Project 1 ‘Part 4’-‘Regression’

Sachin Samal (ECU ID 250008)

Nov 12, 2021

DATA MINING WITH R

Here in this part of project, I have decided to perform the regression analysis on my previous dataset from **Project 1**, **Project 2** and **Project 3**. For this project, I have pulled the data from **English Premier League Results**.

WHAT IS REGRESSION?

It is a data mining technique used to predict the range of numerical values given in a particular dataset. It is different from association but quite similar to classification. For comparison, you can visit, my Project 2 where I have performed classification on this dataset. Regression might be used to predict the cost of product or service, financial forecasting, environmental modeling and analysis of trends.

```
library(readxl)
Results <- read_excel("Results.xlsx")
str(Results)
```

LOADING MY EXCEL DATA INTO R ENVIRONMENT

```
## tibble [380 x 6] (S3: tbl_df/tbl/data.frame)
##  $ Home_team: chr [1:380] "Arsenal" "Watford" "Chelsea" "Crystal Palace" ...
##  $ Away_team: chr [1:380] "Leicester City" "Liverpool" "Burnley" "Huddersfield Town" ...
##  $ Home_goal: num [1:380] 4 3 2 0 1 0 1 0 0 4 ...
##  $ Away_goal: num [1:380] 3 3 3 3 0 0 0 2 2 0 ...
##  $ Result   : chr [1:380] "H" "D" "A" "A" ...
##  $ Season   : chr [1:380] "2017-2018" "2017-2018" "2017-2018" "2017-2018" ...
```

```
x<-c(Results$Home_goal)
y<-c(Results$Away_goal)
```

LETS START OUT REGRESSION MINING... Correlation describes the “degree of relationship” between two variables. It ranges from -1 to +1. Negative values indicate that as one variable increases the other variable decreases. Positive values indicate that as one variable

increase the other variable increases as well. There are three options to calculate correlation in R, and we will introduce two of them below.

```
cor(Results$Home_goal, Results$Away_goal, method="pearson")
```

To get the Correlation R

```
## [1] -0.1300056
```

Here, my correlation value is negative that means, when Home goal increases, Away goal decreases, which practically makes sense.

I think I'm heading in a right direction.

We used the 'Pearson' correlation method here.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{(n-1)s_x s_y}$$

Let x=Home_goal, y=Away_goal, then;

```
sx = sd(x)
sy = sd(y)
n = length(x)
xbar = mean(x)
ybar = mean(y)
numerator = mean((x-xbar)*(y-ybar))*n
r = numerator/((n-1)*sx*sy)
r
```

```
## [1] -0.1300056
```

Now, Lets perform correlation test to futher examine the detail relationship between this two variables:

```
cor.test(Results$Home_goal, Results$Away_goal)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Results$Home_goal and Results$Away_goal
## t = -2.5492, df = 378, p-value = 0.01119
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.22763028 -0.02979344
## sample estimates:
##        cor
## -0.1300056
```

We already had a negative correlation value of -0.13, which is once again verified through this method.

Also, looks like we have a very small p-value and 95% confidence interval of the correlation.

Let's also get the equation of the linear relationship. I just have two variables with numerical values so let me go with one function and see if the intercept in the linear regression is significant.

```
lm(Results$Away_goal~Results$Home_goal)
```

```
##
## Call:
## lm(formula = Results$Away_goal ~ Results$Home_goal)
##
## Coefficients:
##       (Intercept)  Results$Home_goal
##            1.3224            -0.1143
```

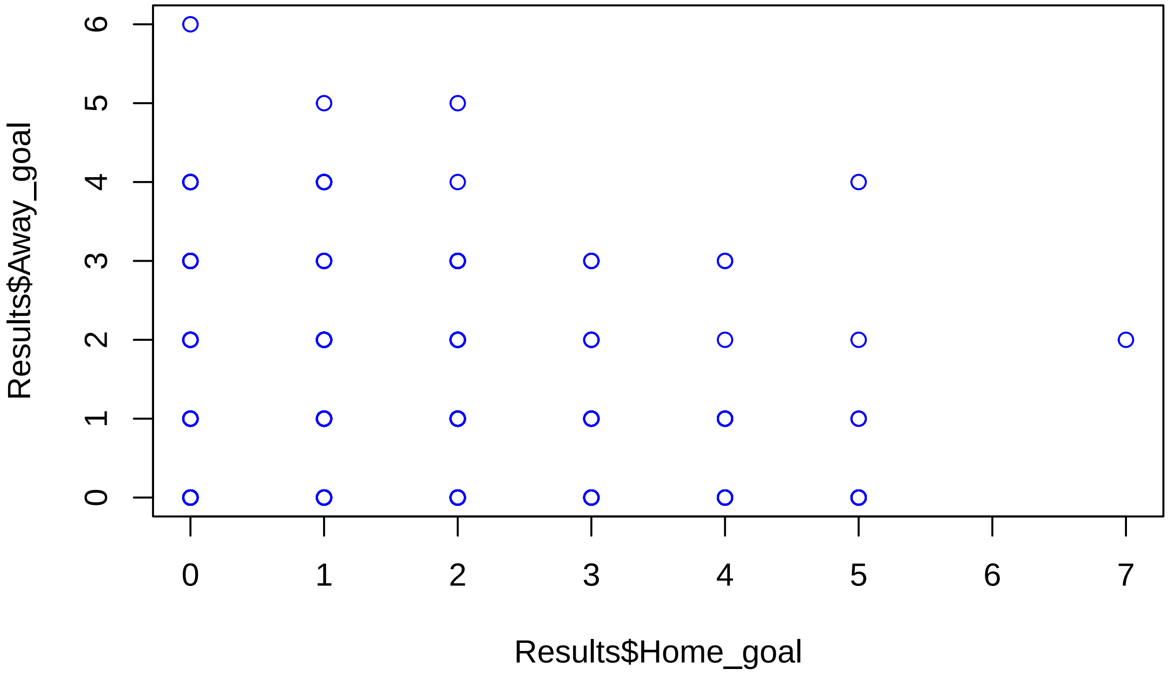
The argument `Results$Away_goal~Results$Home_goal` to lm function is a model formula.

I performed my function as I did **y~x** i.e. **`Results$Away_goal~Results$Home_goal`**. Let's make some predictions:

```
fit = lm(y~x)
slope = coef(fit)[2]
intercept = coef(fit)[1]
yhat <- function(xpredict){
  return( slope*xpredict+intercept)
}
yhat(10)
```

```
##         x
## 0.1797584
```

```
plot(Results$Away_goal~Results$Home_goal, data = Results, col = "blue")
```



```
Results.num = Results[c("Home_goal", "Away_goal")]
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

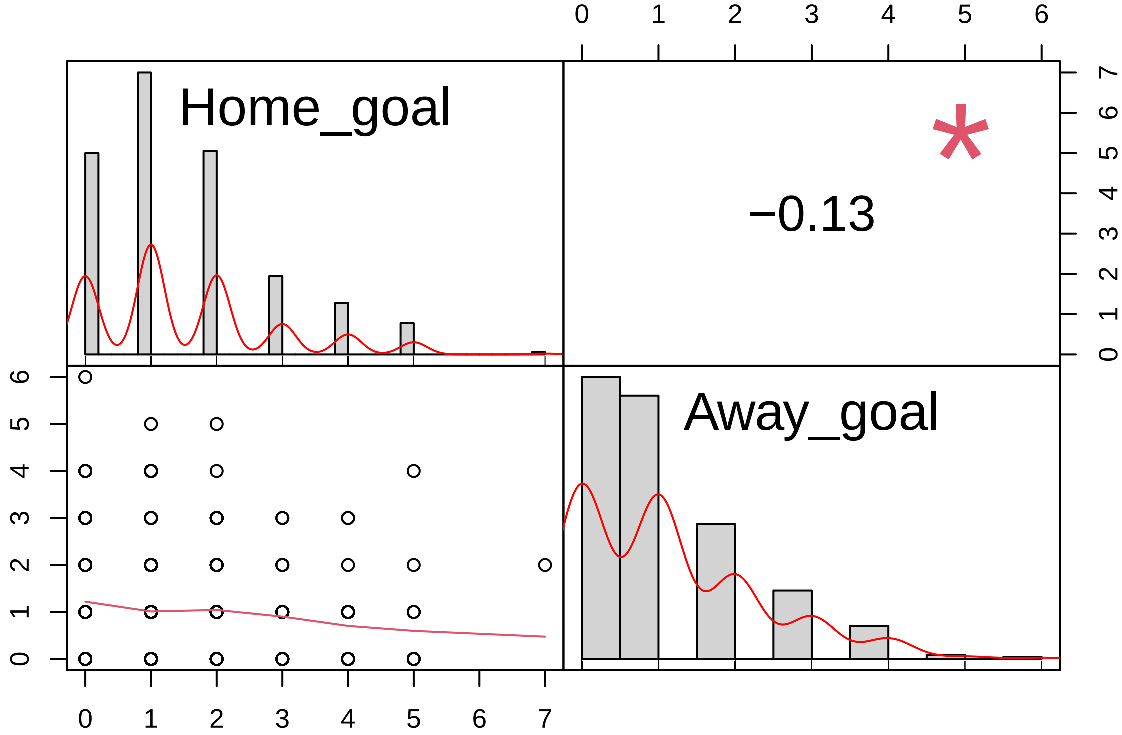
```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(Results.num,
                   method="pearson",
                   histogram=TRUE,
                   pch=10)
```



These statistics vary from −1 to 1, with 0 indicating no correlation, 1 indicating a perfect positive correlation, and −1 indicating a perfect negative correlation. Like other effect size statistics, these statistics are not affected by sample size.

We have already computed our estimate of the test statistic r. Then t is

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

Here the degrees of freedom will be $df = n - 2$

```
t = r*sqrt(n-2)/sqrt(1-r^2)
t
```

```
## [1] -2.549232
```

A negative degree of freedom is valid as long as I don't get Infinity. It suggests that we have more statistics than we have values that can change. In this case, we have more parameters in the model than we have rows of data or observations to train the model which absolutely makes sense.

This 't' value verifies that we are in a right direction as we already calculated it from the cor.test().

In case we want to test the slope of the line. We consider the following as the theorectical fit:

$$\hat{y} = \beta_0 + \beta_1 x$$

The standard error for the slope is

$$SE_{\beta_1} = \sqrt{\frac{MSE}{\sum (x_i - \overline{x})^2}}$$

Where

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n-2}$$

MSE is referred to as the mean square error. It adds all the square errors and divides by the adjusted total $(n-2)$ because of the degrees of freedom!

```
yframe = data.frame('Result' = y)
yhat <- predict(fit,yframe)
MSE = sum((y-yhat)^2)/(n-2)
MSE
```

```
## [1] 1.367358
```

The denominator in the SE computation above is sometimes called the S_{xx} The sum of the squares of x.

```
sxx = sum((x-xbar)^2)
```

Now, we can now calculate the standard error for the slope:

```
SEslope = sqrt(MSE/sxx)
SEslope
```

```
## [1] 0.04482172
```

In case, I have to look at the intercept, β_0, it's standard error is calculated as:

$$SE_{\beta_0} = \sqrt{MSE\left(\frac{1}{n} + \frac{\overline{x}}{S_{xx}}\right)}$$

All the value have been computed already. So, the next thing is going to be:

```
SEintercept = sqrt(MSE*(1/n+xbar^2/sxx))
SEintercept
```

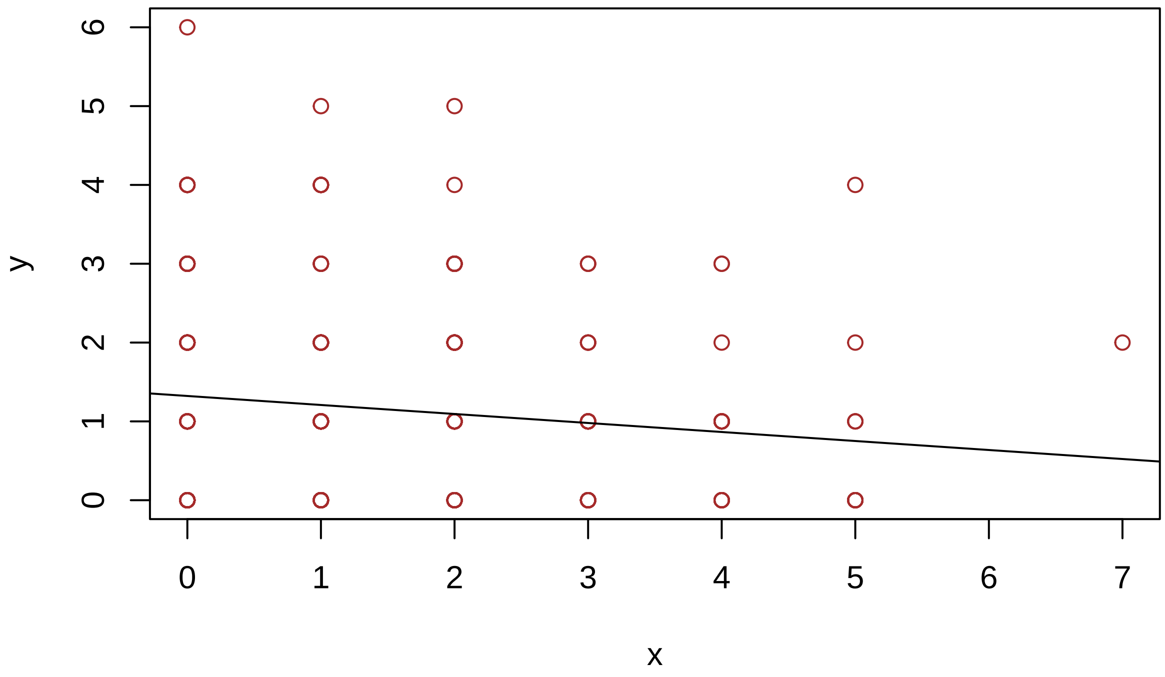
```
## [1] 0.09116391
```

Now, Let me see the summary before I proceed towards the graph.

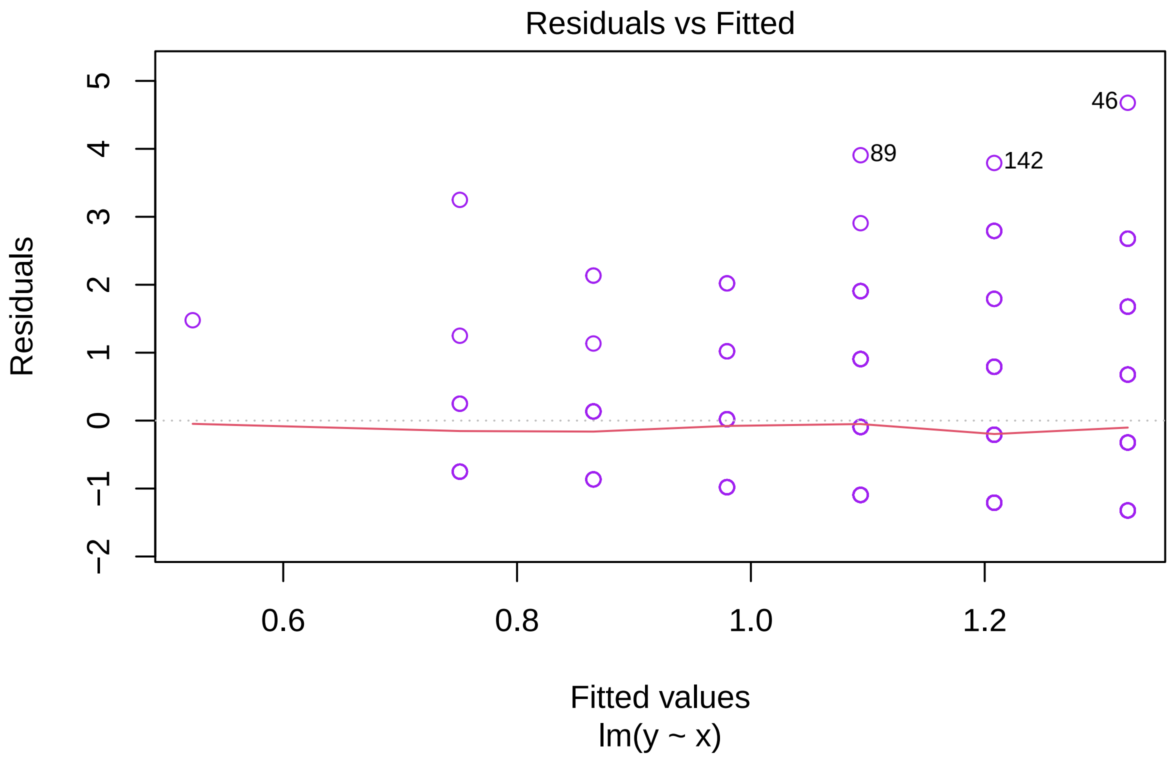
```
summary(fit)
```

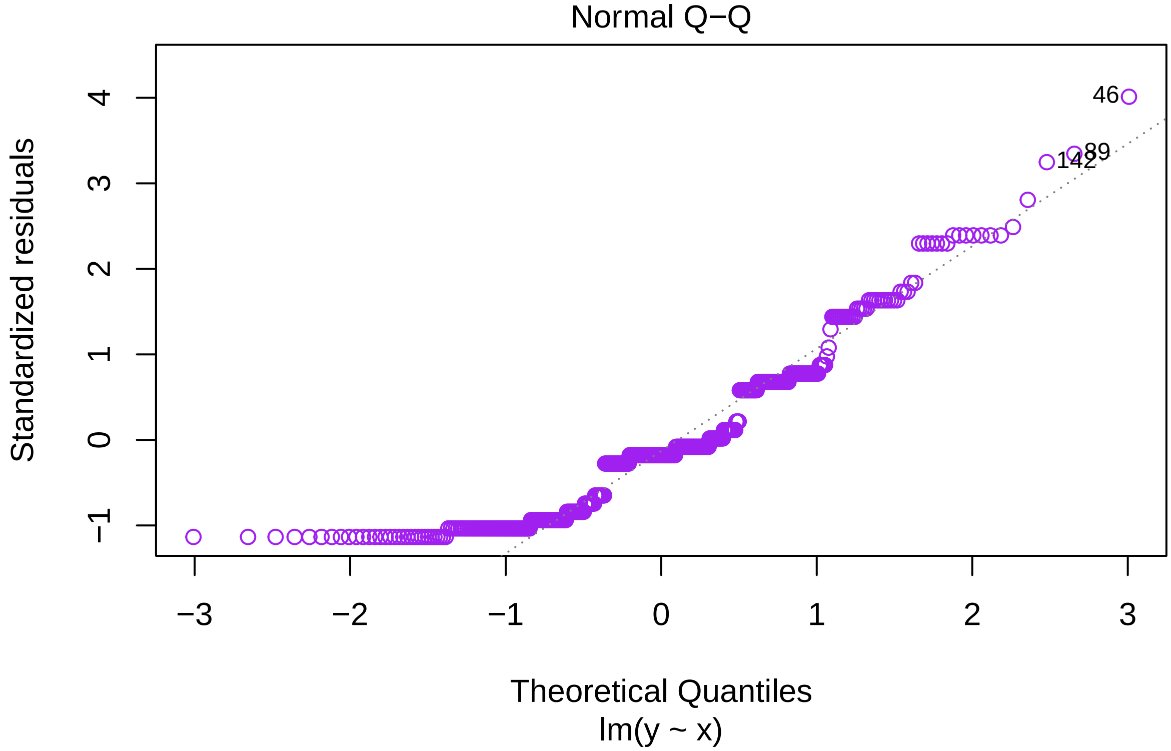
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3224 -1.0938 -0.2081  0.7919  4.6776
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.32237    0.09116  14.505   <2e-16 ***
## x           -0.11426    0.04482  -2.549   0.0112 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.169 on 378 degrees of freedom
## Multiple R-squared:  0.0169, Adjusted R-squared:  0.0143
## F-statistic: 6.499 on 1 and 378 DF,  p-value: 0.01119
```

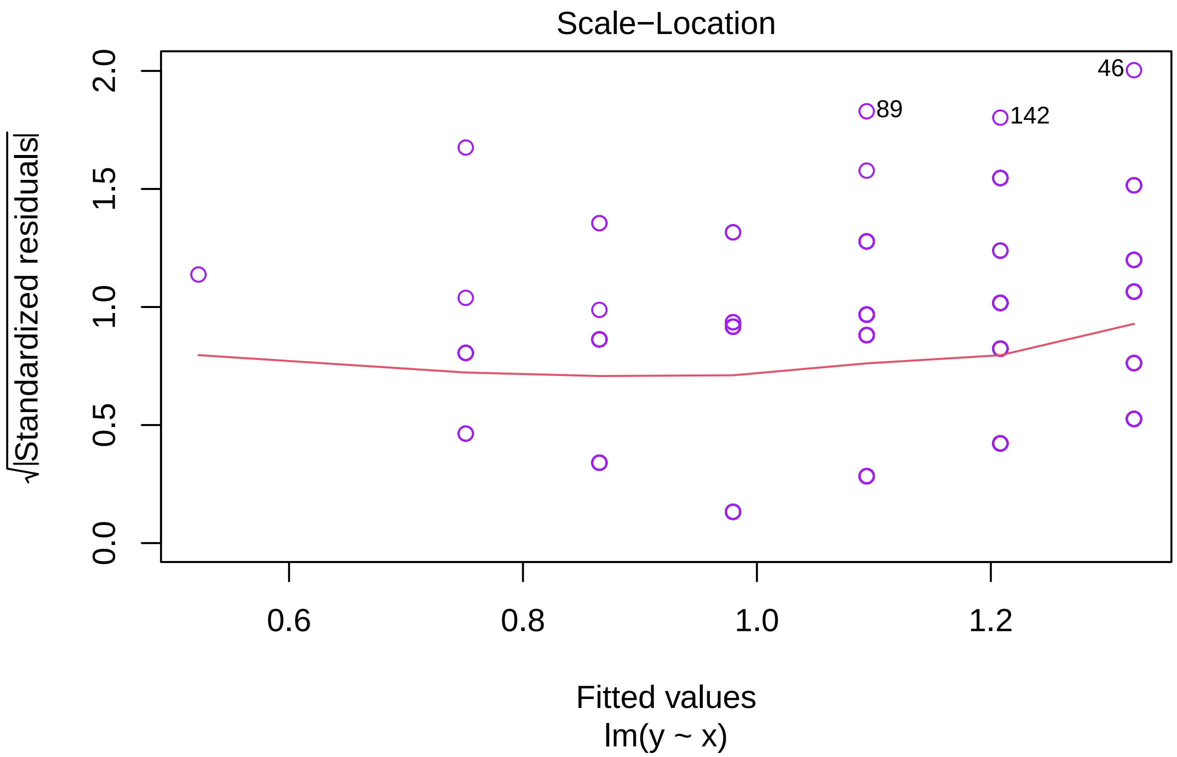
```
plot(x,y, col='brown')
abline(fit)
```

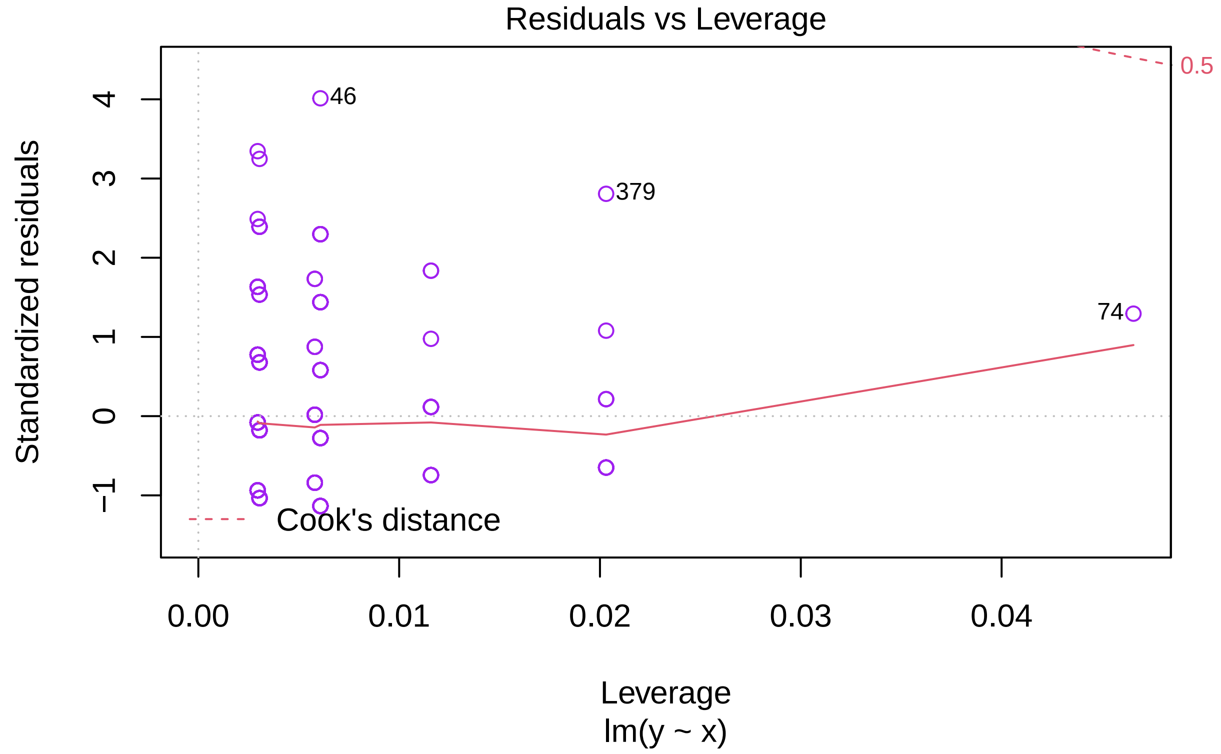


```
plot(fit, col="purple")
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
−2
−1
0
1
2
3
4
5
0.6
0.8
1.0
1.2
46
89
142


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
−1
0
1
2
3
4
−3
−2
−1
0
1
2
3
46
142
89

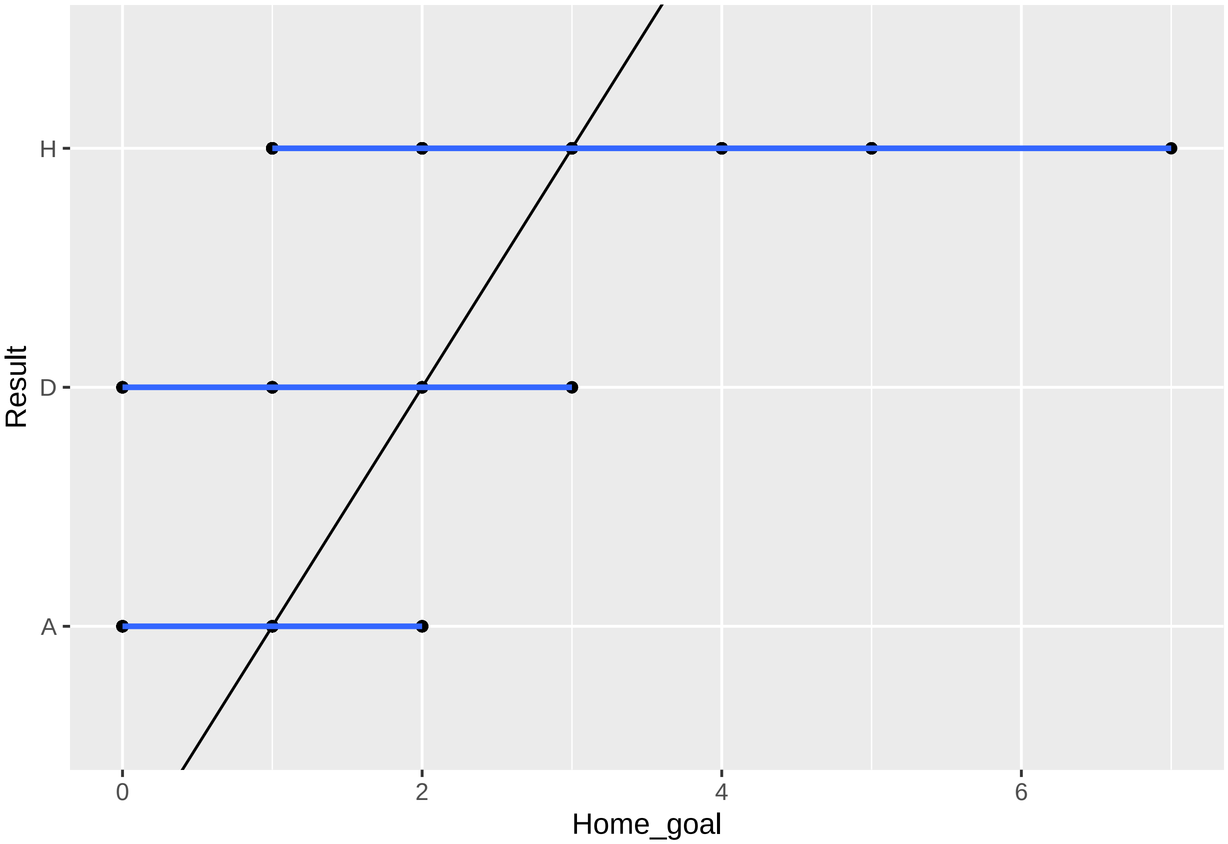




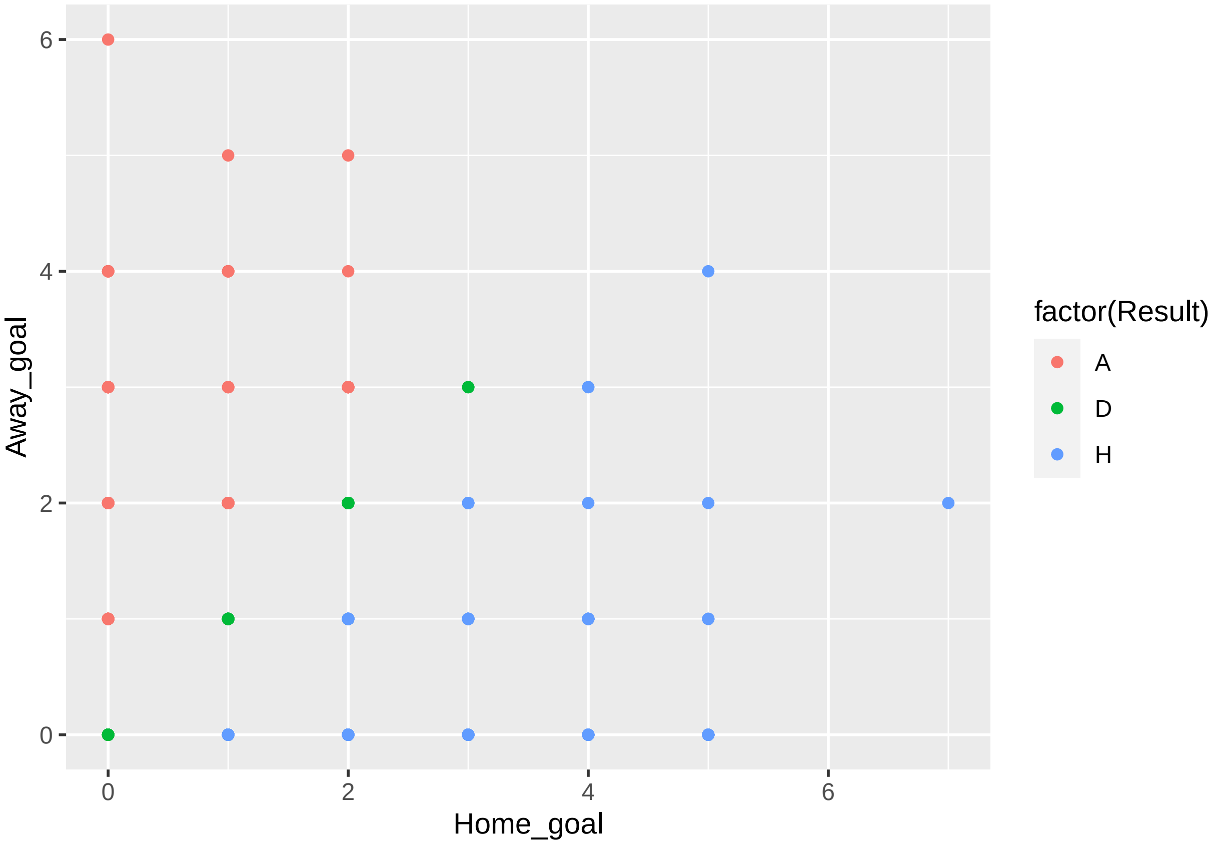
Finally, I'll conclude with a graph.

```
library('ggplot2')
ggplot(data = Results, aes(x = Home_goal, y = Result)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth(method = "lm", se = FALSE)
```

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



```
ggplot(data = Results, aes(x = Home_goal, y = Away_goal, color = factor(Result))) +
  geom_point()
```



REGRESSION ANALYSIS

From our regression analysis on the dataset of home goals and away goals, we got the negative regression value. This means that the chance of increase in one variable decrease the chance of another variable. We can relate this principle on our calculation. As we jump into our score dataset, we can see that there has been numbers of cases when the increase in probability of home goal has decreased the chance of away goal and away win and vice versa is true as well. From this analysis, I have found that the proposed regression rule for data mining can be effective to extract football tactics from the team's individual performance.

CONCLUSION

The fact that degree of freedom was negative indicates that there were more parameters in the model than we have rows of data or observations to train the model which absolutely makes sense. Though we calculated the regression value for two variables, there can be other parameters in a dataset, which can contribute to the change in calculation. Even in my dataset, I had non-numeric argument like "Result" which needed to be converted into vectors scaled values for calculation else it gets error or neglected.

Although the presented technique is not a sophisticated measure for establishing a general recommendation pattern in this dataset, it provides us with an underlying relationships between the teams and their brand value. Such approach can also be incorporated in many activities, for instance in players values in teams win or win rates.

———————————————————END———————————————————————