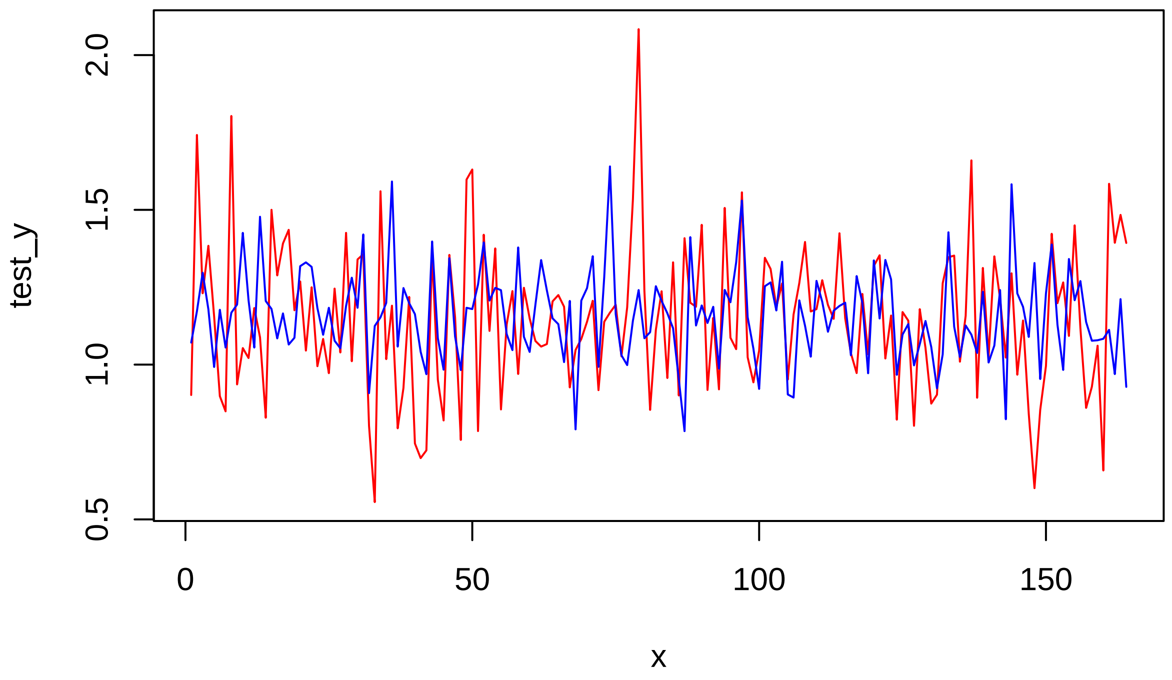
```
# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



Test random forest on two most important features generated from LASSO

```
#random forest 1 year
model_forest1 <- randomForest(Stock.Return..1.Year. ~ EBITDA3 + SP500.2.Year, data = train, proximity =

model_forest1
```

```
# MAE calculation
predictions_forest1 <- predict(model_forest1, newdata = test)

actual_values_1 <- test$Stock.Return..1.Year.

mae_forest1 <- mean(abs(predictions_forest1 - actual_values_1))

print(mae_forest1)

# Visualize the model, actual and predicted data
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l")
lines(x, predictions_forest1, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

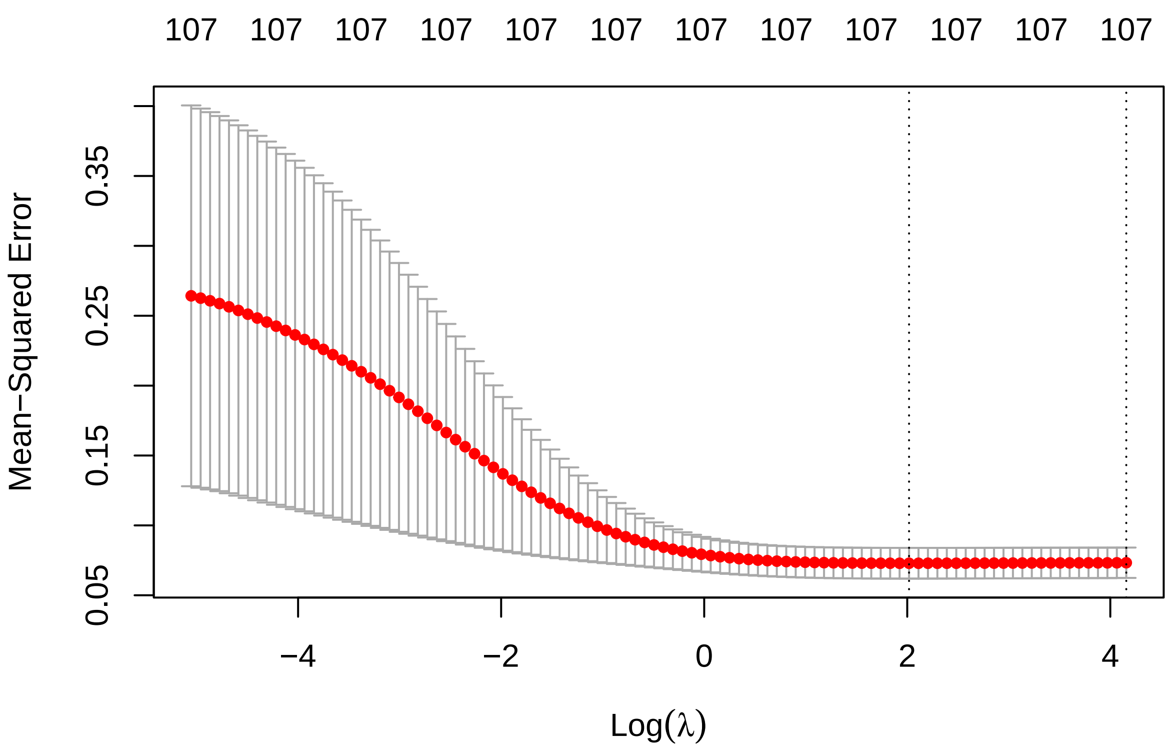


```
forest_features <- round(importance(model_forest1), 2)
```

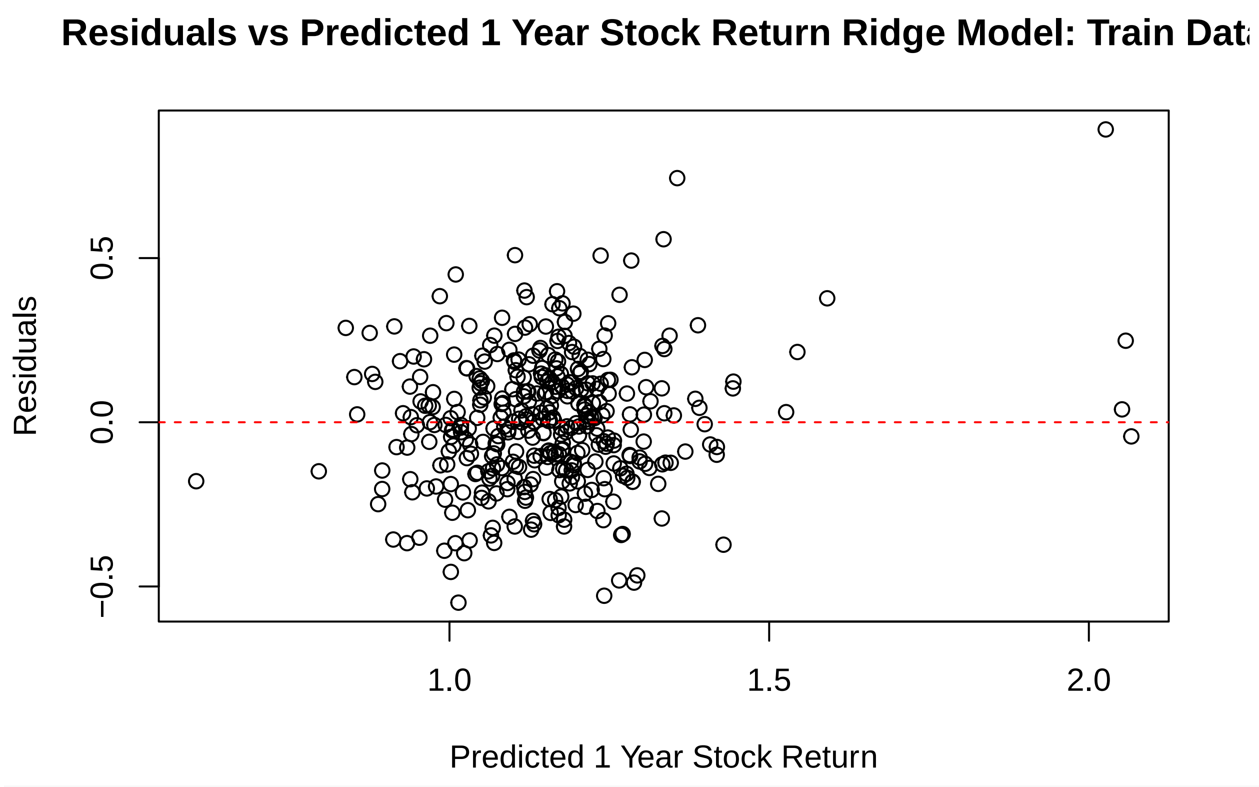
RIDGE REGRESSION

1 YEAR

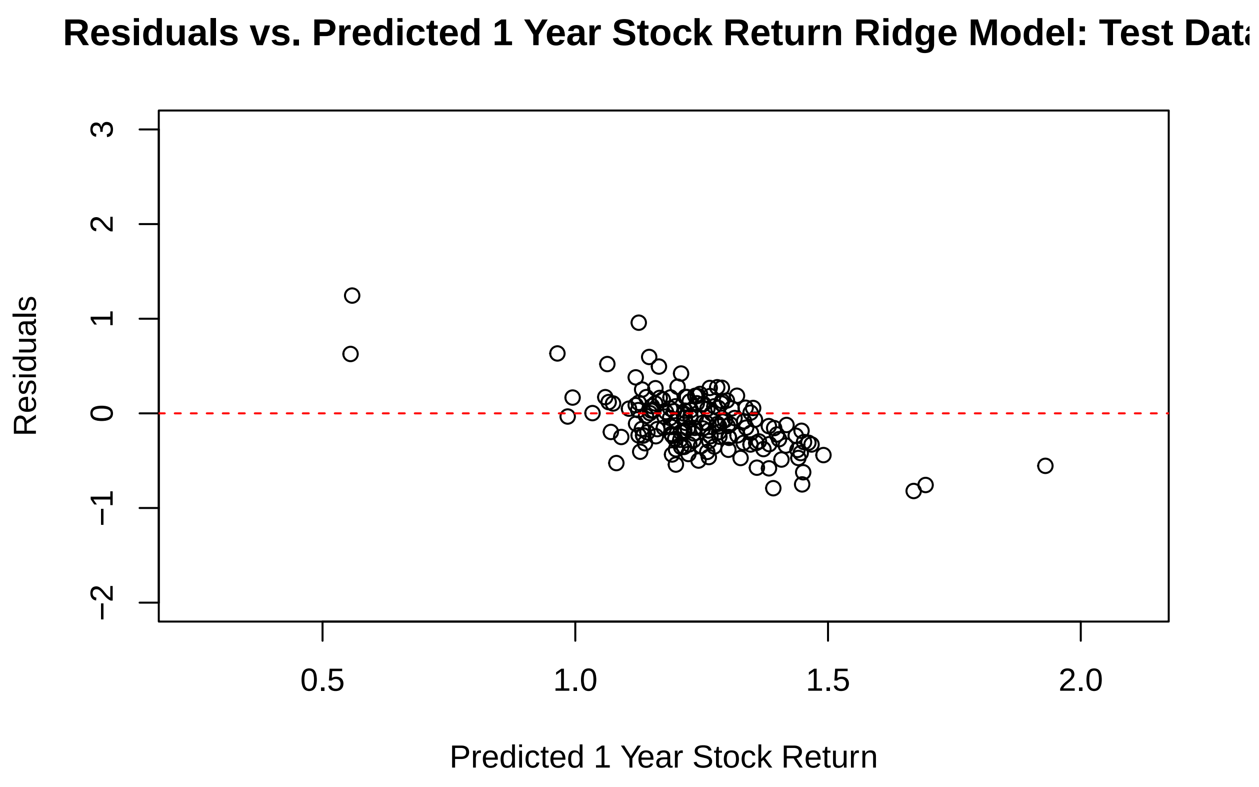
```
#RIDGE regression 1 year

#perform k-fold cross-validation to find optimal lambda value
cv_model_ridge <- cv.glmnet((independents), response, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda_ridge <- cv_model_ridge$lambda.min
best_lambda_ridge

#produce plot of test MSE by lambda value
plot(cv_model_ridge)
```



```r
#find coefficients of best model
best_model_ridge <- glmnet(independents, response, alpha = 0, lambda = best_lambda)
coef(best_model_ridge)

# Predict on the training data
predictions_ridge <- predict(best_model_ridge, type = "response", newx = independents)

coefficients_tidy_ridge <- tidy(best_model_ridge)[, c("term", "estimate")]
print(coefficients_tidy_ridge, digits = 4)

# Calculate R^2 manually for training
SS_Residual_ridge <- sum((response - predictions_ridge)^2)
mae_training_1_ridge <-  mean(abs(predictions_ridge - train_lm$Stock.Return..1.Year.))
SS_Total_ridge <- sum((response - mean(response))^2)
r_squared_ridge <- 1 - (SS_Residual_ridge / SS_Total_ridge)

# Make predictions on the testing data
ridge_predictions <- predict(best_model_ridge, newx = (independents1))

# Calculate Mean Absolute Error (MAE)
mae_ridge_test_1 <- mean(abs(ridge_predictions - test_lm$Stock.Return..1.Year.))

residuals_ridge_1 <- train_lm$Stock.Return..1.Year. - predictions_ridge

# Residuals plot on training data
plot(predictions_ridge, residuals_ridge_1, main = "Residuals vs Predicted 1 Year Stock Return Ridge Mode
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

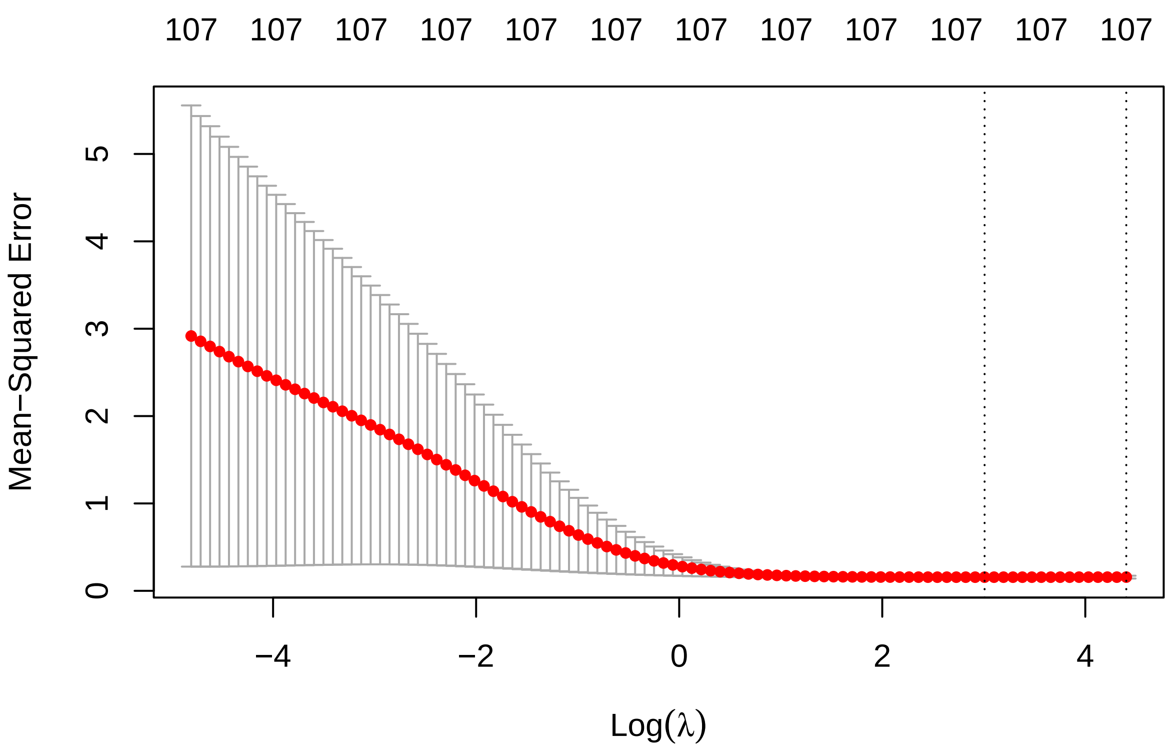


```r
# Create a scatter plot of residuals of test data
plot(x = ridge_predictions, y = test_lm$Stock.Return..1.Year. - ridge_predictions, main = "Residuals vs
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals", ylim = c(-2,3), xlim = c(0.25, 2.1))

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```

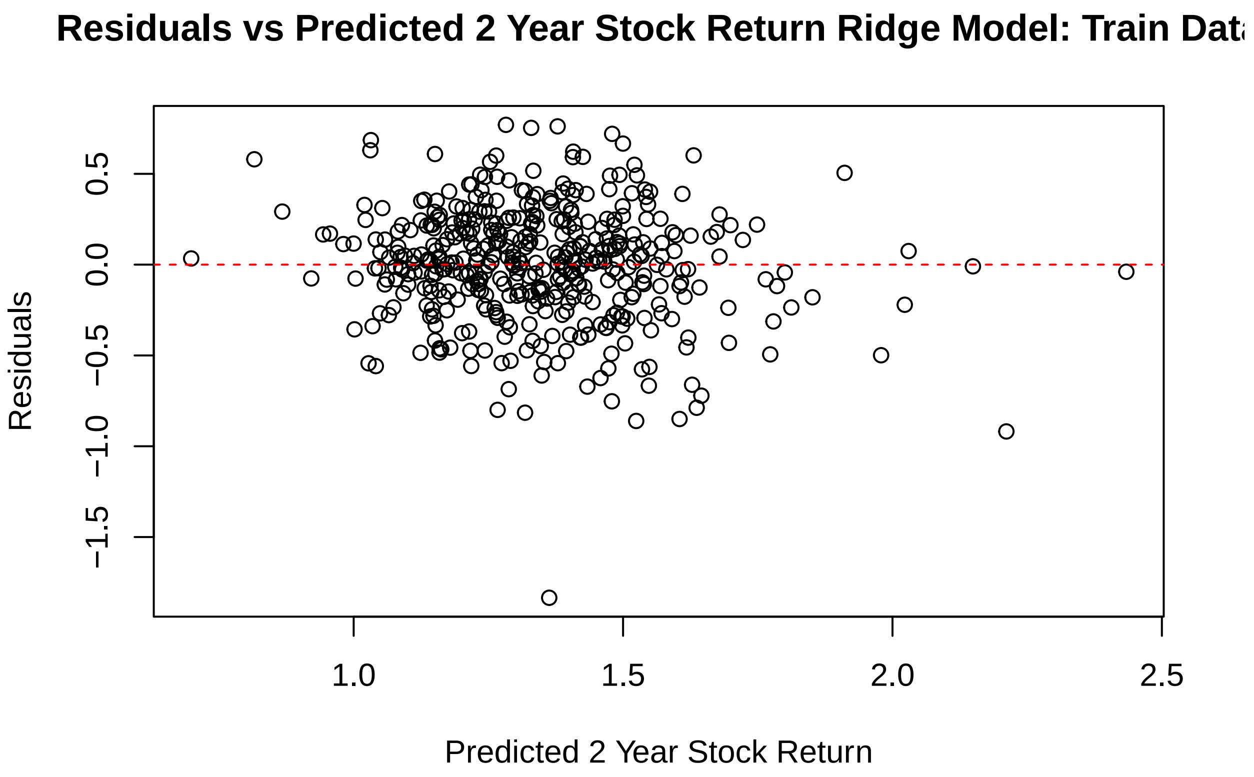


2 YEAR

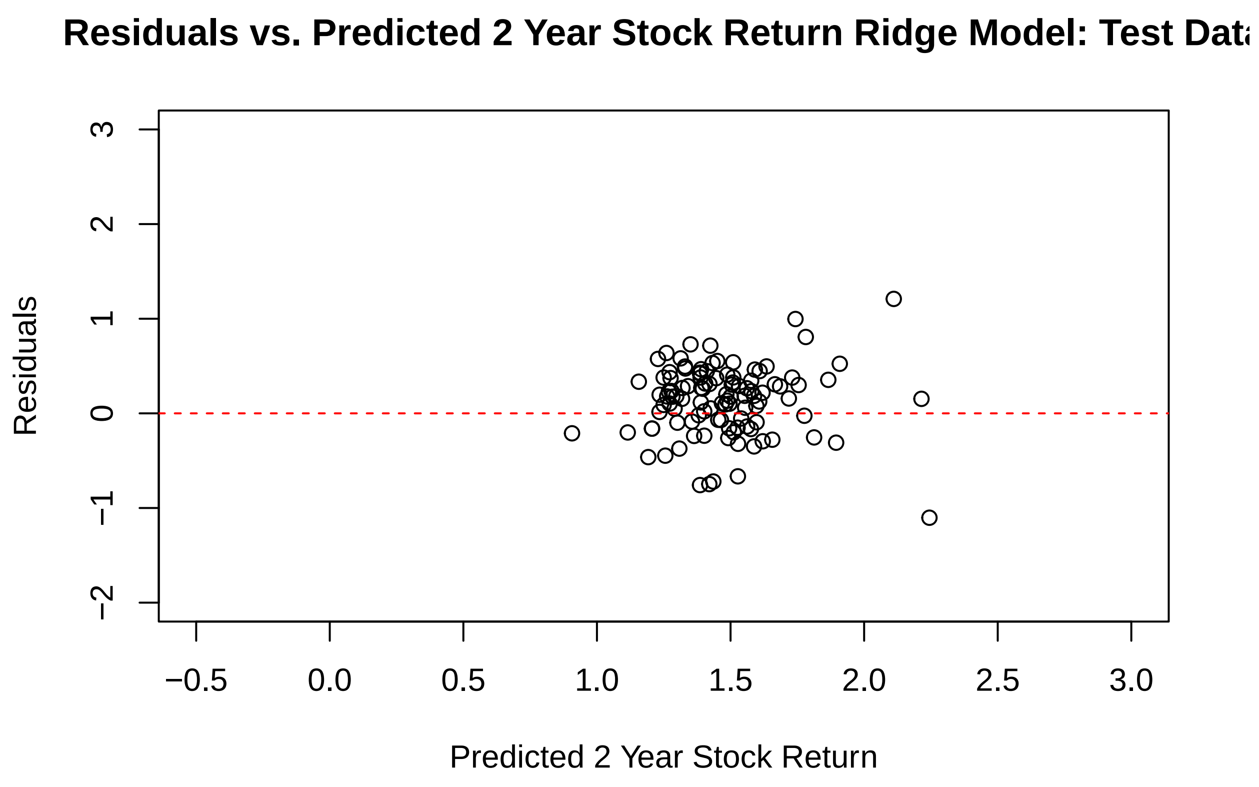
```
#RIDGE regression 2 year

#perform k-fold cross-validation to find optimal lambda value
cv_model2_ridge <- cv.glmnet(independents2, response1, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda2_ridge <- cv_model2_ridge$lambda.min
best_lambda2_ridge

#produce plot of test MSE by lambda value
plot(cv_model2_ridge)
```



```r
#find coefficients of best model
best_model2_ridge <- glmnet(independents2, response1, alpha = 0, lambda = best_lambda2)
coef(best_model2_ridge)

# Predict on the training data
predictions_ridge2 <- predict(best_model2_ridge, type = "response", newx = independents2)

coefficients_tidy2_ridge <- tidy(best_model2_ridge)[, c("term", "estimate")]
print(coefficients_tidy2_ridge, digits = 4)

# Calculate R^2 manually for training
SS_Residual2_ridge <- sum((response1 - predictions_ridge2)^2)
mae_training_2_ridge <-  mean(abs(predictions_ridge2 - train_lm2$Stock.Return..2.Year.))
SS_Total2_ridge <- sum((response1 - mean(response1))^2)
r_squared2_ridge <- 1 - (SS_Residual2_ridge / SS_Total2_ridge)

# Make predictions on the testing data
ridge_predictions2 <- predict(best_model2_ridge, newx = independents3)

# Calculate Mean Absolute Error (MAE)
mae_ridge_test_2 <- mean(abs(ridge_predictions2 - test_lm2$Stock.Return..2.Year.))

residuals_ridge_2 <- predictions_ridge2 - train_lm2$Stock.Return..2.Year.

# Residuals plot of training data
plot(predictions_ridge2, residuals_ridge_2, main = "Residuals vs Predicted 2 Year Stock Return Ridge Mo
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

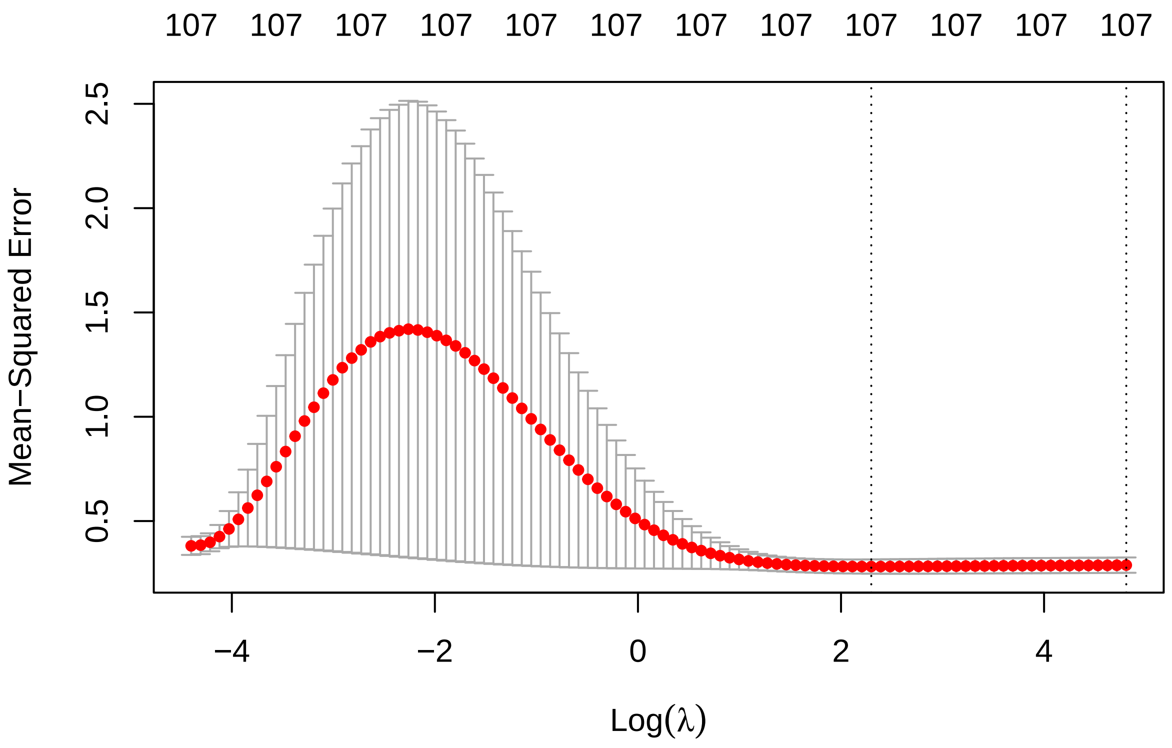


```r
# Create a scatter plot of residuals of test data
plot(x = ridge_predictions2, y = ridge_predictions2 - test_lm2$Stock.Return..2.Year., main = "Residuals
     xlab = "Predicted 2 Year Stock Return", ylab = "Residuals", ylim = c(-2,3), xlim = c(-0.5, 3))

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```

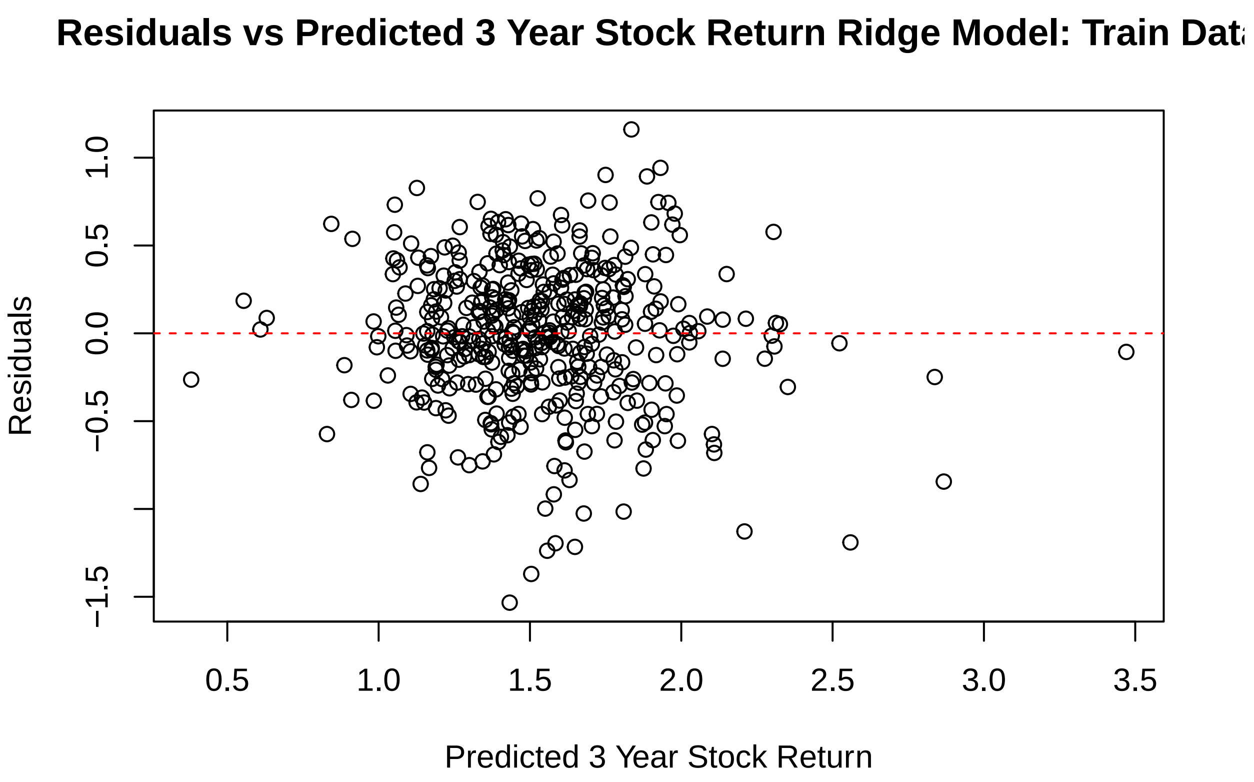


3 YEAR

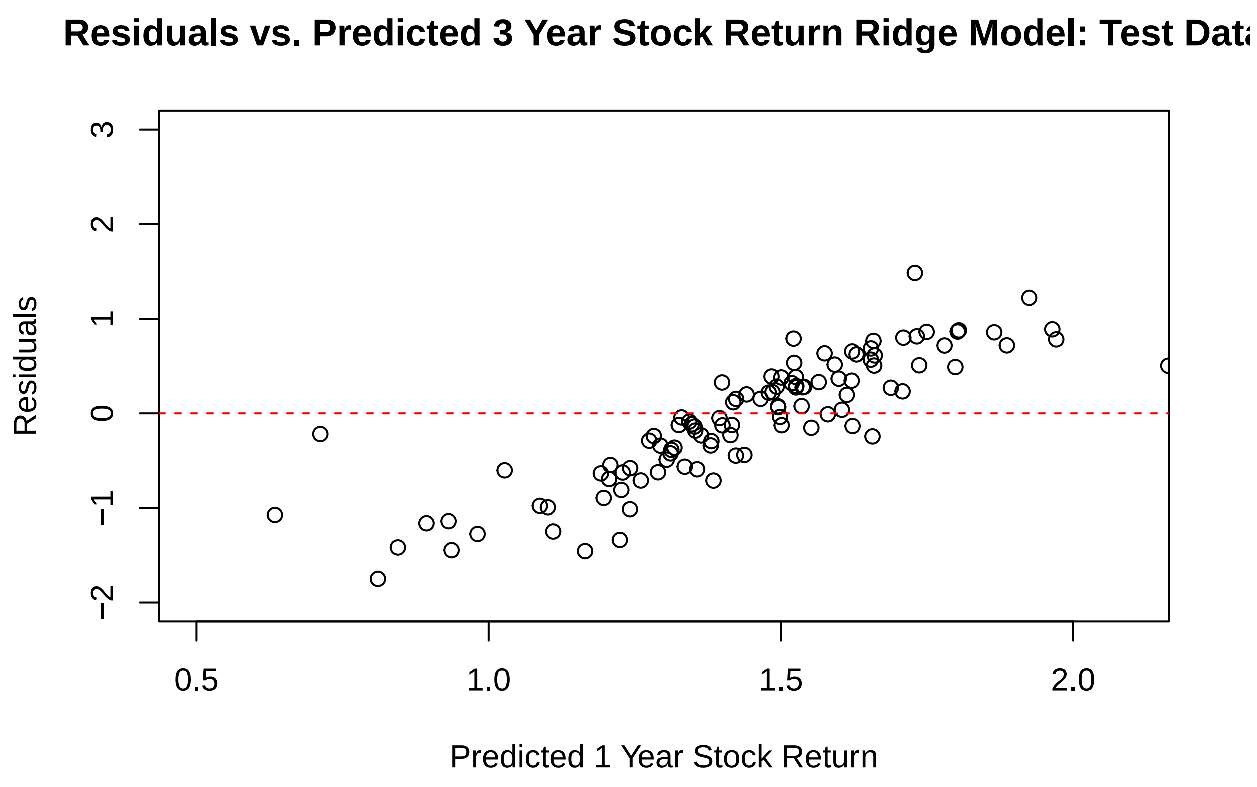
```
#RIDGE regression 3 year

#perform k-fold cross-validation to find optimal lambda value
cv_model3_ridge <- cv.glmnet(independents4, response2, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda3_ridge <- cv_model3_ridge$lambda.min
best_lambda3_ridge

#produce plot of test MSE by lambda value
plot(cv_model3_ridge)
```



```
#find coefficients of best model
best_model3_ridge <- glmnet(independents4, response2, alpha = 0, lambda = best_lambda3)
coef(best_model3_ridge)

# Predict on the training data
predictions_ridge3 <- predict(best_model3_ridge, type = "response", newx = independents4)

coefficients_tidy3_ridge <- tidy(best_model3_ridge)[, c("term", "estimate")]
print(coefficients_tidy3_ridge, digits = 4)

# Calculate R^2 manually for training
SS_Residual3_ridge <- sum((response2 - predictions_ridge3)^2)
mae_training_3_ridge <-  mean(abs(predictions_ridge3 - train_lm3$Stock.Return..3.Year.))
SS_Total3_ridge <- sum((response2 - mean(response2))^2)
r_squared3_ridge <- 1 - (SS_Residual3_ridge / SS_Total3_ridge)

# Make predictions on the testing data
ridge_predictions3 <- predict(best_model3, newx = independents5)

# Calculate Mean Absolute Error (MAE)
mae_ridge_test_3 <- mean(abs(ridge_predictions3 - test_lm3$Stock.Return..3.Year.))

residuals_ridge_3 <- predictions_ridge3 - train_lm3$Stock.Return..3.Year.

# Residuals plot for training data
plot(predictions_ridge3, residuals_ridge_3, main = "Residuals vs Predicted 3 Year Stock Return Ridge Mo
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



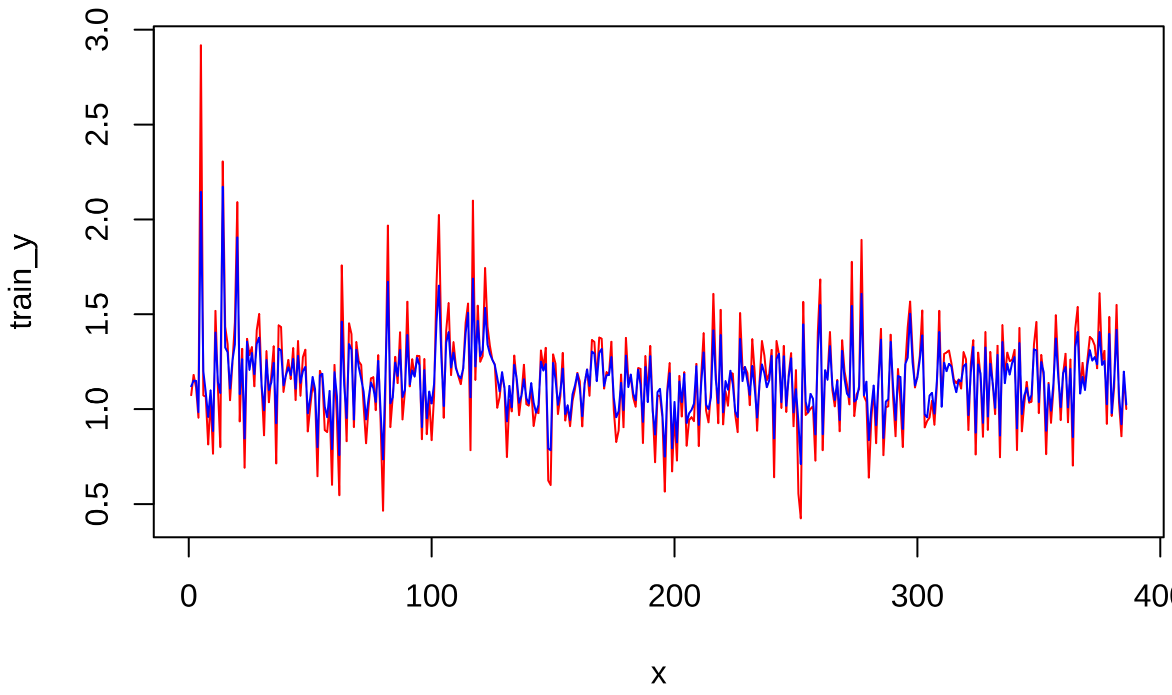
```r
# Create a scatter plot of residuals on test data
plot(x = ridge_predictions3, y = ridge_predictions3 - test_lm3$Stock.Return..3.Year., main = "Residuals
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals", ylim = c(-2,3), xlim = c(0.5, 2.1))

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



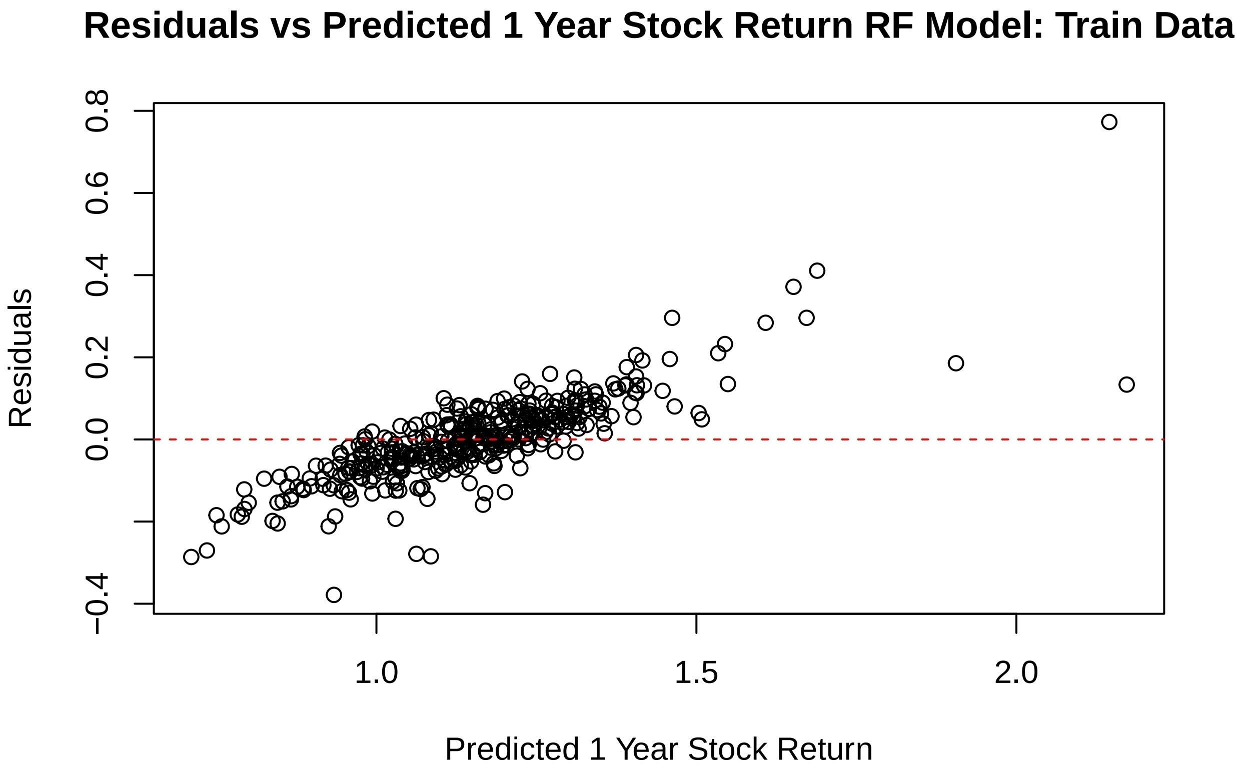
RANDOM FORESTS

1 YEAR

```
#random forest 1 year
model_forest <- randomForest(Stock.Return..1.Year. ~ . - Company - Year, data = train, proximity = TRUE

model_forest

pred_y_train_forest = predict(model_forest, train)

# Visualize the model, actual and predicted data
x = 1:length(train_y)
plot(x, train_y, col = "red", type = "l")
lines(x, pred_y_train_forest, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

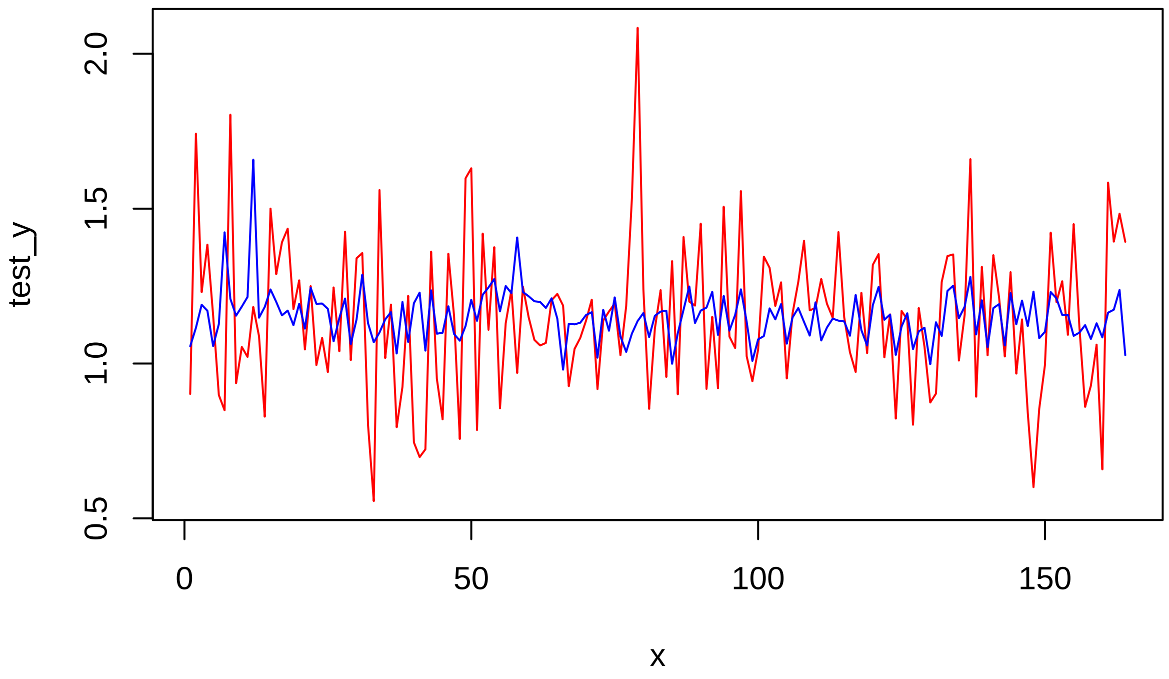


```r
residuals_rf_1 <- train_y - pred_y_train_forest

# Residuals plot of training data
plot(pred_y_train_forest, residuals_rf_1, main = "Residuals vs Predicted 1 Year Stock Return RF Model: '
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

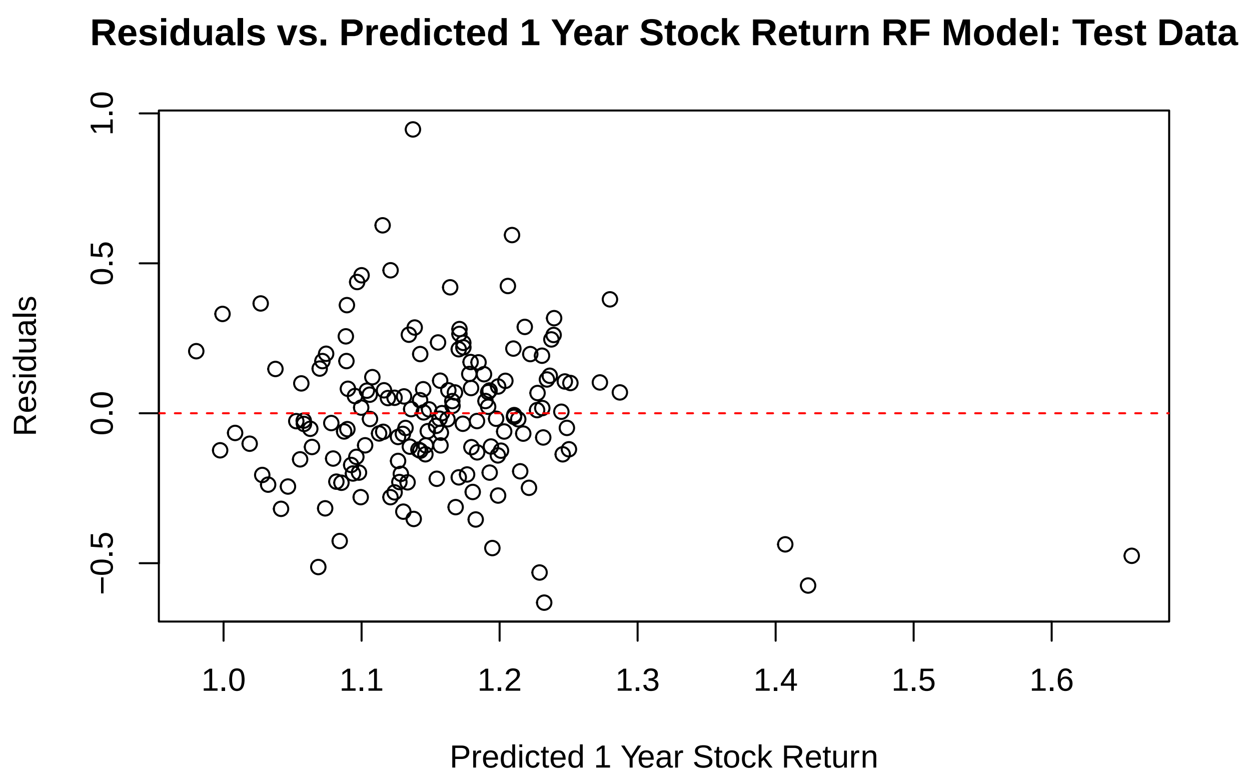


```r
# MAE calculations
predictions_forest1 <- predict(model_forest, newdata = test)
```

```r
actual_values_1 <- test$Stock.Return..1.Year.

mae_forest1 <- mean(abs(predictions_forest1 - actual_values_1))

print(mae_forest1)

# Visualize the model, actual and predicted data
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l")
lines(x, predictions_forest1, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```r
# Create a scatter plot of residuals of test data
plot(x = predictions_forest1, y = actual_values_1 - predictions_forest1, main = "Residuals vs. Predicte
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```

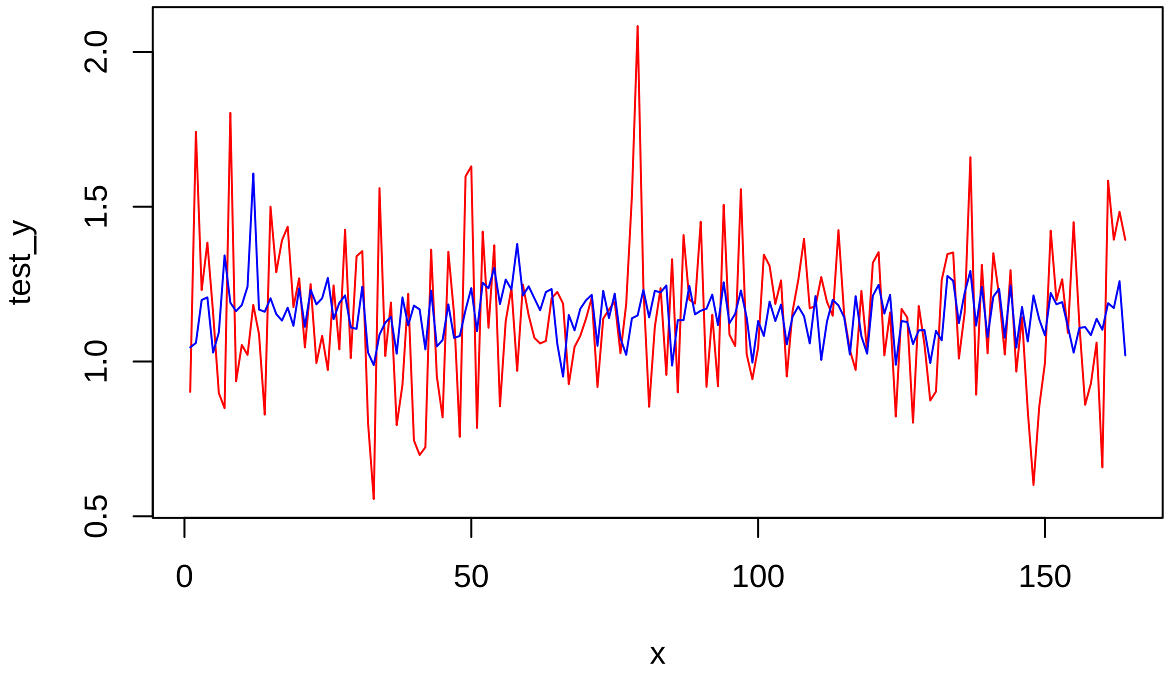


```
forest_features <- round(importance(model_forest), 2)
```

Example random forest of top features

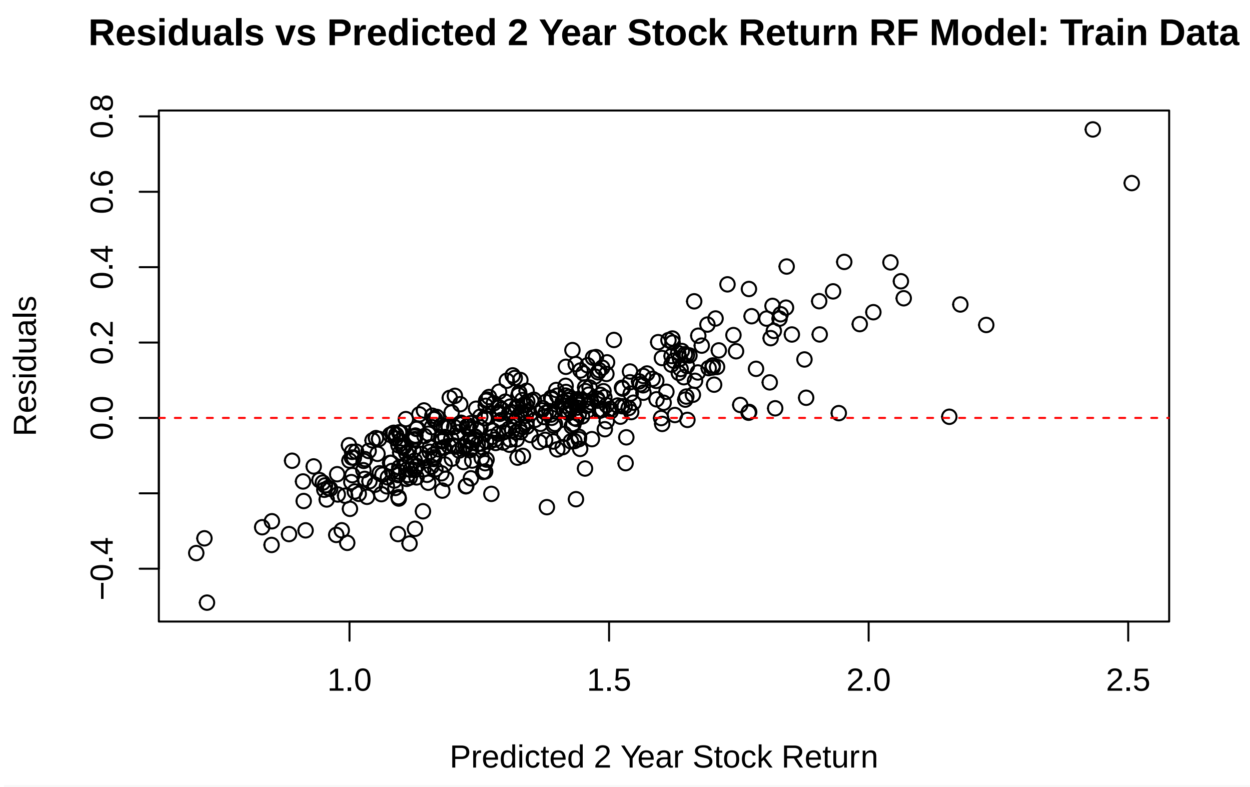
```
top_features <- order(forest_features, decreasing = TRUE)[1:30]  # Select top 30 features

# Extract the names of the top features
selected_features <- rownames(forest_features)[top_features]

# Train New Random Forest with Selected Features
model_selected_features <- randomForest(
  Stock.Return..1.Year. ~ .,
  data = train[, c(selected_features, "Stock.Return..1.Year.")],
  ntree = 1000,
  mtry = 10,
  nodesize = 2
)

model_selected_features

# MAE calculations
predictions_forest1_top <- predict(model_selected_features, newdata = test)

actual_values_1_top <- test$Stock.Return..1.Year.

mae_forest1_top <- mean(abs(predictions_forest1_top - actual_values_1_top))

print(mae_forest1_top)
```

```
# Visualize the model, actual and predicted data
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l")
lines(x, predictions_forest1_top, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

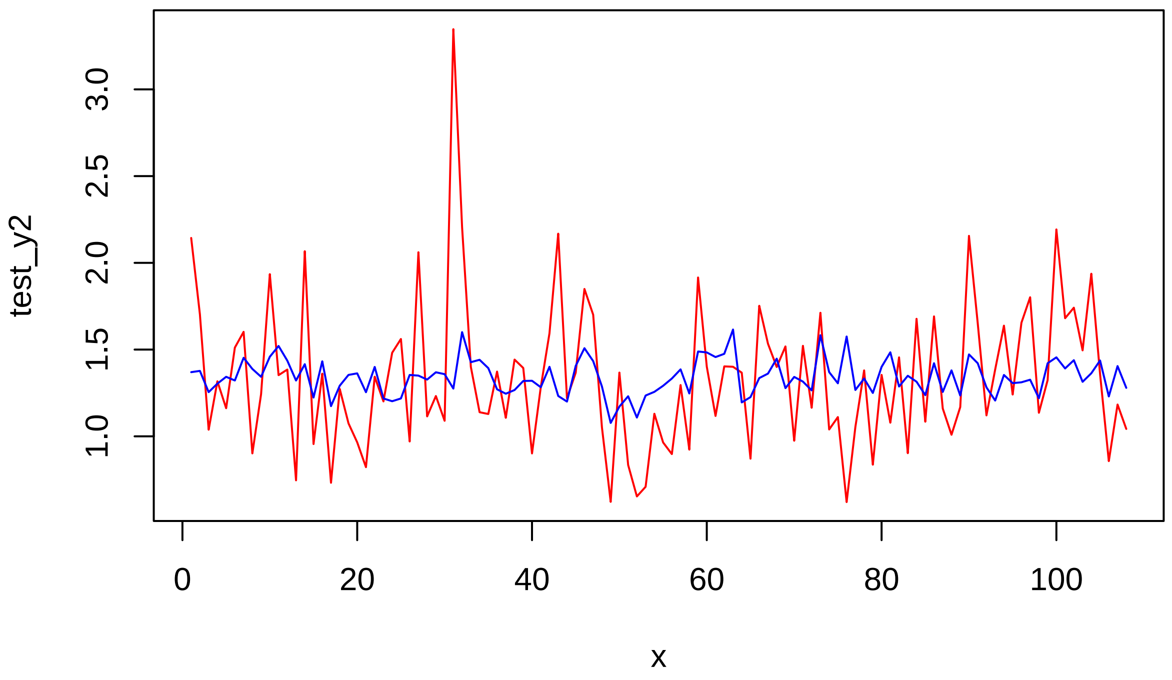


2 YEAR

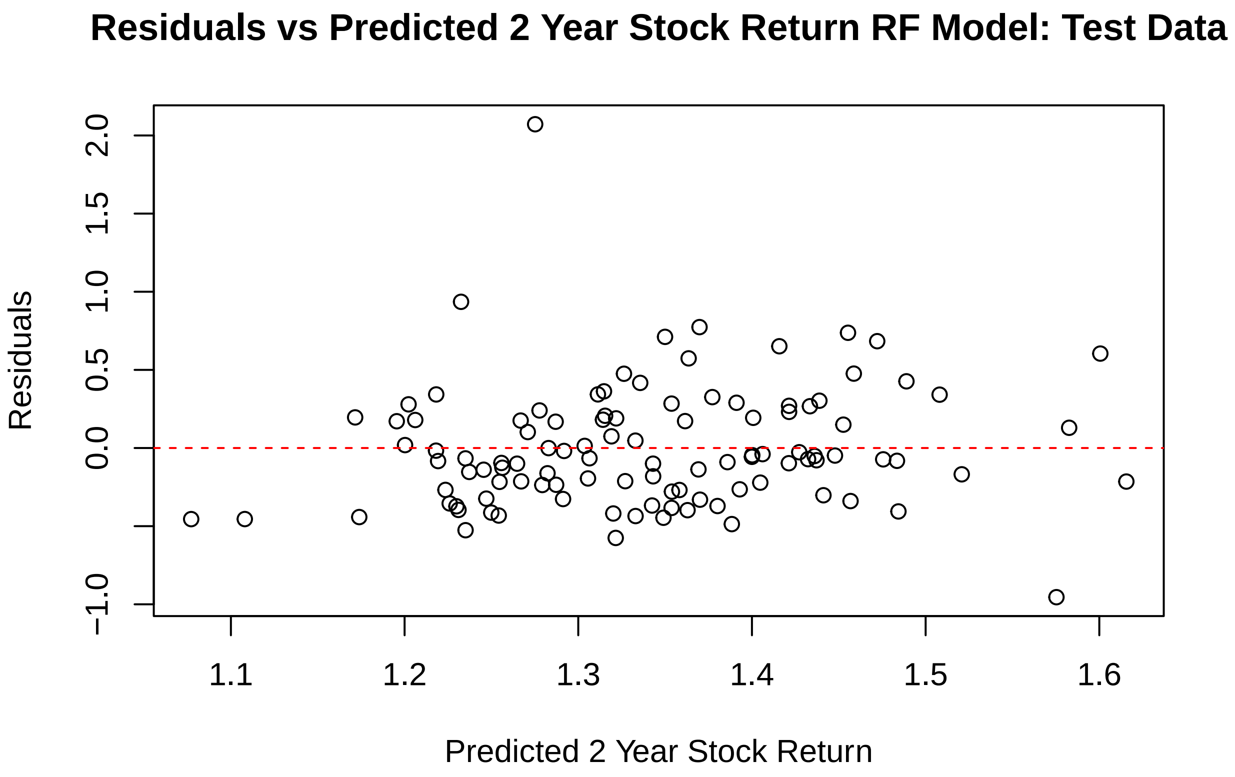
```
#random forest 2 year
model_forest2 <- randomForest(Stock.Return..2.Year. ~ . - Company - Year, data = train2, proximity = TRU

model_forest2

pred_y_train_forest2 = predict(model_forest2, train2)

residuals_rf_2 <- train_y2 - pred_y_train_forest2

# Residuals plot of training data
plot(pred_y_train_forest2, residuals_rf_2, main = "Residuals vs Predicted 2 Year Stock Return RF Model:
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```r
# MAE calculations
predictions_forest2 <- predict(model_forest2, newdata = test2)

actual_values_2 <- test2$Stock.Return..2.Year.

mae_forest2 <- mean(abs(predictions_forest2 - actual_values_2))

print(mae_forest2)

# Visualize the model, actual and predicted data
x = 1:length(test_y2)
plot(x, test_y2, col = "red", type = "l")
lines(x, predictions_forest2, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



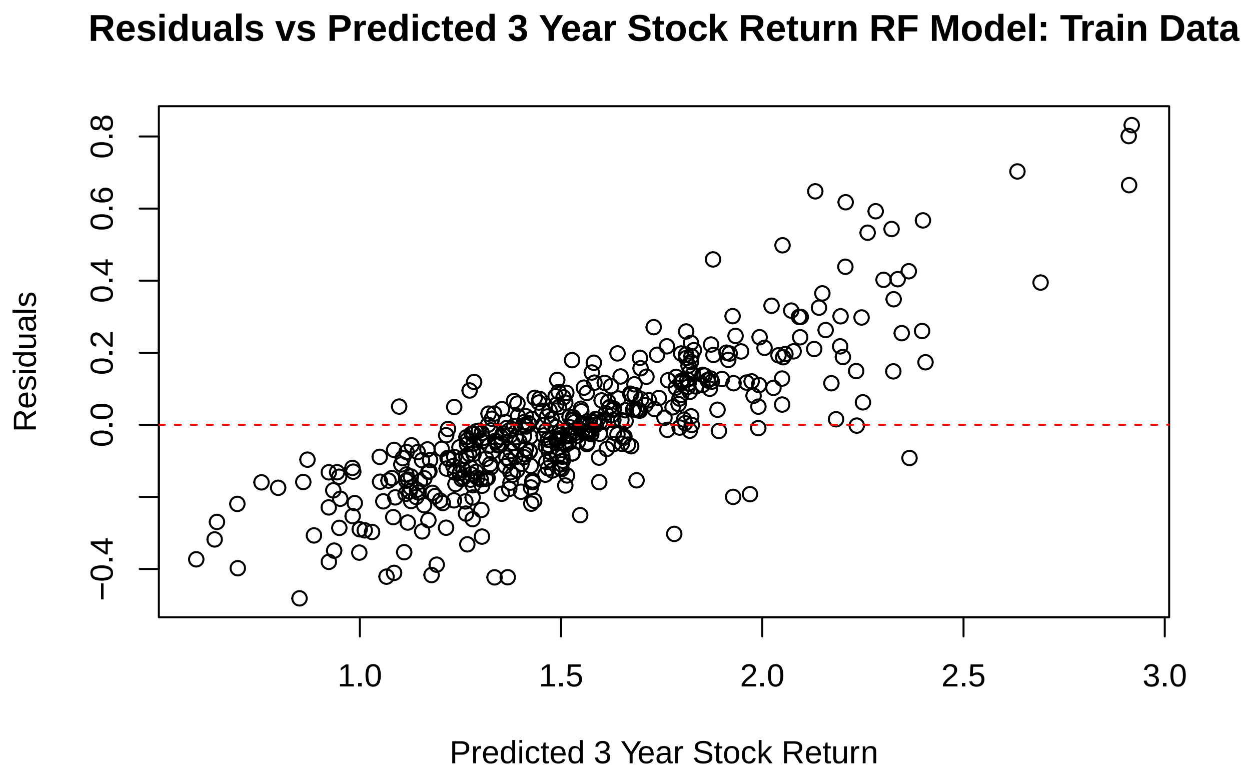
```
# Residuals plot of test data
plot(predictions_forest2, actual_values_2 - predictions_forest2, main = "Residuals vs Predicted 2 Year S
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



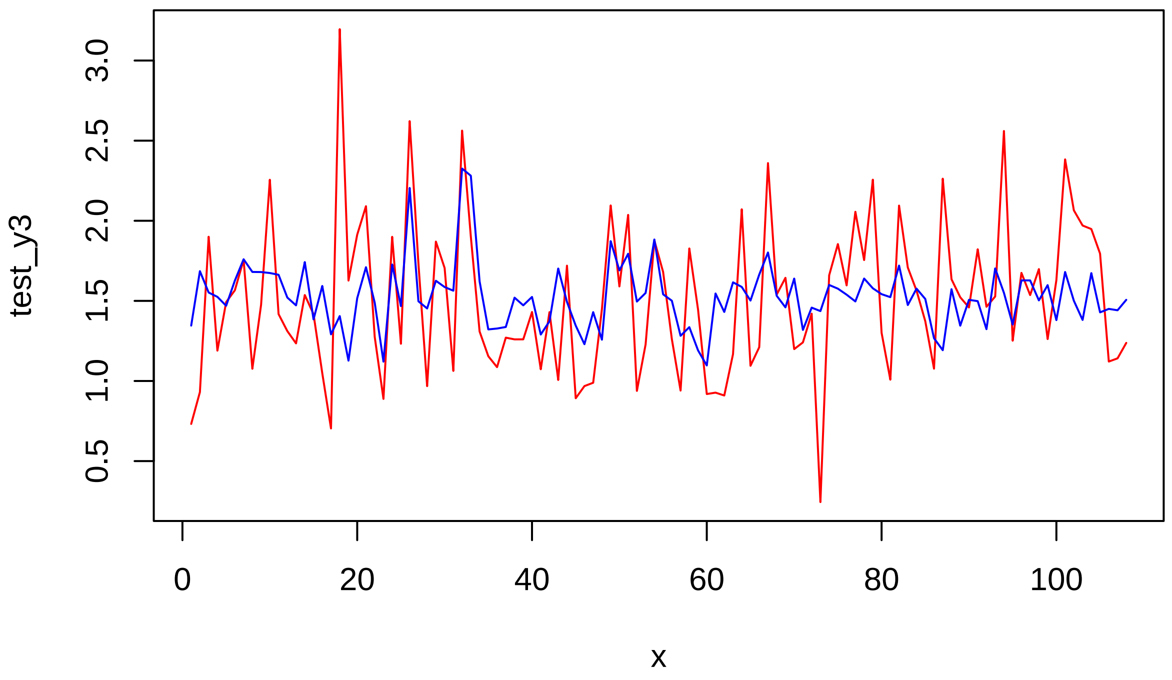
```
forest_features2 <- round(importance(model_forest2), 2)
```

3 YEAR

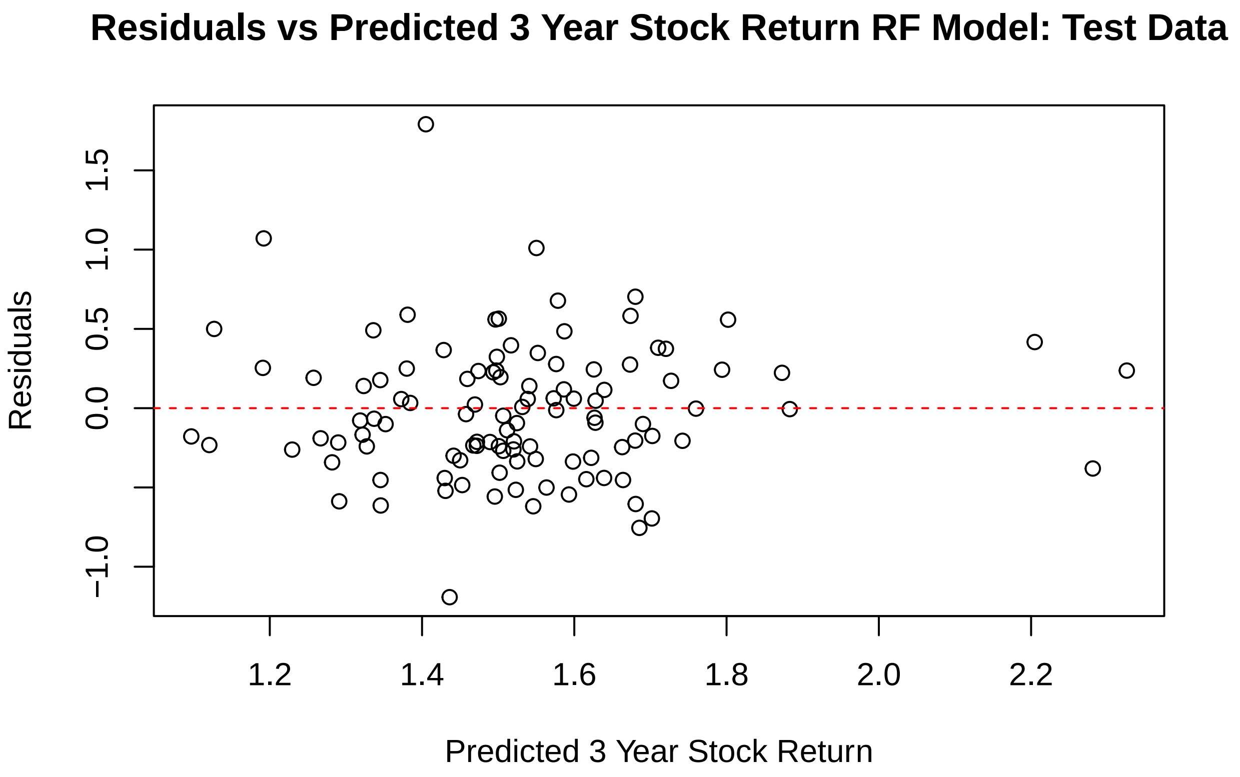
```
#random forest 3 year
model_forest3 <- randomForest(Stock.Return..3.Year. ~ . - Company - Year, data = train3, proximity = TRU

model_forest3

pred_y_train_forest3 = predict(model_forest3, train3)

residuals_rf_3 <- train_y3 - pred_y_train_forest3

# Residuals plot of training data
plot(pred_y_train_forest3, residuals_rf_3, main = "Residuals vs Predicted 3 Year Stock Return RF Model:
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```
# MAE calculations
predictions_forest3 <- predict(model_forest3, newdata = test3)

actual_values_3 <- test3$Stock.Return..3.Year.

mae_forest3 <- mean(abs(predictions_forest3 - actual_values_3))

print(mae_forest3)

# Visualize the model, actual and predicted data
x = 1:length(test_y3)
plot(x, test_y3, col = "red", type = "l")
lines(x, predictions_forest3, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```
# Residuals plot of test data
plot(predictions_forest3, actual_values_3 - predictions_forest3, main = "Residuals vs Predicted 3 Year S
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

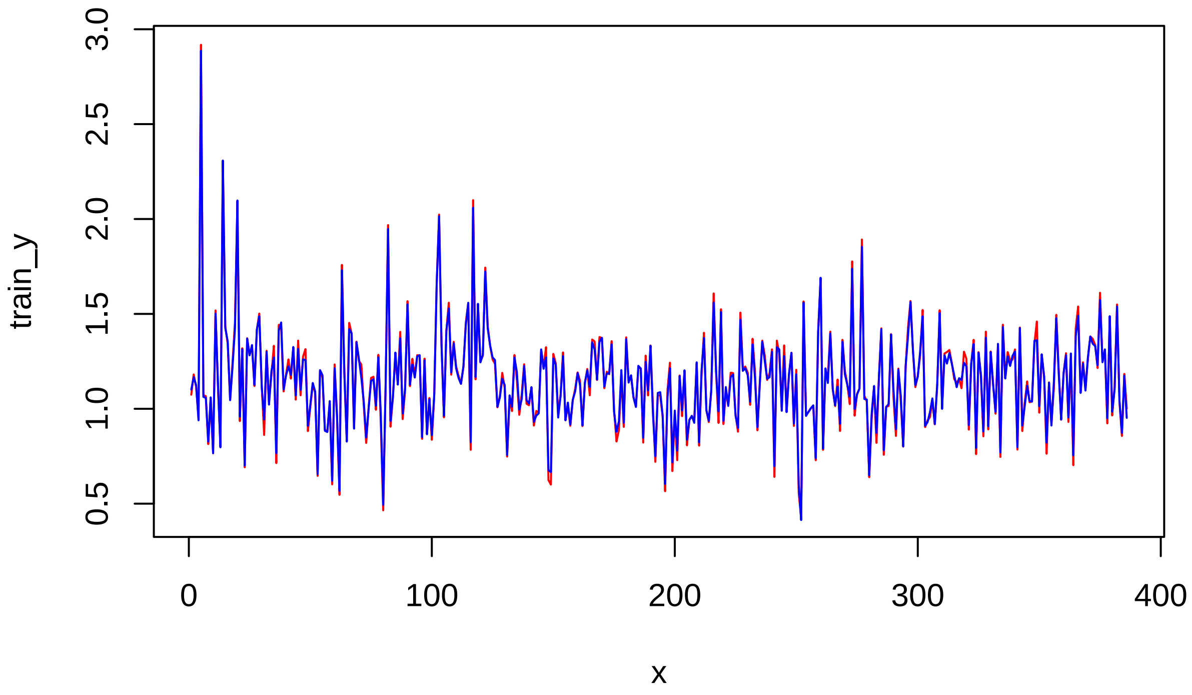


```
forest_features3 <- round(importance(model_forest3), 2)
```

GRADIENT BOOSTING

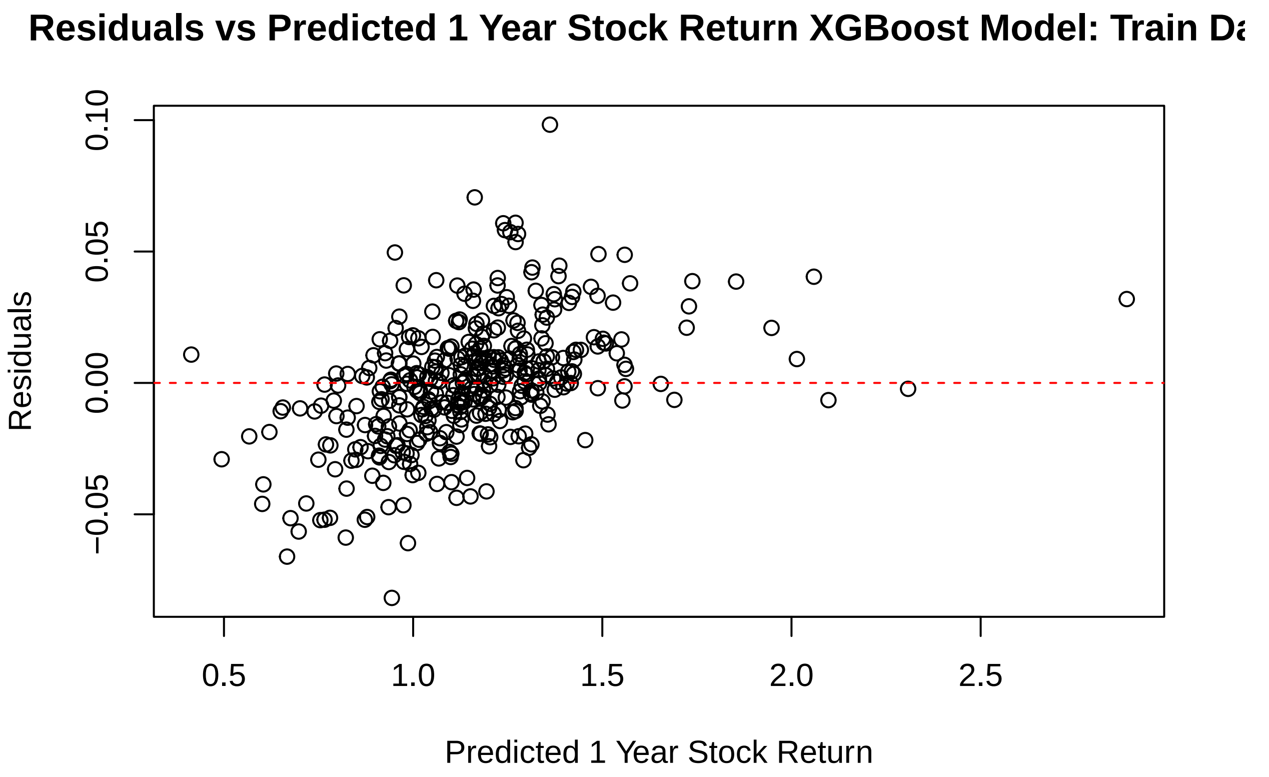
1 YEAR

```r
#gradient boosting year 1
set.seed(100)

#define final training and testing sets
xgb_train = xgb.DMatrix(data = train_x, label = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)

#defining a watchlist
watchlist = list(train=xgb_train, test=xgb_test)

#fit XGBoost model and display training and testing data at each iteartion
model = xgb.train(data = xgb_train, max.depth = 3, watchlist=watchlist, nrounds = 9)

#define final model
model_xgboost = xgboost(data = xgb_train, max.depth = 3, nrounds = 86, verbose = 0)

summary(model_xgboost)

pred_y_train = predict(model_xgboost, xgb_train)

# Visualize the model, actual and predicted data
x = 1:length(train_y)
plot(x, train_y, col = "red", type = "l")
lines(x, pred_y_train, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



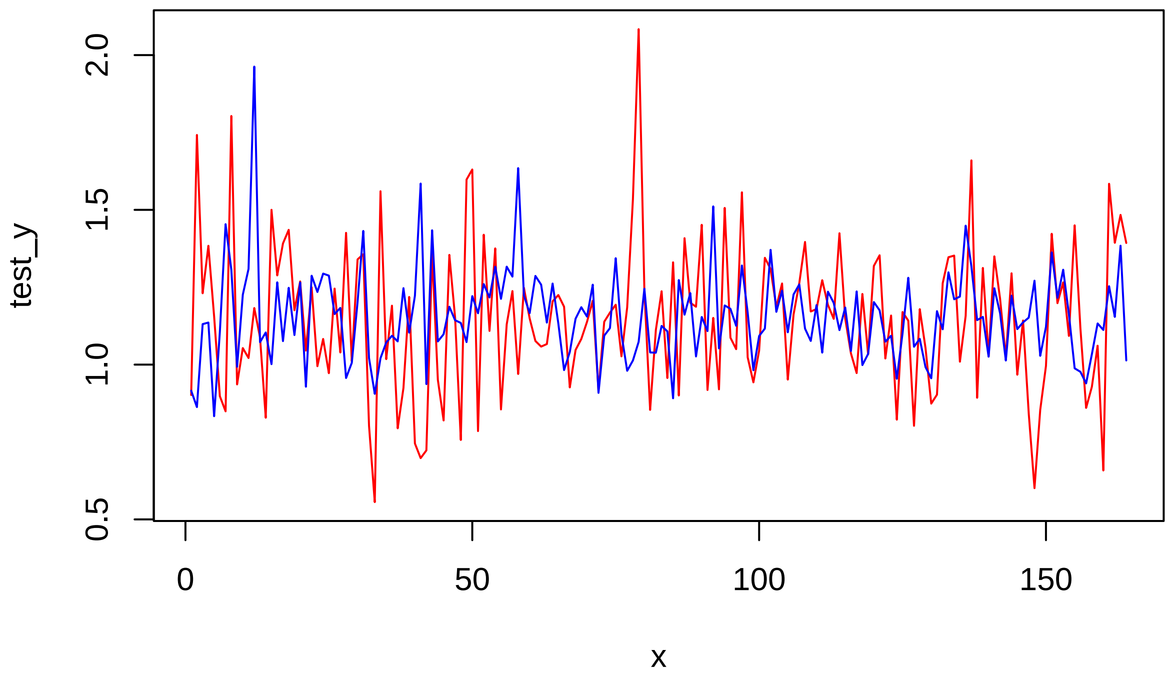
```r
residuals_xg_1 <- train_y - pred_y_train
```

```
# Residuals plot of train data
plot(pred_y_train, residuals_xg_1, main = "Residuals vs Predicted 1 Year Stock Return XGBoost Model: Tra
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

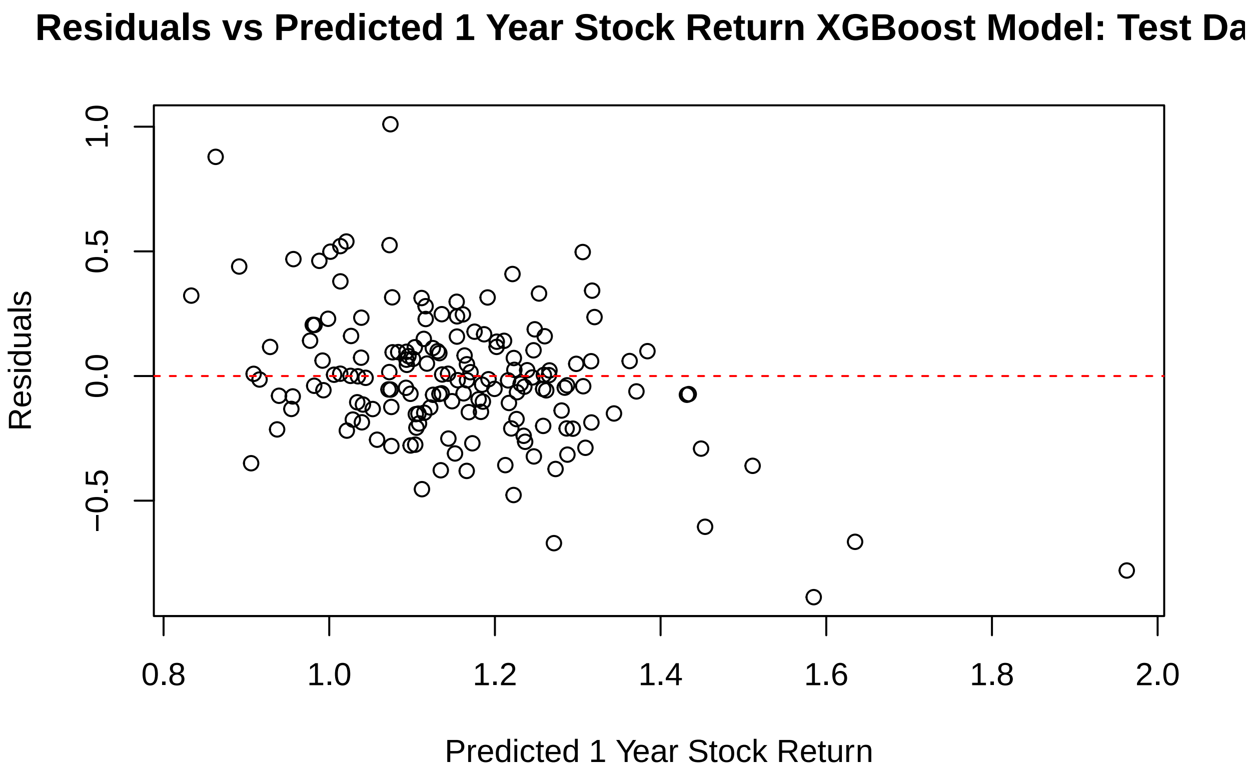


```
pred_y = predict(model_xgboost, xgb_test)
```

```
# Calculations of model metrics
mean((test_y - pred_y)^2) #mse - Mean Squared Error

mae_grad1 <- mean(abs(test_y - pred_y))

caret::RMSE(test_y, pred_y) #rmse - Root Mean Squared Error

y_test_mean = mean(test_y)
# Calculate total sum of squares
tss =  sum((test_y - y_test_mean)^2 )
# Calculate residual sum of squares
rss =  sum((test_y - pred_y)^2)
# Calculate R-squared
rsq_xgb1  =  1 - (rss/tss)
cat('The R-square of the test data is ', round(rsq_xgb1,3), '\n')

# Visualize the model, actual and predicted data
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l")
lines(x, pred_y, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



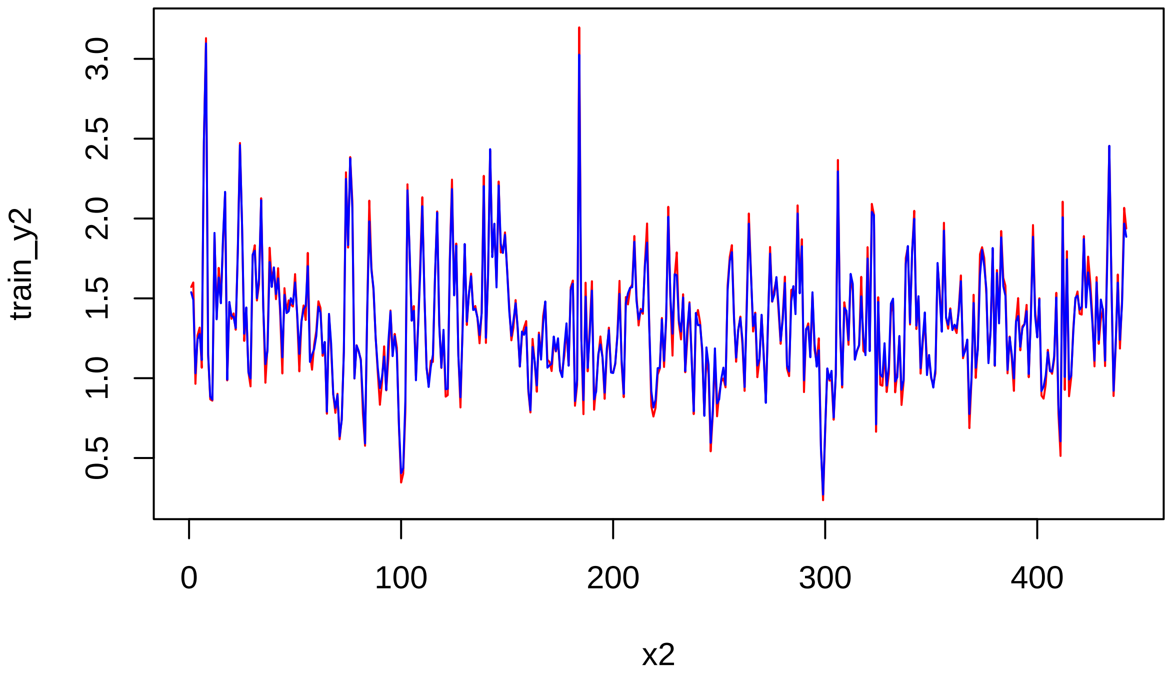
```
# Residuals plot of test data
plot(pred_y, test_y - pred_y, main = "Residuals vs Predicted 1 Year Stock Return XGBoost Model: Test Da
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



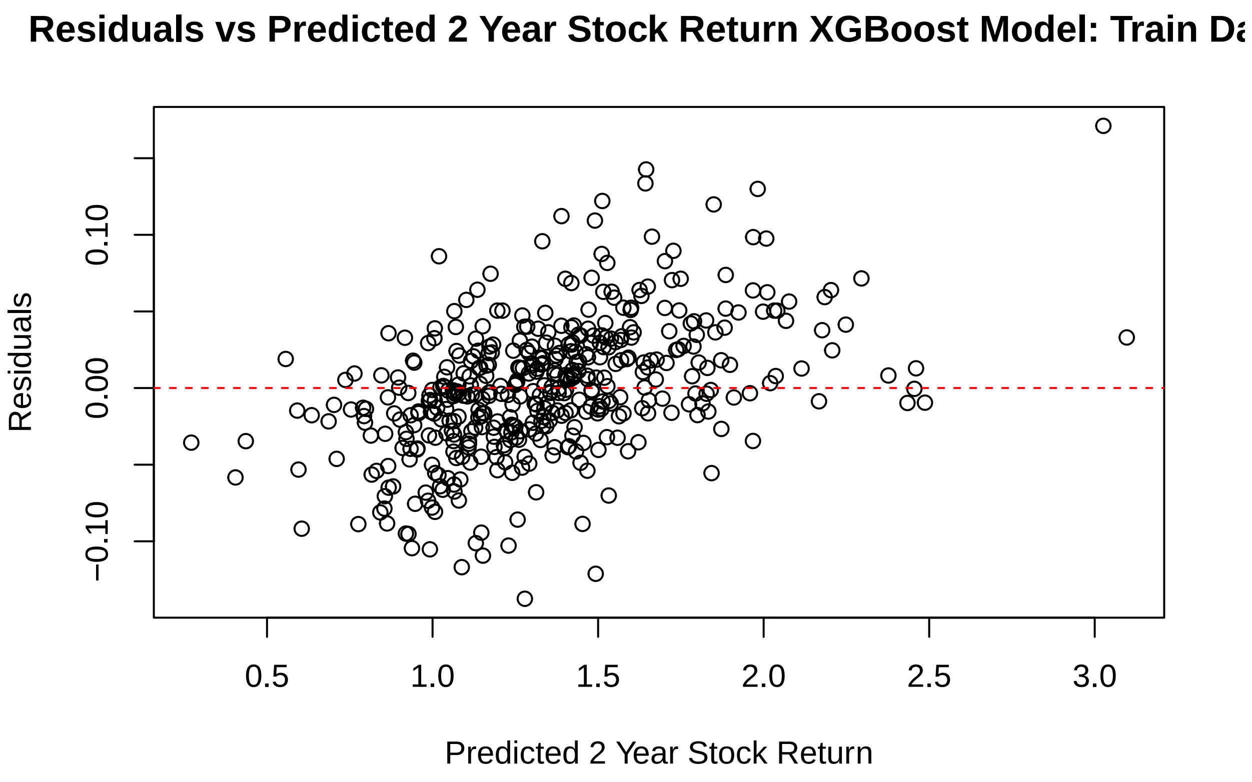
```
importance_matrix_gradient <- xgb.importance(model = model_xgboost)
```

2 YEAR

```
#gradient boosting year 2
set.seed(150)

#define final training and testing sets
xgb_train2 = xgb.DMatrix(data = train_x2, label = train_y2)
xgb_test2 = xgb.DMatrix(data = test_x2, label = test_y2)

#defining a watchlist
watchlist2 = list(train=xgb_train2, test=xgb_test2)

#fit XGBoost model and display training and testing data at each iteration
model2 = xgb.train(data = xgb_train2, max.depth = 1, watchlist=watchlist2, nrounds = 35)

#define final model
model_xgboost2 = xgboost(data = xgb_train2, max.depth = 3, nrounds = 86, verbose = 0,
  eval_metric = "mae",
  objective = "reg:squarederror",
  min_child_weight = 5)

summary(model_xgboost2)

pred_y_train2 = predict(model_xgboost2, xgb_train2)

# Visualize the model, actual and predicted data
x2 = 1:length(train_y2)
plot(x2, train_y2, col = "red", type = "l")
lines(x2, pred_y_train2, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

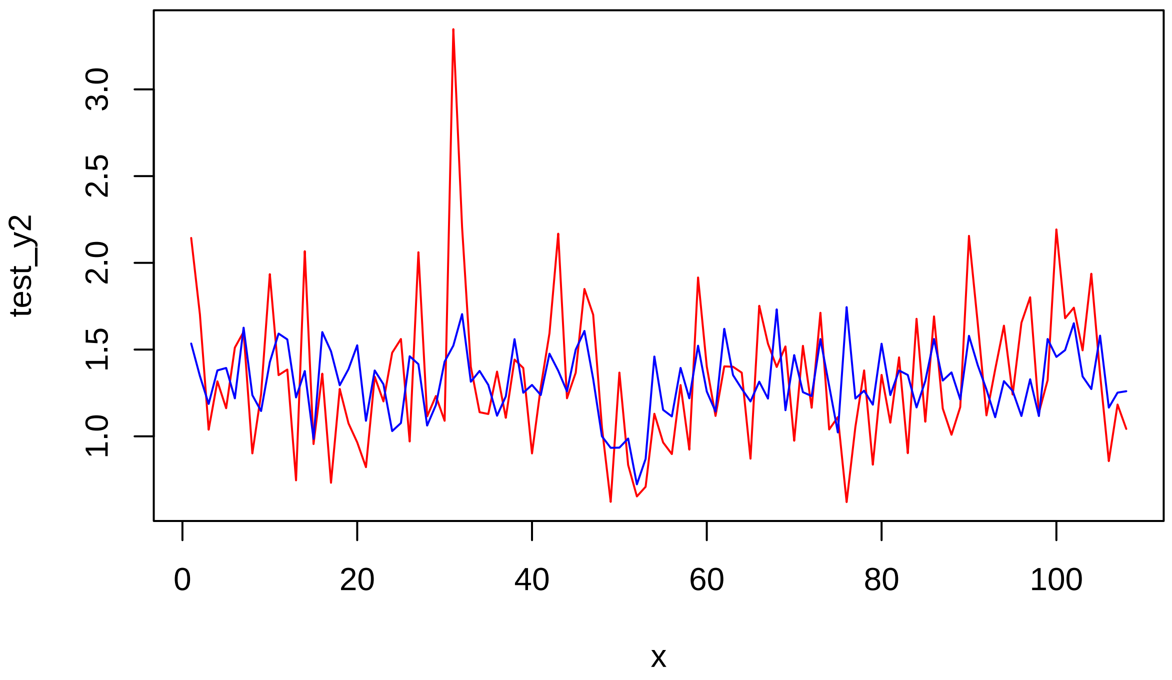


```r
residuals_xg_2 <- train_y2 - pred_y_train2

# Residuals plot of train data
plot(pred_y_train2, residuals_xg_2, main = "Residuals vs Predicted 2 Year Stock Return XGBoost Model: T
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

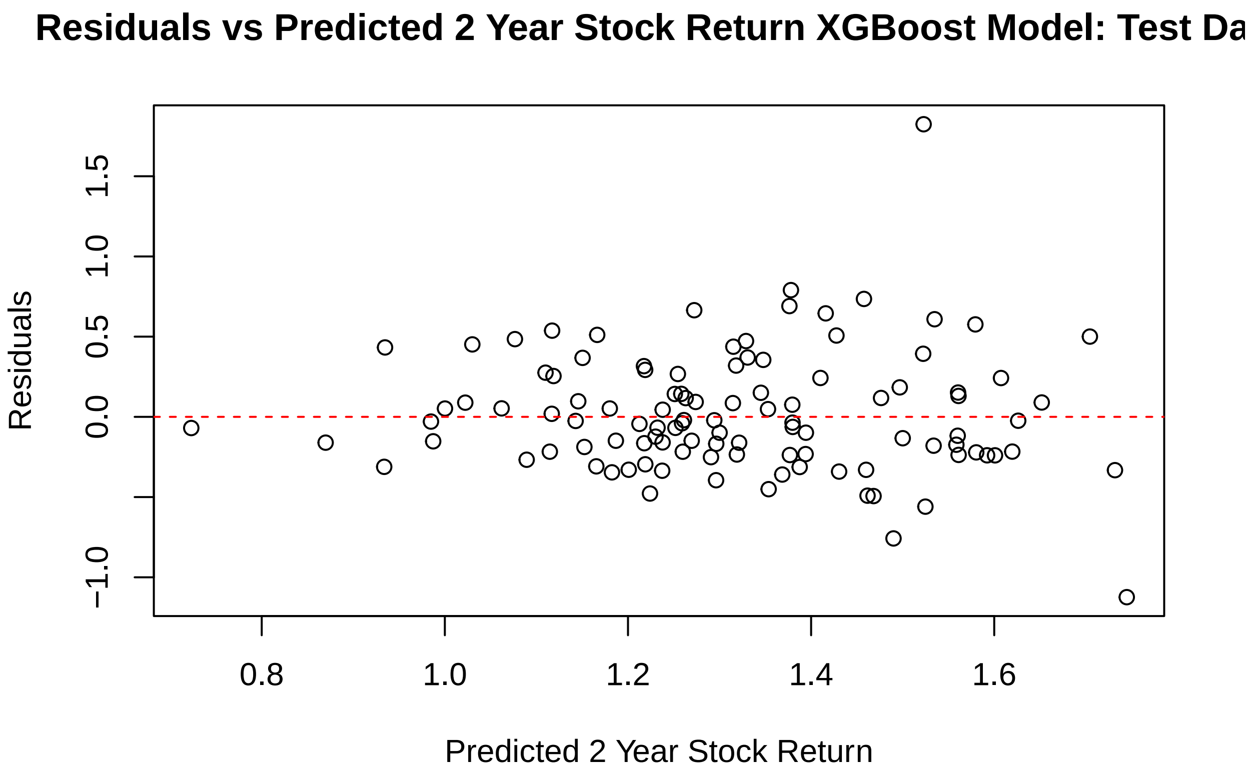


```r
pred_y2 = predict(model_xgboost2, xgb_test2)
```

```r
# Model metrics calculation
mean((test_y2 - pred_y2)^2) #mse - Mean Squared Error

mae_grad2 <- mean(abs(test_y2 - pred_y2))

caret::RMSE(test_y2, pred_y2) #rmse - Root Mean Squared Error

y_test_mean2 = mean(test_y2)
# Calculate total sum of squares
tss2 =  sum((test_y2 - y_test_mean2)^2 )
# Calculate residual sum of squares
rss2 =  sum((test_y2 - pred_y2)^2)
# Calculate R-squared
rsq_xgb2  =  1 - (rss2/tss2)
cat('The R-square of the test data is ', round(rsq_xgb2,3), '\n')

# Visualize the model, actual and predicted data
x = 1:length(test_y2)
plot(x, test_y2, col = "red", type = "l")
lines(x, pred_y2, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



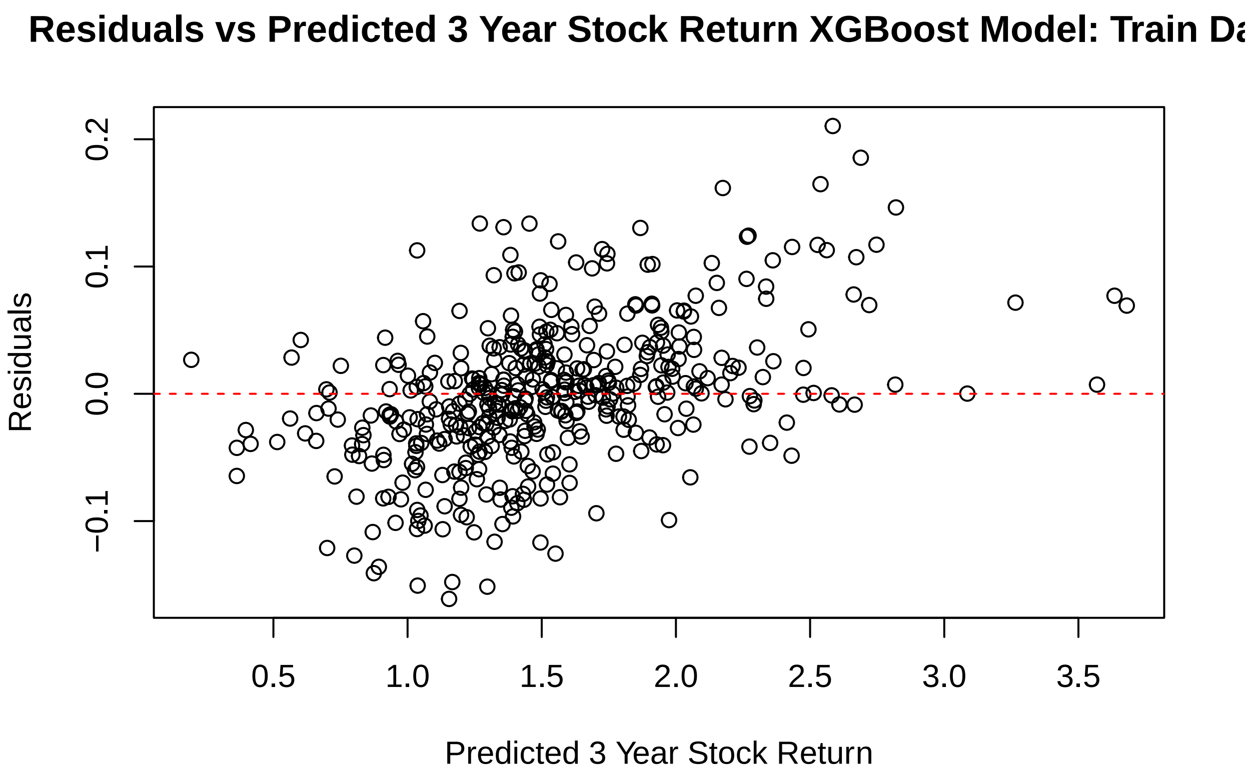
```
# Residuals plot of test data
plot(pred_y2, test_y2 - pred_y2, main = "Residuals vs Predicted 2 Year Stock Return XGBoost Model: Test
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```
importance_matrix_gradient2 <- xgb.importance(model = model_xgboost2)
```

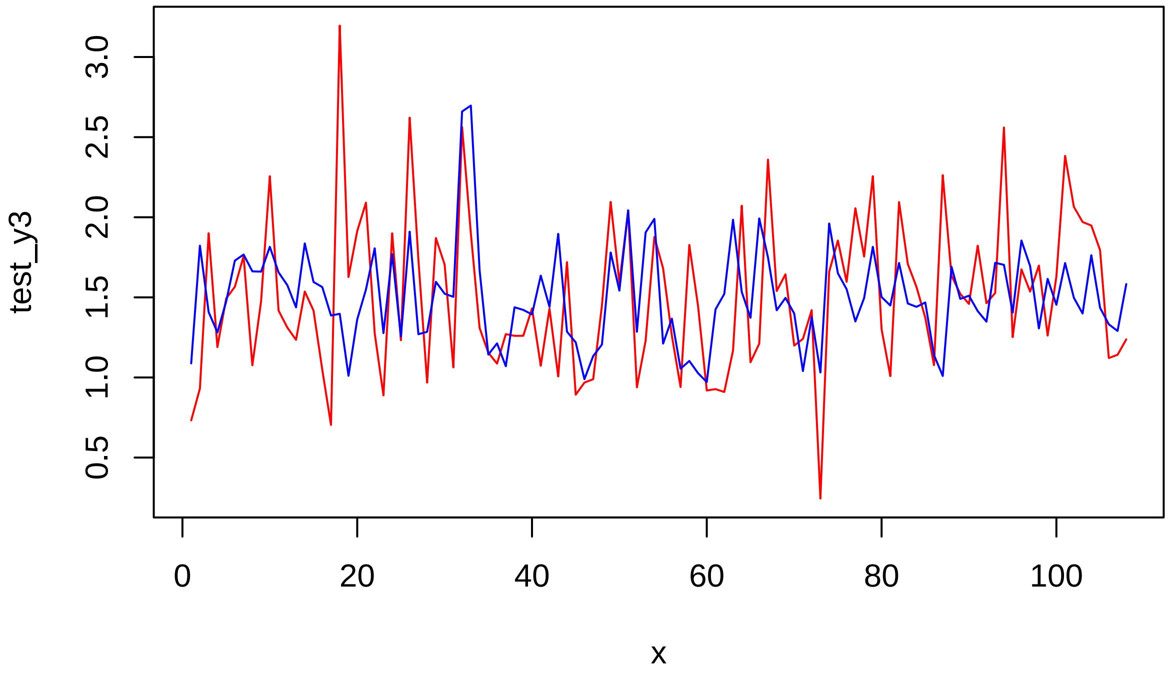
3 YEAR

```
#gradient boosting year 3
set.seed(200)

#define final training and testing sets
xgb_train3 = xgb.DMatrix(data = train_x3, label = train_y3)
xgb_test3 = xgb.DMatrix(data = test_x3, label = test_y3)

#defining a watchlist
watchlist3 = list(train=xgb_train3, test=xgb_test3)

#fit XGBoost model and display training and testing data at each iteartion
model3 = xgb.train(data = xgb_train3, max.depth = 3, watchlist=watchlist3, nrounds = 9)

#define final model
model_xgboost3 = xgboost(data = xgb_train3, max.depth = 3, nrounds = 86, verbose = 0)

summary(model_xgboost3)

pred_y_train3 <- predict(model_xgboost3, xgb_train3)

residuals_xg_3 <- train_y3 - pred_y_train3

# Residuals plot of train data
plot(pred_y_train3, residuals_xg_3, main = "Residuals vs Predicted 3 Year Stock Return XGBoost Model: T
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```

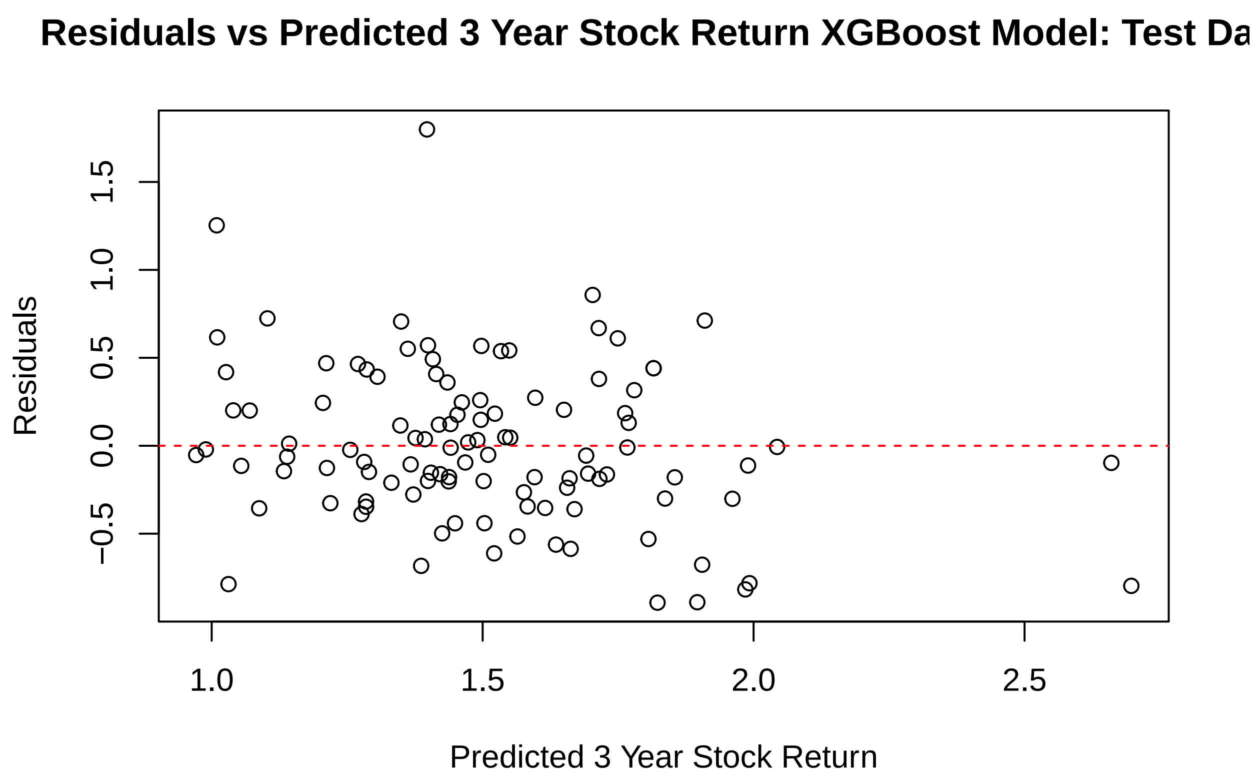


```
pred_y3 = predict(model_xgboost3, xgb_test3)
```

```r
# Model metrics calculations
mean((test_y3 - pred_y3)^2) #mse - Mean Squared Error

mae_grad3 <- mean(abs(test_y3 - pred_y3))

caret::RMSE(test_y3, pred_y3) #rmse - Root Mean Squared Error

y_test_mean3 = mean(test_y3)
# Calculate total sum of squares
tss3 =  sum((test_y3 - y_test_mean3)^2 )
# Calculate residual sum of squares
rss3 =  sum((test_y3 - pred_y3)^2)
# Calculate R-squared
rsq_xgb3  =  1 - (rss3/tss3)
cat('The R-square of the test data is ', round(rsq_xgb3,3), '\n')

# Visualize the model, actual and predicted data
x = 1:length(test_y3)
plot(x, test_y3, col = "red", type = "l")
lines(x, pred_y3, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```r
# Residuals plot of test data
plot(pred_y3, test_y3 - pred_y3, main = "Residuals vs Predicted 3 Year Stock Return XGBoost Model: Test
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



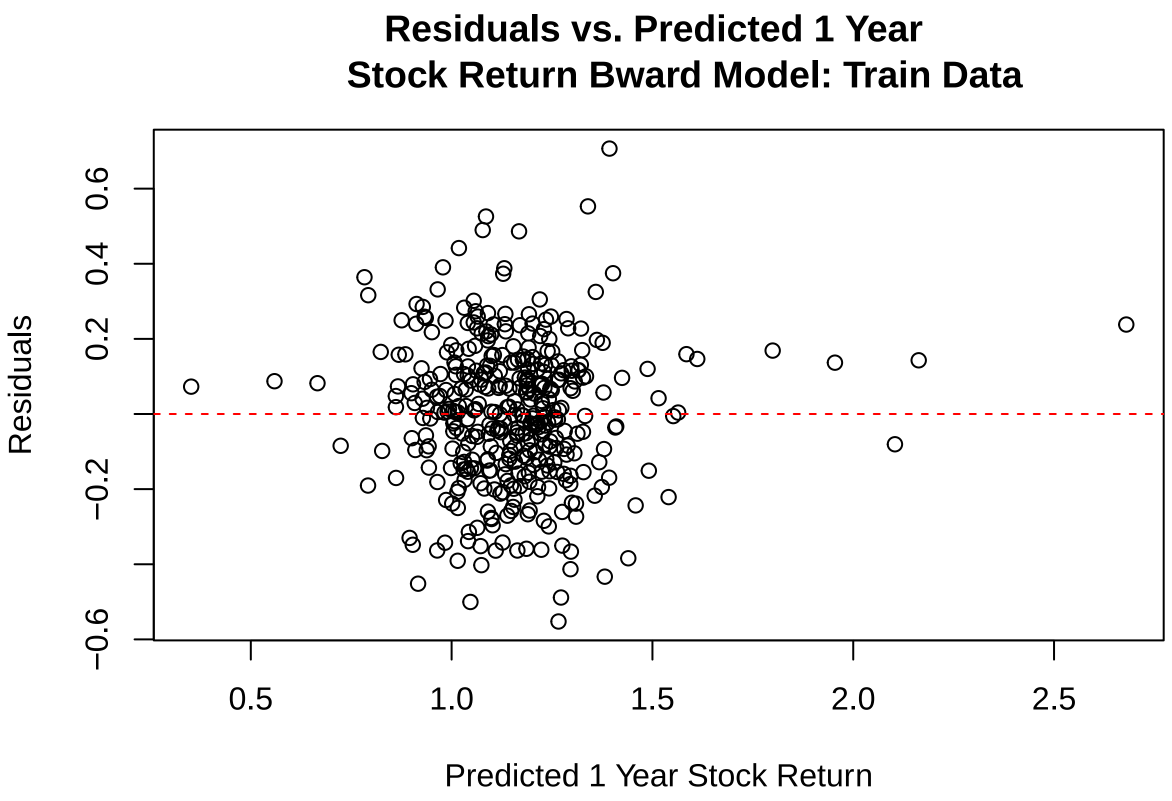
```r
importance_matrix_gradient3 <- xgb.importance(model = model_xgboost3)
```

BACKWARD ELIMINATION

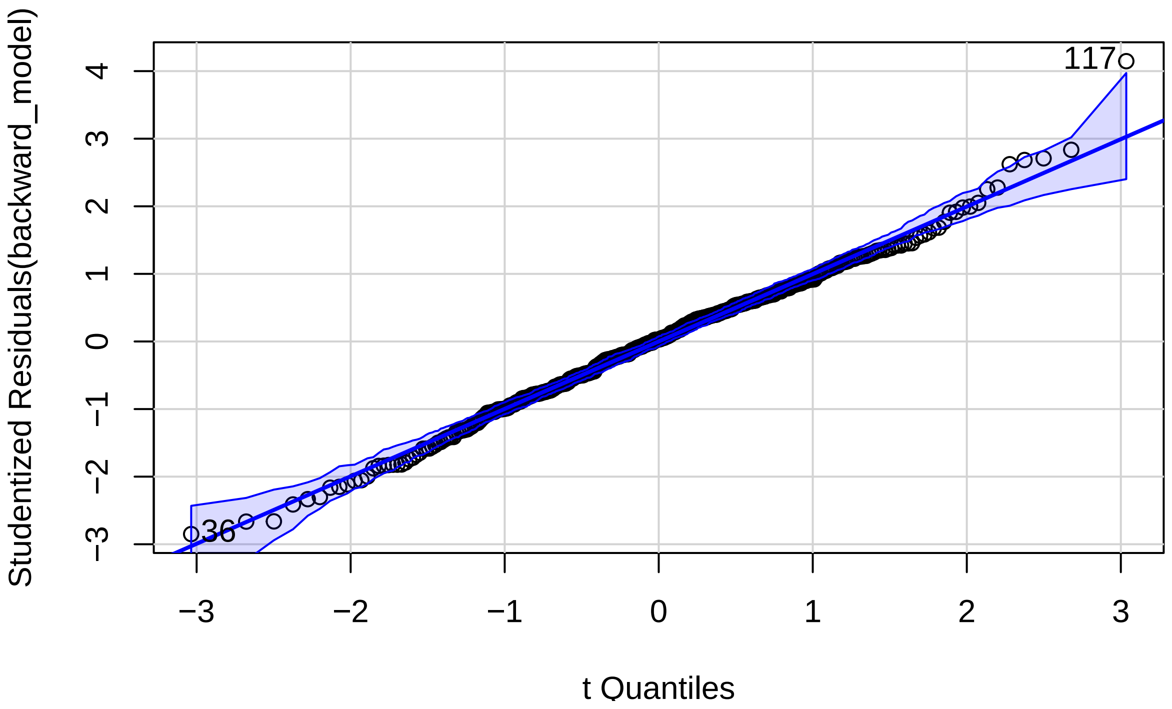
1 YEAR

```r
# Create a full model
# Scale data
numeric_columns <- sapply(train_lm, is.numeric)

stock_return <- train_lm$Stock.Return..1.Year.

# Scale only numeric columns
train_lm[, numeric_columns] <- scale(train_lm[, numeric_columns])

train_lm$Stock.Return..1.Year. <- stock_return

full_model <- lm((Stock.Return..1.Year.) ~ . - Company - Year, data = (train_lm))

# Perform backward elimination
backward_model <- step(full_model, direction = "backward", quiet=T, trace=0)

# Print the summary of the final model
summary(backward_model)
```

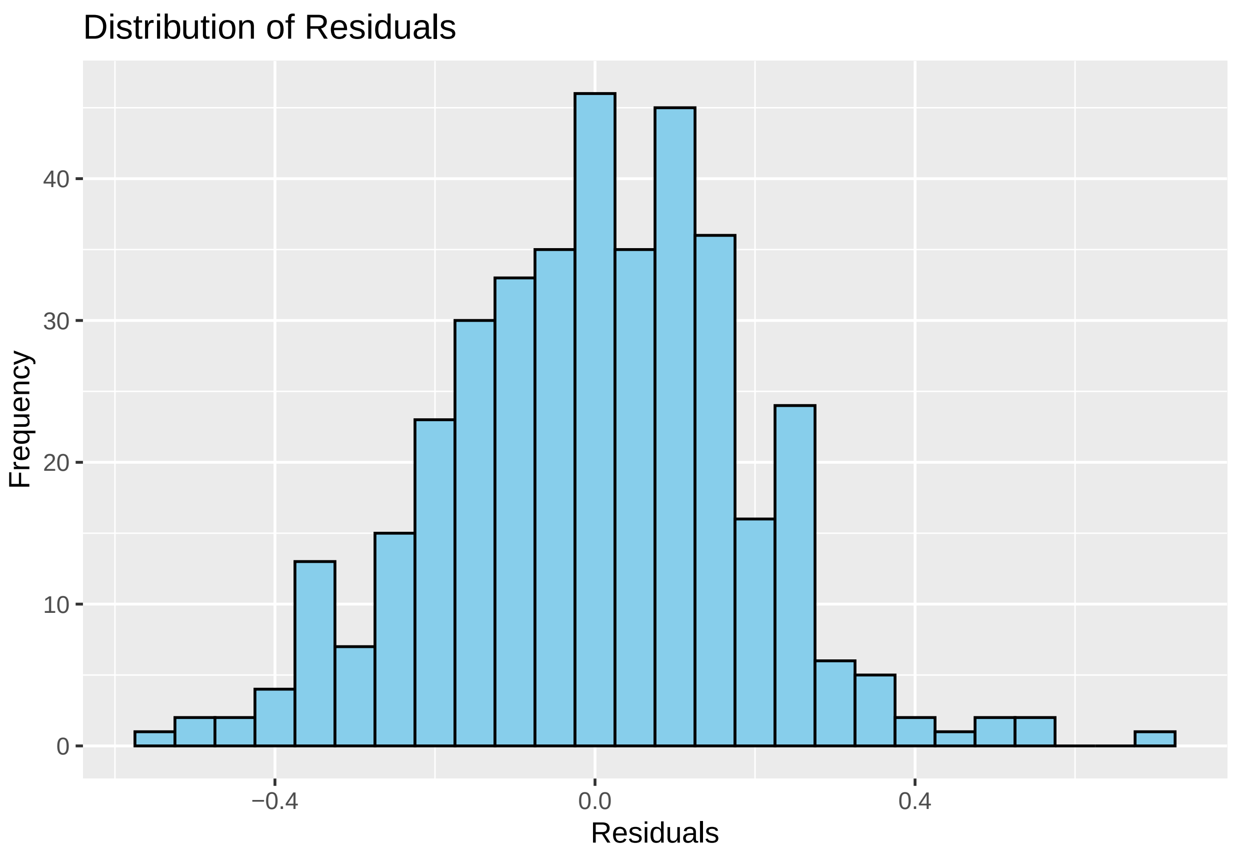
```r
residuals_backward <- residuals(backward_model)
fitted_values <- fitted(backward_model)

# Resdiuals plot of train data
plot(fitted_values, residuals_backward, main = "Residuals vs. Predicted 1 Year
     Stock Return Bward Model: Train Data", xlab = "Predicted 1 Year Stock Return",
     ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```
# Tests
ncvTest(backward_model) #constant variance
dwtest(backward_model) #independence
shapiro.test(residuals(backward_model)) #normality
qqPlot(backward_model)
```

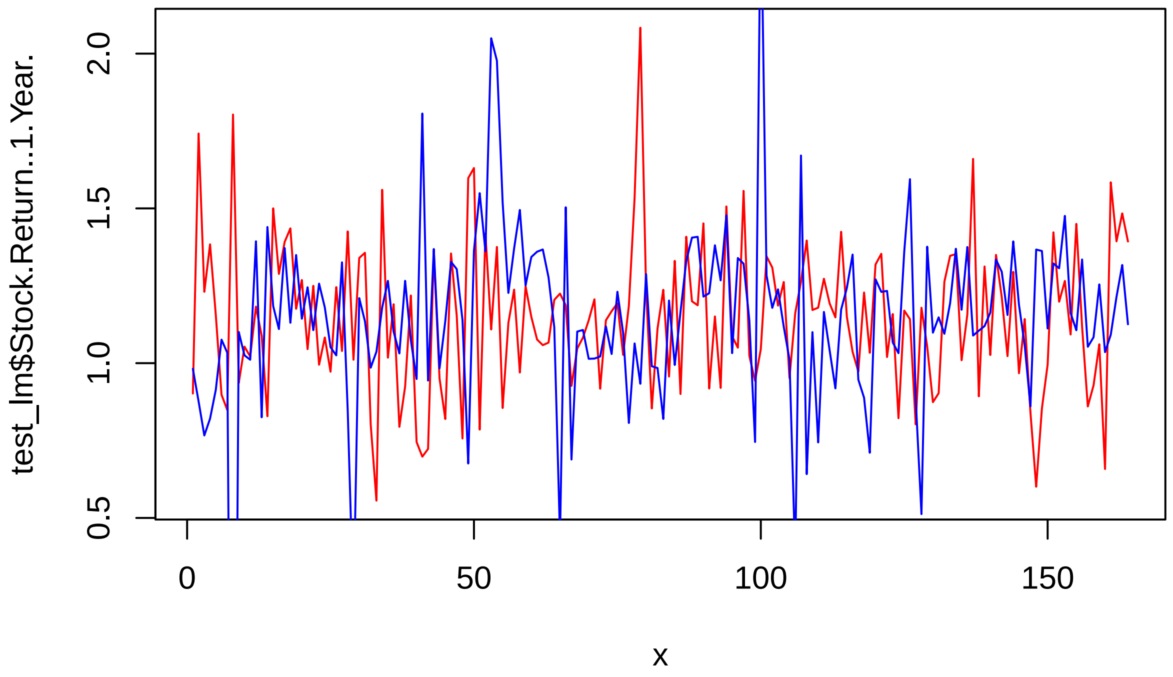


```
residuals_df <- as.data.frame(residuals_backward)

# Histogram of residuals
ggplot(residuals_df, aes(x = residuals_backward)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```

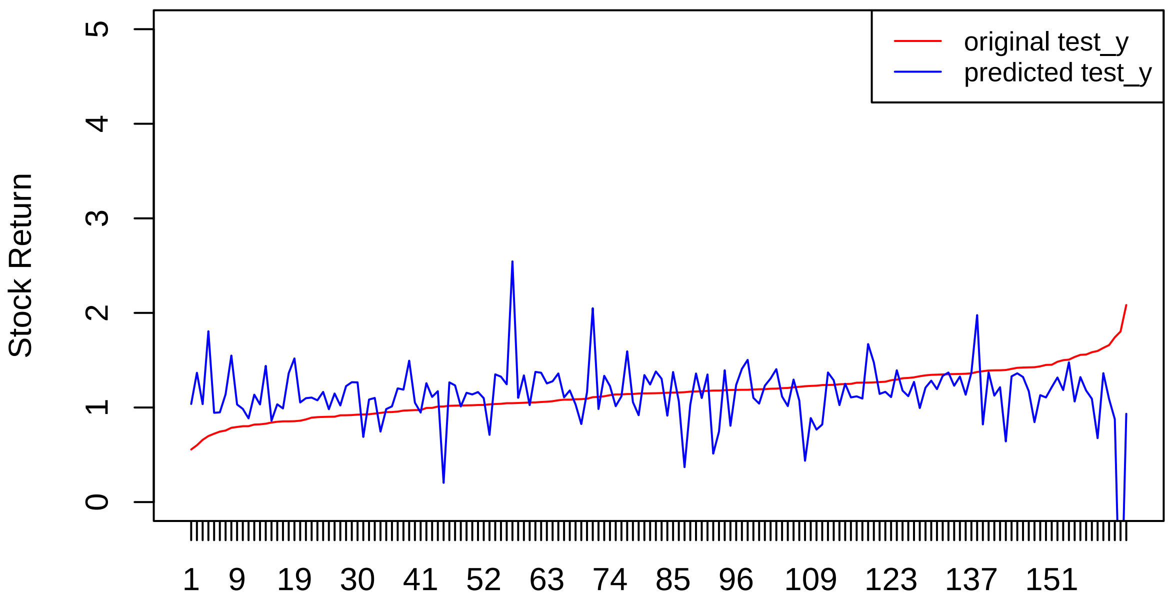


```
# Summary stats
cat("The mean of the distribution is:", paste(mean((residuals_df$residuals_backward)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df$residuals_backward)), "\n")
cat("The range of the distribution is:[", paste(range((residuals_df$residuals_backward))), "]\n")

# Scale test data
numeric_columns11 <- sapply(test_lm, is.numeric)

stock_return11 <- test_lm$Stock.Return..1.Year.

# Scale only numeric columns
test_lm[, numeric_columns11] <- scale(test_lm[, numeric_columns11])

test_lm$Stock.Return..1.Year. <- stock_return11

# MAE calculations
predicted_values_backward <- as.data.frame(predict(backward_model, newdata = test_lm, interval = "confi

predicted_values_backward$actual <- test_lm$Stock.Return..1.Year.

predicted_values_backward$resid <- (predicted_values_backward$actual) - (predicted_values_backward$fit)

predicted_values_backward$resid_mag <- abs(predicted_values_backward$resid)

mae_bward1 <- mean(predicted_values_backward$resid_mag)

# Visualize the model, actual and predicted data
```

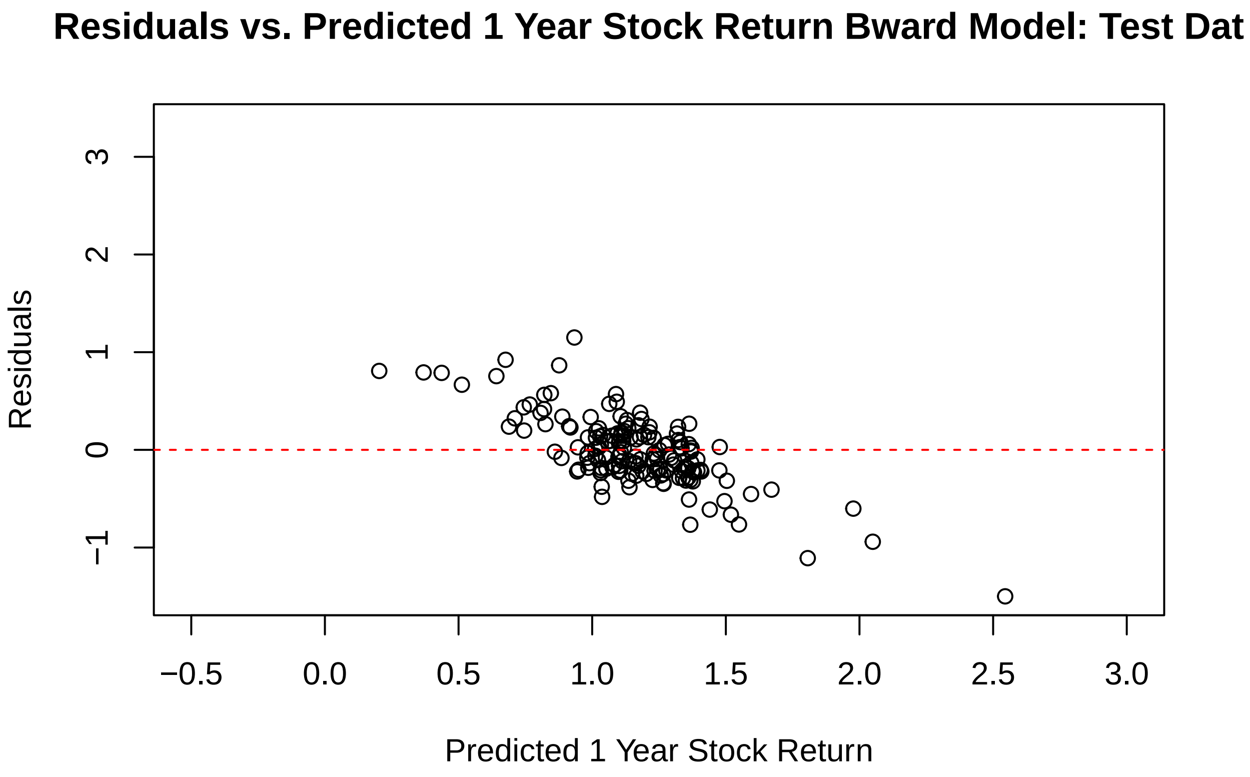
```
x = 1:length(test_lm$Stock.Return..1.Year.)
plot(x, test_lm$Stock.Return..1.Year., col = "red", type = "l")
lines(x, (predicted_values_backward$fit), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```
plot_data <- data.frame(
  x = 1:length(test_lm$Stock.Return..1.Year.),
  actual = test_lm$Stock.Return..1.Year.,
  predicted = (predicted_values_backward$fit)
)

# Order the data frame by the actual stock return
plot_data <- plot_data[order(plot_data$actual), ]

# Plot the ordered data
plot(plot_data$actual, col = "red", type = "l", xaxt = "n", xlab = "", ylab = "Stock Return", ylim = c(
lines(plot_data$predicted, col = "blue", type = "l")

# Add x-axis labels
axis(1, at = plot_data$x, labels = plot_data$x)

# Add legend
legend(x = "topright", legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



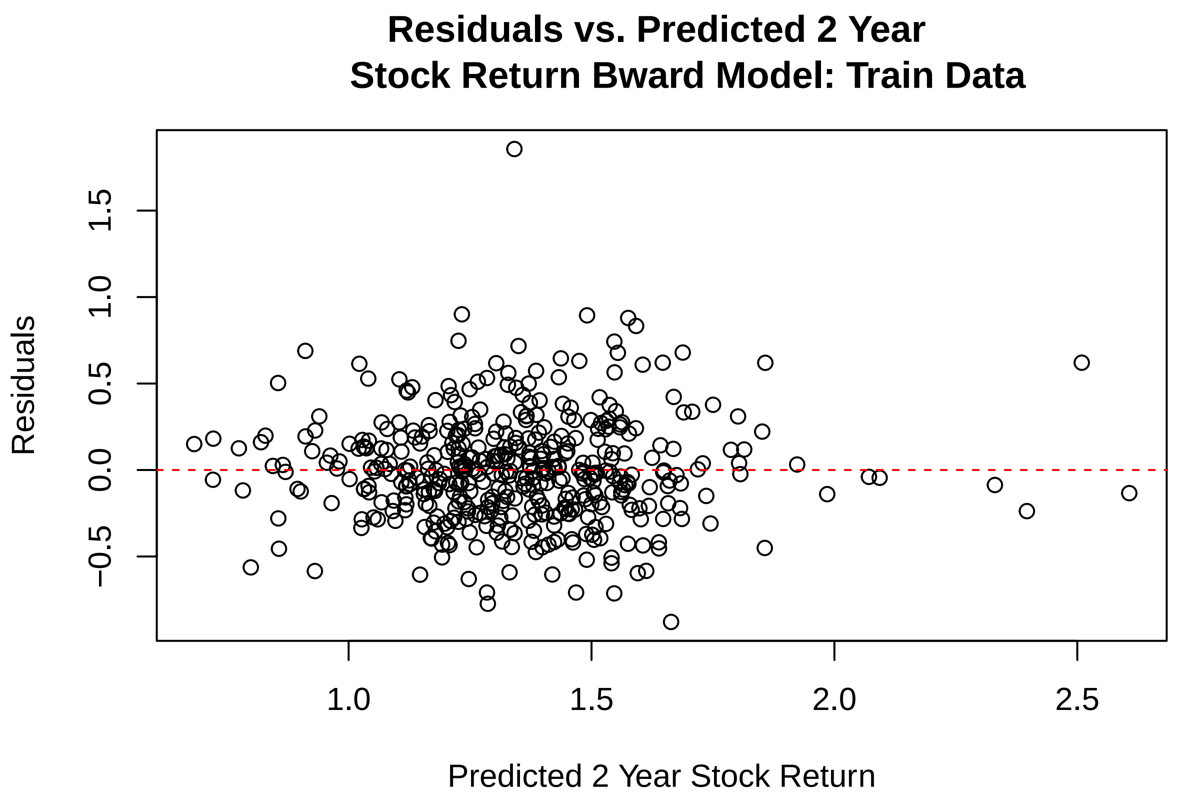
```r
# Create a scatter plot of residuals of test data
plot(x = predicted_values_backward$fit, y = predicted_values_backward$resid, main = "Residuals vs. Predi
     xlab = "Predicted 1 Year Stock Return", ylab = "Residuals", xlim = c(-0.5,3))

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



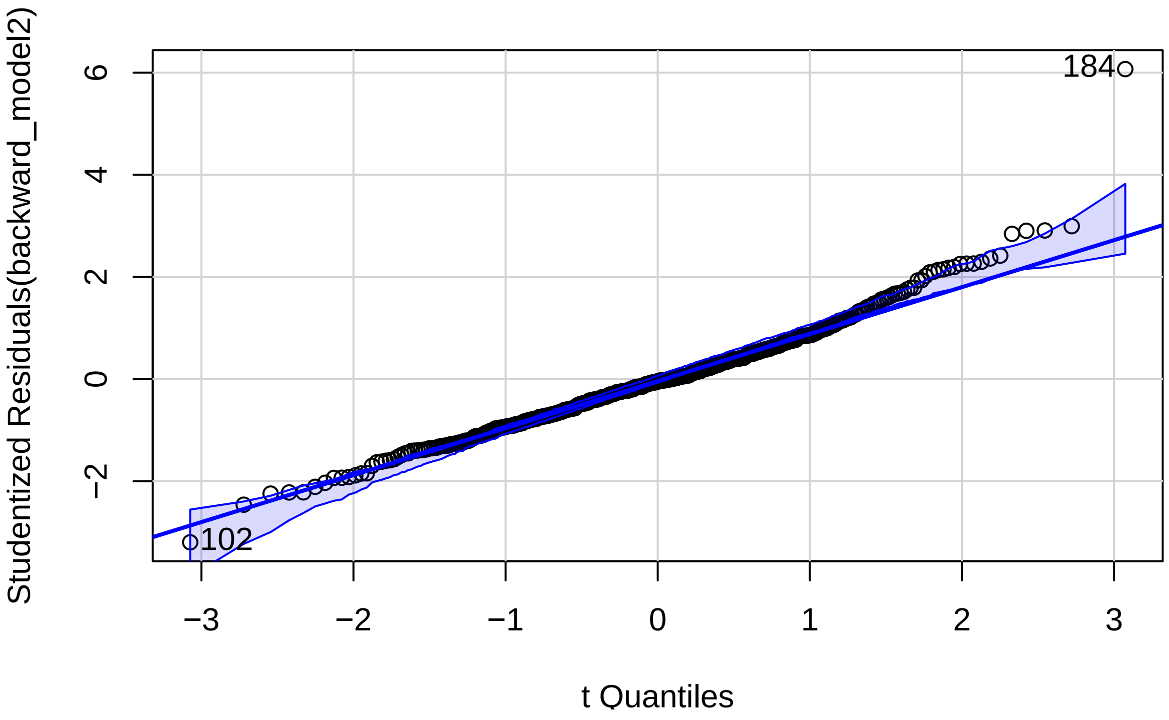
YEAR 2

```r
# Create a full model for year 2
# Scale data
numeric_columns2 <- sapply(train_lm2, is.numeric)

stock_return2 <- train_lm2$Stock.Return..2.Year.

# Scale only numeric columns
train_lm2[, numeric_columns2] <- scale(train_lm2[, numeric_columns2])

train_lm2$Stock.Return..2.Year. <- stock_return2

full_model2 <- lm((Stock.Return..2.Year.) ~ . - Company - Year, data = train_lm2)

# Perform backward elimination
backward_model2 <- step(full_model2, direction = "backward", quiet=F, trace=0)

# Print the summary of the final model
summary(backward_model2)
```

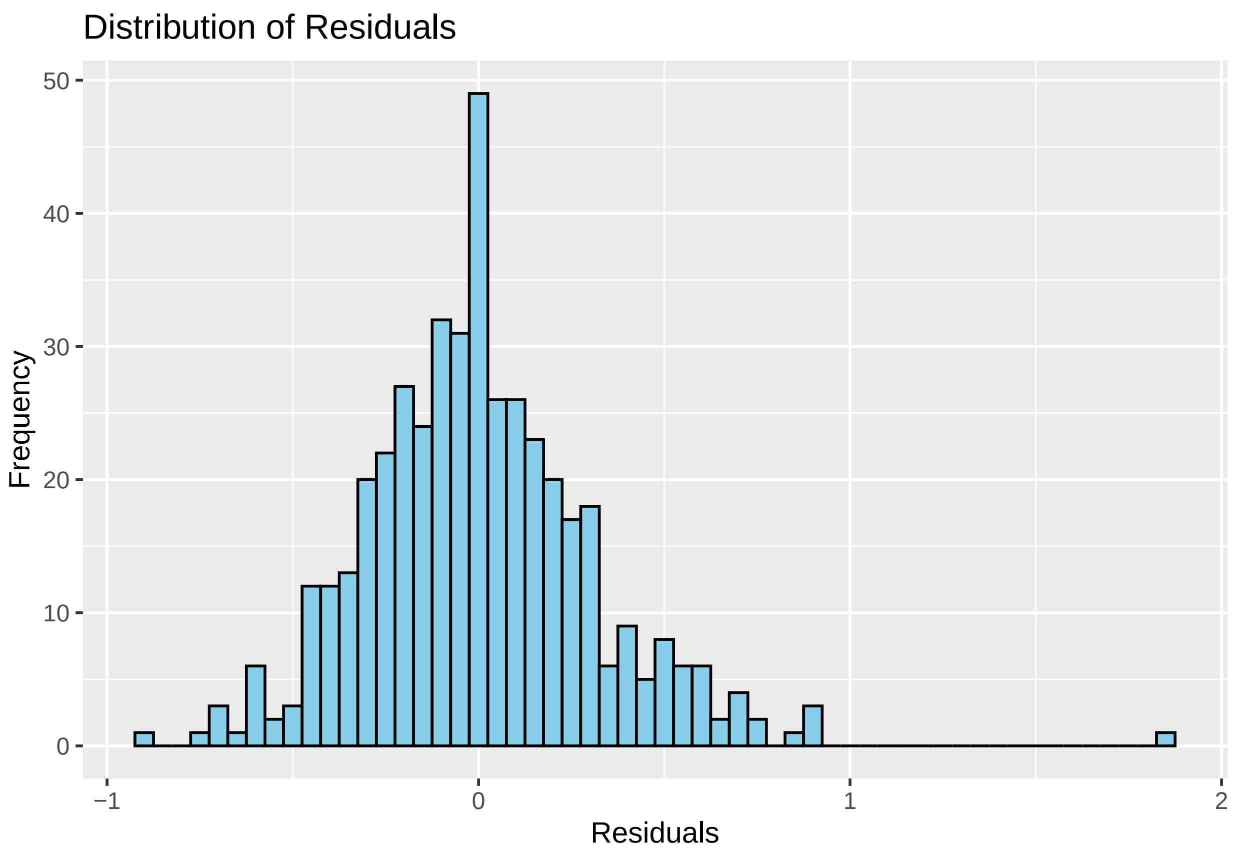
```r
residuals_backward2 <- residuals(backward_model2)
fitted_values2 <- fitted(backward_model2)

# Residuals plot of train data
plot(fitted_values2, residuals_backward2, main = "Residuals vs. Predicted 2 Year
     Stock Return Bward Model: Train Data", xlab = "Predicted 2 Year Stock Return",
     ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```r
# Tests
ncvTest(backward_model2) #constant variance
dwtest(backward_model2) #independence
shapiro.test(residuals(backward_model2)) #normality
qqPlot(backward_model2)
```

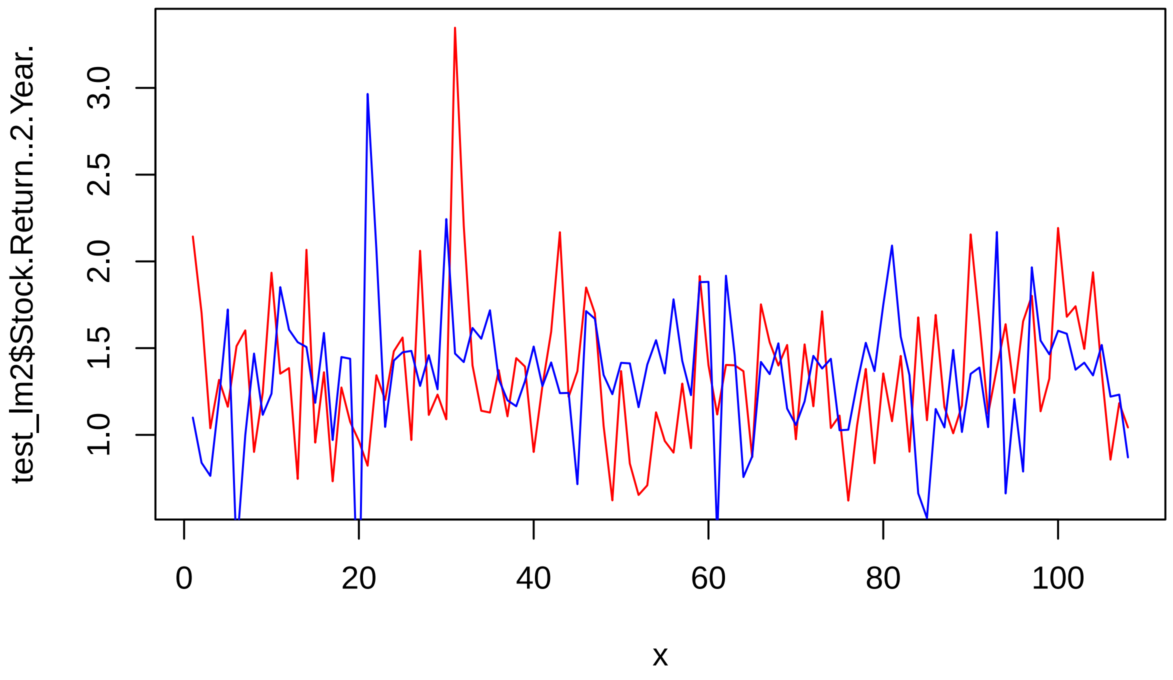


```r
residuals_df2 <- as.data.frame(residuals_backward2)

# Histogram of residuals
ggplot(residuals_df2, aes(x = residuals_backward2)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```

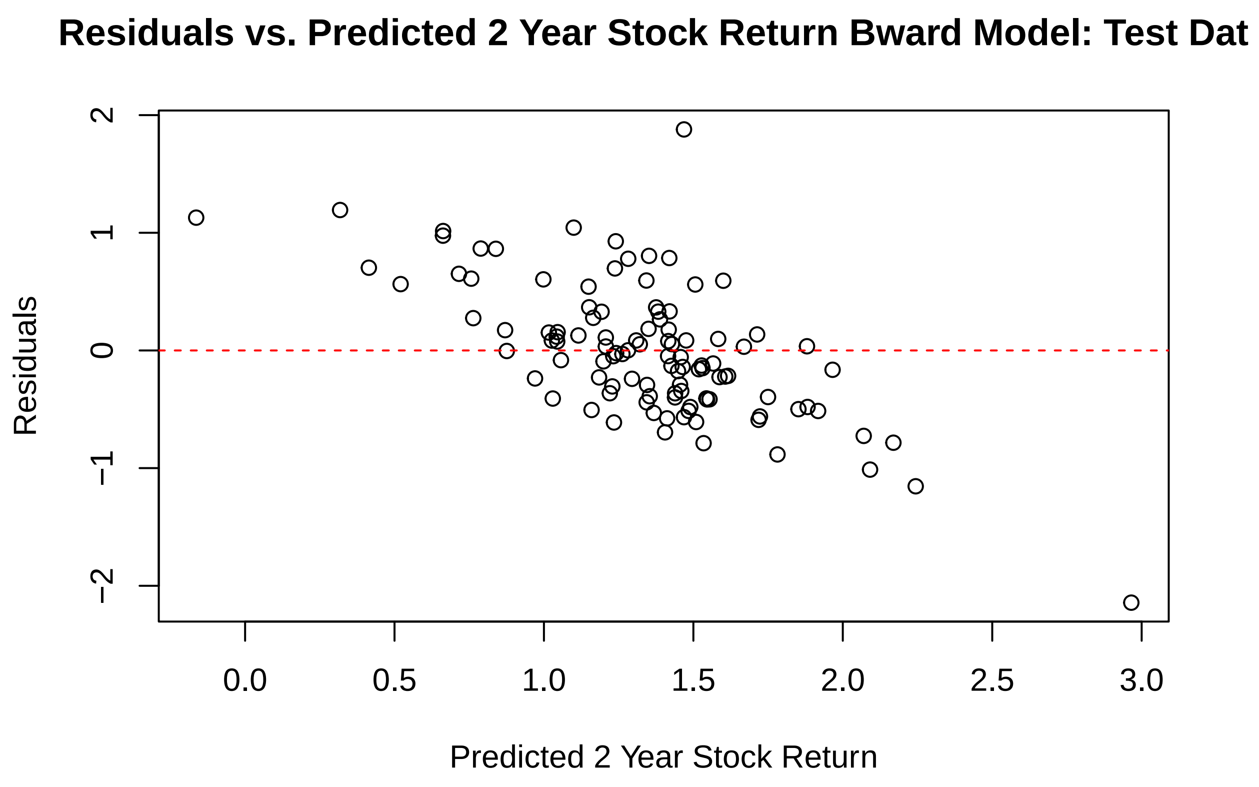


```
# Summary stats
cat("The mean of the distribution is:", paste(mean((residuals_df2$residuals_backward2)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df2$residuals_backward2)), "\n
cat("The range of the distribution is:[", paste(range((residuals_df2$residuals_backward2))), "]\n")

# Scale test data
numeric_columns3 <- sapply(test_lm2, is.numeric)

stock_return22 <- test_lm2$Stock.Return..2.Year.

# Scale only numeric columns
test_lm2[, numeric_columns3] <- scale(test_lm2[, numeric_columns3])

test_lm2$Stock.Return..2.Year. <- stock_return22

# MAE calculations
predicted_values_backward2 <- as.data.frame(predict(backward_model2, newdata = test_lm2, interval = "co

predicted_values_backward2$actual <- test_lm2$Stock.Return..2.Year.

predicted_values_backward2$resid <- (predicted_values_backward2$actual) - predicted_values_backward2$fi

predicted_values_backward2$resid_mag <- abs(predicted_values_backward2$resid)

mae_bward2 <- mean(predicted_values_backward2$resid_mag)

# Visualize the model, actual and predicted data
```

```r
x = 1:length(test_lm2$Stock.Return..2.Year.)
plot(x, test_lm2$Stock.Return..2.Year., col = "red", type = "l")
lines(x, (predicted_values_backward2$fit), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



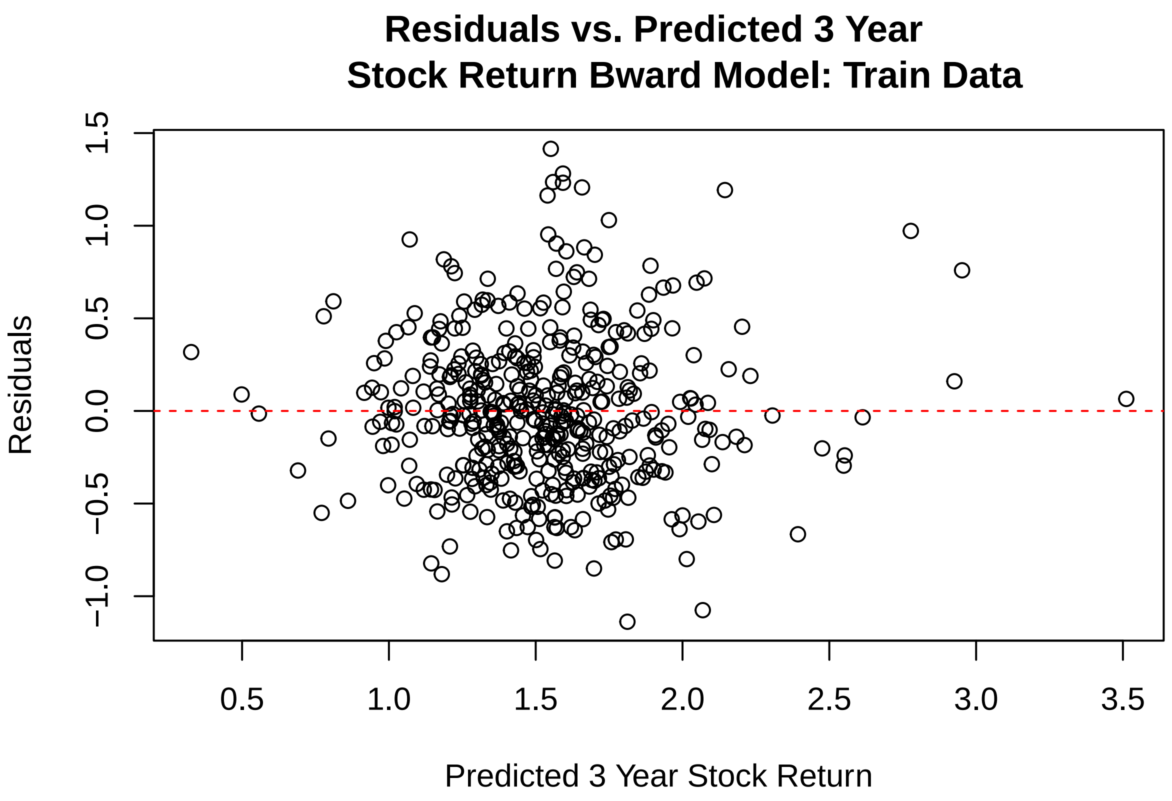
```r
# Create a scatter plot of residuals of test data
plot(x = predicted_values_backward2$fit, y = predicted_values_backward2$resid, main = "Residuals vs. Pre
     xlab = "Predicted 2 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```



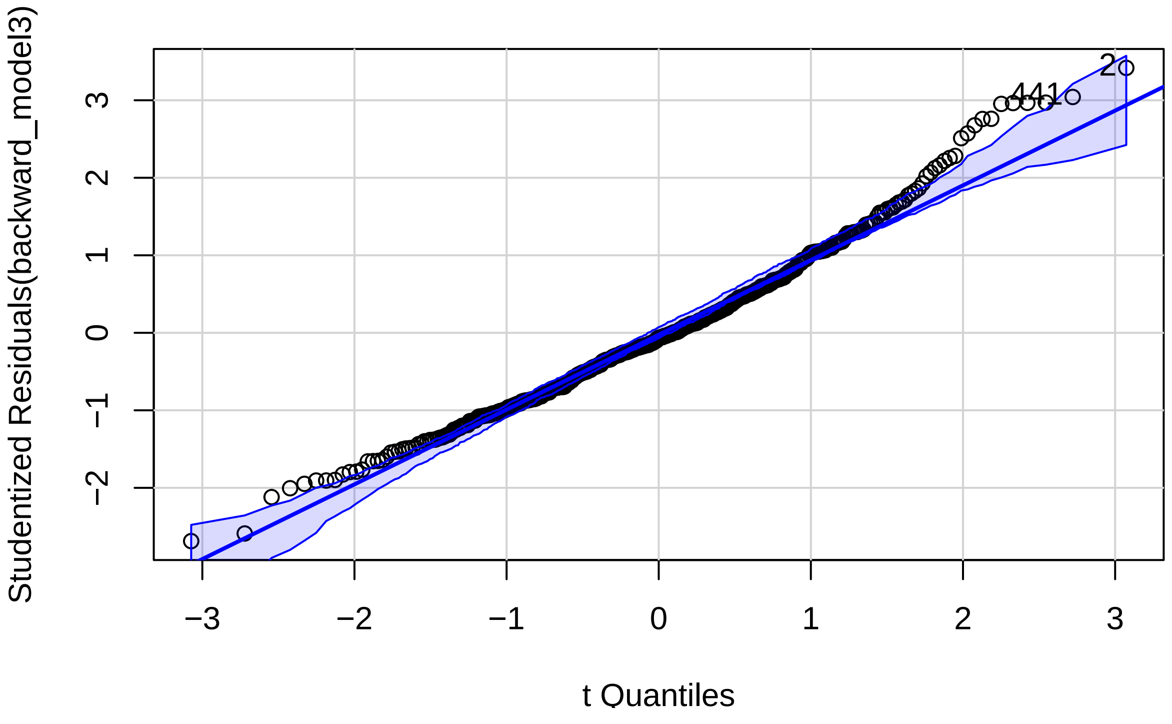
YEAR 3

```r
# Create a full model for year 3
# Scale data
numeric_columns3 <- sapply(train_lm3, is.numeric)

stock_return3 <- train_lm3$Stock.Return..3.Year.

# Scale only numeric columns
train_lm3[, numeric_columns3] <- scale(train_lm3[, numeric_columns3])

train_lm3$Stock.Return..3.Year. <- stock_return3

full_model3 <- lm((Stock.Return..3.Year.) ~ . - Company - Year, data = train_lm3)

# Perform backward elimination
backward_model3 <- step(full_model3, direction = "backward", quiet=T, trace=0)

# Print the summary of the final model
summary(backward_model3)
```

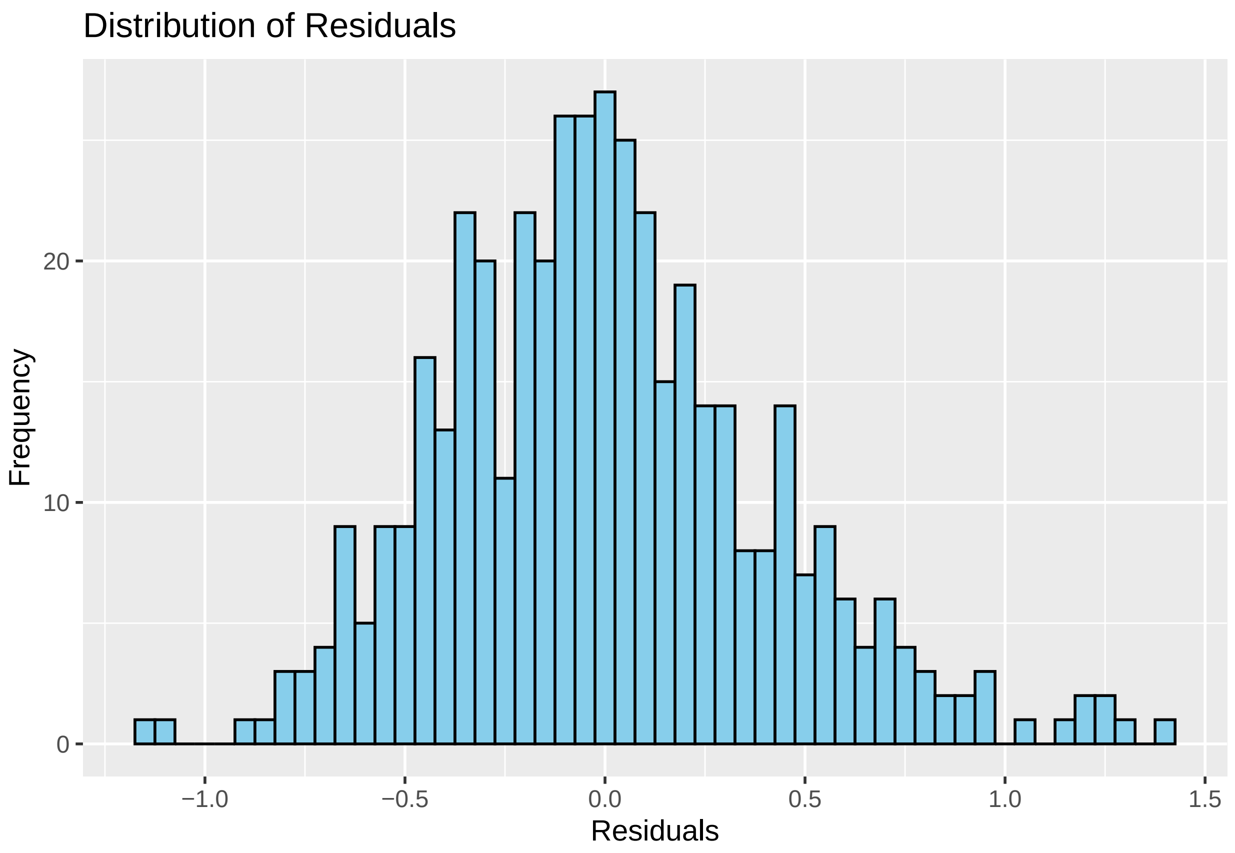
```r
residuals_backward3 <- residuals(backward_model3)
fitted_values3 <- fitted(backward_model3)

# Residuals plot of train data
plot(fitted_values3, residuals_backward3, main = "Residuals vs. Predicted 3 Year
     Stock Return Bward Model: Train Data", xlab = "Predicted 3 Year Stock Return",
     ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)  # Add a horizontal line at y = 0
```



```r
# Tests
ncvTest(backward_model3) #constant variance
dwtest(backward_model3) #independence
shapiro.test(residuals(backward_model3)) #normality
qqPlot(backward_model3)
```

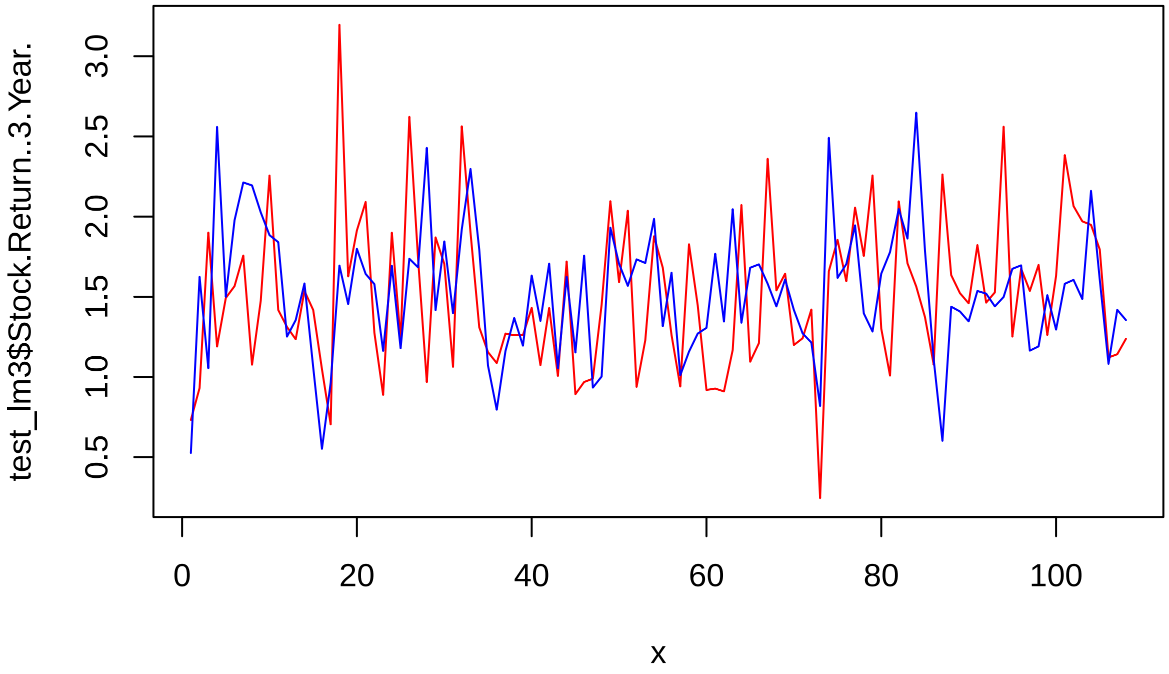


```r
residuals_df3 <- as.data.frame(residuals_backward3)

# Histogram fo residuals
ggplot(residuals_df3, aes(x = residuals_backward3)) +
  geom_histogram(binwidth = 0.05, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Residuals", x = "Residuals", y = "Frequency")
```

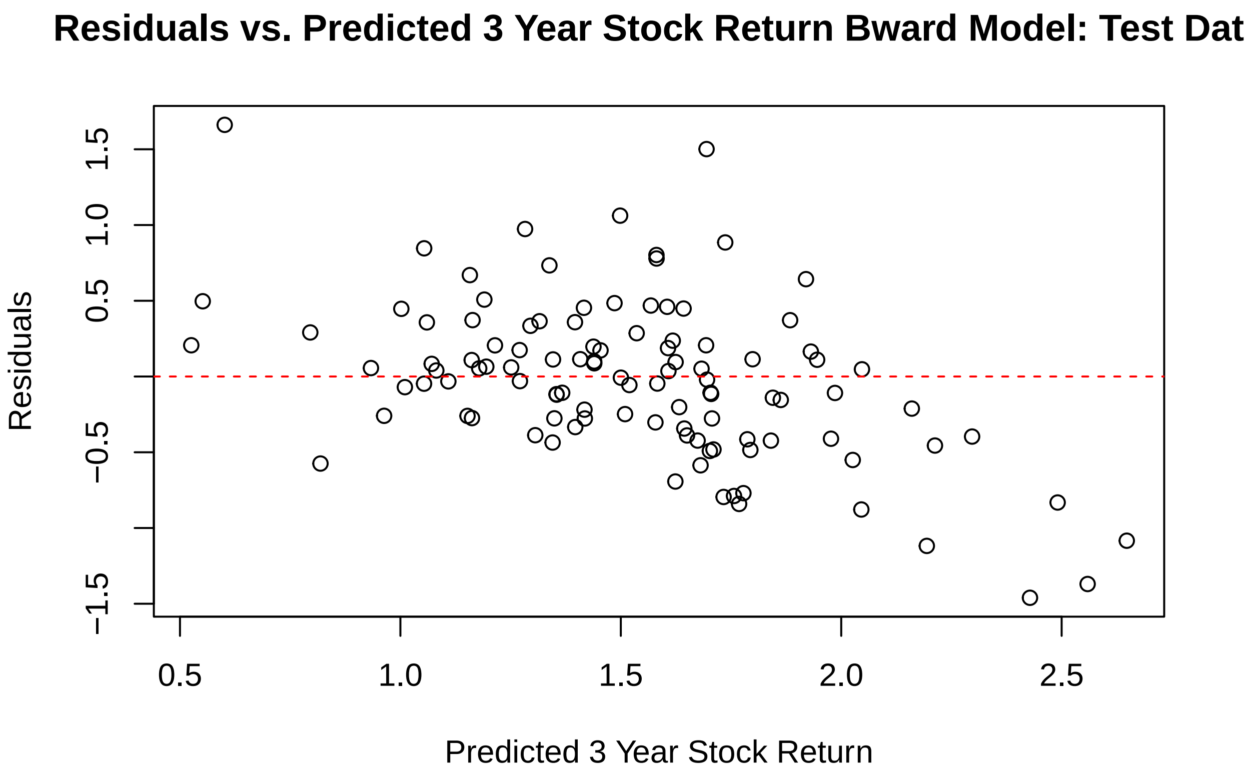


```r
cat("The mean of the distribution is:", paste(mean((residuals_df3$residuals_backward3)^2)), "\n")
cat("The standard deviation of the distribution is:", paste(sd(residuals_df3$residuals_backward3)), "\n
cat("The range of the distribution is:[", paste(range((residuals_df3$residuals_backward3))), "]\n")

# Scale test data
numeric_columns4 <- sapply(test_lm3, is.numeric)

stock_return33 <- test_lm3$Stock.Return..3.Year.

# Scale only numeric columns
test_lm3[, numeric_columns4] <- scale(test_lm3[, numeric_columns4])

test_lm3$Stock.Return..3.Year. <- stock_return33

# MAE calculations
predicted_values_backward3 <- as.data.frame(predict(backward_model3, newdata = test_lm3, interval = "co

predicted_values_backward3$actual <- test_lm3$Stock.Return..3.Year.

predicted_values_backward3$resid <- (predicted_values_backward3$actual) - predicted_values_backward3$fi
```

```r
predicted_values_backward3$resid_mag <- abs(predicted_values_backward3$resid)

mae_bward3 <- (mean(predicted_values_backward3$resid_mag))

backward3 <- augment(backward_model3)

# Visualize the model, actual and predicted data
x = 1:length(test_lm3$Stock.Return..3.Year.)
plot(x, test_lm3$Stock.Return..3.Year., col = "red", type = "l")
lines(x, (predicted_values_backward3$fit), col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```r
# Create a scatter plot of residuals of test data
plot(x = predicted_values_backward3$fit, y = predicted_values_backward3$resid, main = "Residuals vs. Pre
     xlab = "Predicted 3 Year Stock Return", ylab = "Residuals")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red", lty = 2)
```


Residuals vs. Predicted 3 Year Stock Return Bward Model: Test Dat
Residuals
1.5
1.0
0.5
−0.5
−1.5
0.5
1.0
1.5
2.0
2.5
Predicted 3 Year Stock Return