HW10

14.1

After reading the data, I looked at the summary statistics to find out that column 7 has 16 missing values. As a next step, I replaced all missing values with the mean of the values in that column. Similarly, I replaced the missing values with the mode of the values in that column.

As a next step in the analysis, I used Regression to determine the missing values. I fit the regression model using all the other columns and treated the column 7 as the response variable. I then determined the missing values using this regression model. As shown in the Q-Q plots, I don't think regression model is the best way to determine the missing values. This is also substantiated by a 69.9% adjusted R-squared.

I also created a copy of the original dataset and deleted all rows that have missing values.

I also added random noise to the predictions from regression.

Finally, I used all the dataset to classify using SVM and Knn classifiers and obtained the test accuracies.

If I look at the results, I see the best test accuracy of 98.5% for an SVM classifier, when I use the mean value for the missing values.

When I use mode the test accuracy drops to 96.6%.

When Regression is used the test accuracy drops to 95.2% and as expected with noise the accuracy is the lowest 94.2%.

When we drop the missing values the accuracy is 95.6%, this high accuracy even after dropping the missing values, could just be pure chance and could also be the cause of overfitting.

However, when I used Knn on the above-mentioned dataset, I observed the below test accuracies:

1) Mean: 57.6%
2) Mode: 57.1%
3) Regression: 58.0%
4) Regression with noise: 60.0%
5) Missing data deleted: 62.4%

15.1

I work for a major grocer and optimization can be used to determine the correct quantity of groceries that should be ordered to satisfy the customer demand. For example, the variables could be how many gallons of Coke, Pepsi should be ordered, the amount of chips, cookies to be ordered and so on. Overall, there could be thousands of products for which the order quantities need to be determined, so there could be that many variables in the model.

The constraints should be designed to make sure that the demand is met. In other words, the amount of grocery ordered for any item should be enough to meet the demand for that product. In addition, we could also specify the maximum and minimum quantities that should be ordered, i.e. we don't want to order way too many gallons of Pepsi only to have them not sold and sitting in the stores. Similarly, we don't want to order small quantities either, so that we end up reordering all the time. Moreover, we can specify constraints to make sure that the same type product is not ordered too much i.e. amount Pepsi, Coke and other beverages ordered should not exceed a certain limit.

The objective function could be to minimize the cost. Basically, as mentioned above there could be a penalty if we order too few or too many quantities. Also, there could be costs associated with every order and so on. Therefore, the objective is to find the optimal quantity to be ordered for each item such that the overall cost is minimized. Time is also a factor as we want the supplies delivered and be available on time before any stock outs. Therefore, we can also determine the cost for late and speed deliveries and incorporate that in the objective function.

```r
library(kernlab)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha
```

```r
library(class)
cancer_data = read.table("breast-cancer-wisconsin.data.txt", header=FALSE, sep=",", na.strings="NA")

cancer_data[] = lapply(cancer_data, as.numeric)
```

```
## Warning in lapply(cancer_data, as.numeric): NAs introduced by coercion
```

```r
#View(cancer_data)
print("Summary of the original data")
```

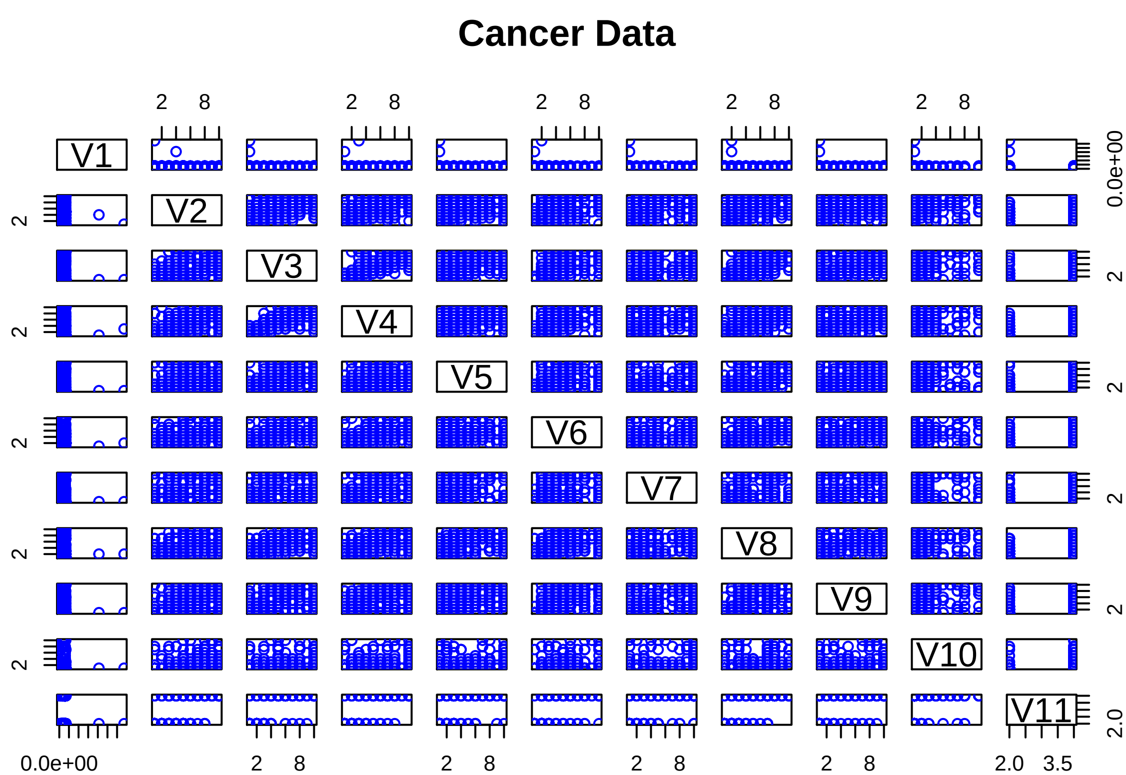
```
## [1] "Summary of the original data"
```

```r
summary(cancer_data)
```

```
##        V1                  V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.545   Mean   : 3.438
```

```
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##                                    NA's   :16
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
##
```

```
plot(cancer_data, col="blue", main="Cancer Data")
```




```
## Mean
library(na.tools)
cancer_data_deleted = cancer_data[is.na(cancer_data[,7])==F,]

cancer_data_mean = cancer_data
cancer_data_mode = cancer_data
cancer_data_noise = cancer_data

## Mean
cancer_data_mean[is.na(cancer_data_mean[,7]),7] = mean(cancer_data[,7], na.rm=TRUE)
print("Summary after imputing the NA's with mean of the values in that column")
```

```
## [1] "Summary after imputing the NA’s with mean of the values in that column"
```

```r
summary(cancer_data_mean)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.545   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```r
temp = na.mode(cancer_data[,7])

cancer_data_mode[,7] = temp
print("Summary of data after imputing the NA’s with mode of the values in that column")
```

```
## [1] "Summary of data after imputing the NA’s with mode of the values in that column"
```

```r
summary(cancer_data_mode)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.486   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
```

```
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
print("Summary of data after deleting the rows with NAs")
```

```
## [1] "Summary of data after deleting the rows with NAs"
```

```
summary(cancer_data_deleted)
```

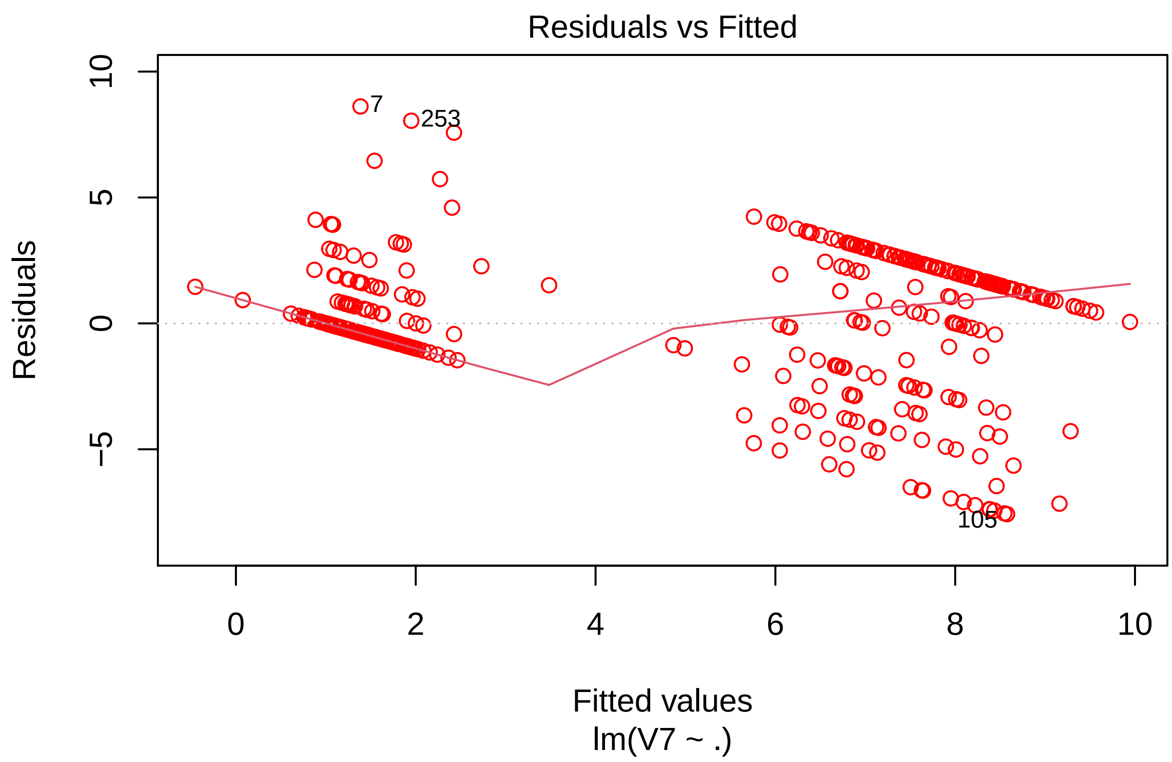
```
##        V1                  V2               V3               V4
##  Min.   :   63375   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  877617   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171795   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1076720   Mean   : 4.442   Mean   : 3.151   Mean   : 3.215
##  3rd Qu.: 1238705   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5              V6               V7               V8
##  Min.   : 1.00   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.00   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.00   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.83   Mean   : 3.234   Mean   : 3.545   Mean   : 3.445
##  3rd Qu.: 4.00   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.00   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9             V10              V11
##  Min.   : 1.00   Min.   : 1.000   Min.   :2.0
##  1st Qu.: 1.00   1st Qu.: 1.000   1st Qu.:2.0
##  Median : 1.00   Median : 1.000   Median :2.0
##  Mean   : 2.87   Mean   : 1.603   Mean   :2.7
##  3rd Qu.: 4.00   3rd Qu.: 1.000   3rd Qu.:4.0
##  Max.   :10.00   Max.   :10.000   Max.   :4.0
```

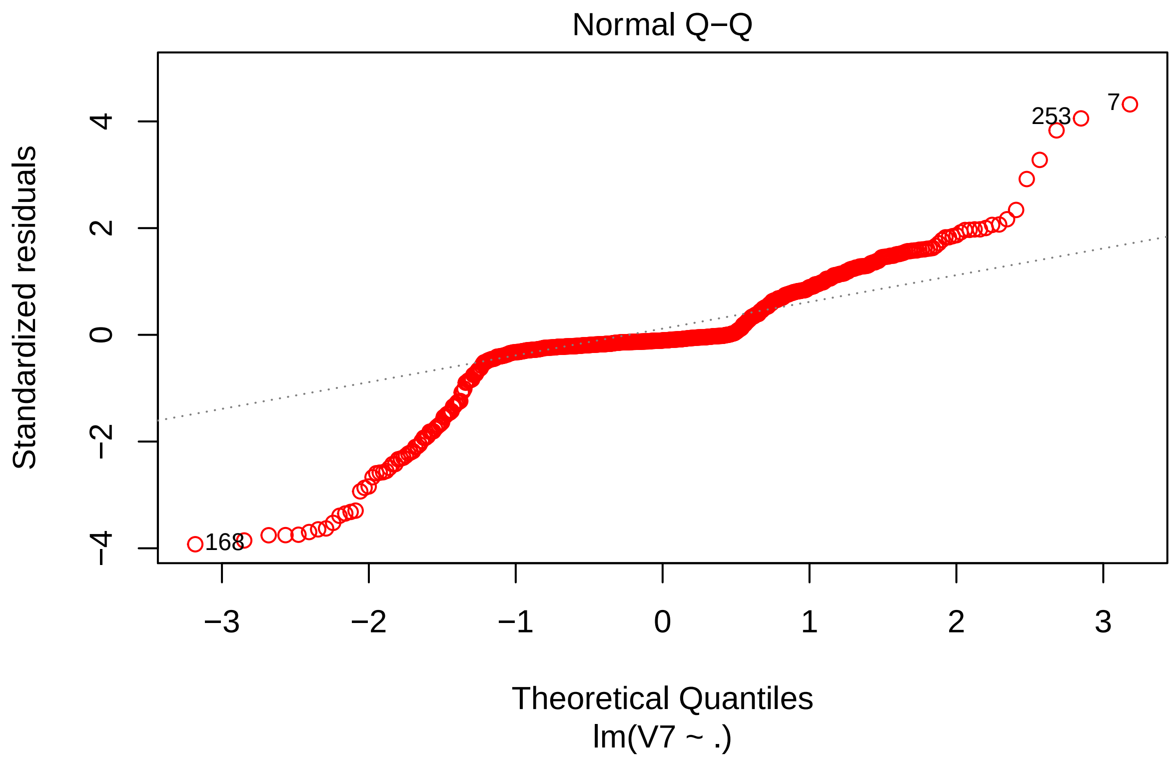
```
###Regression
set.seed(10)
mod = lm(V7 ~., data=cancer_data)
summary(mod)
```

```
##
## Call:
## lm(formula = V7 ~ ., data = cancer_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5771 -0.4427 -0.2088  0.8940  8.6145
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.039e+00  3.487e-01 -11.582  < 2e-16 ***
## V1          -1.656e-07  1.240e-07  -1.335  0.18223
## V2           1.825e-02  3.960e-02   0.461  0.64499
## V3          -1.594e-01  6.731e-02  -2.369  0.01813 *
```

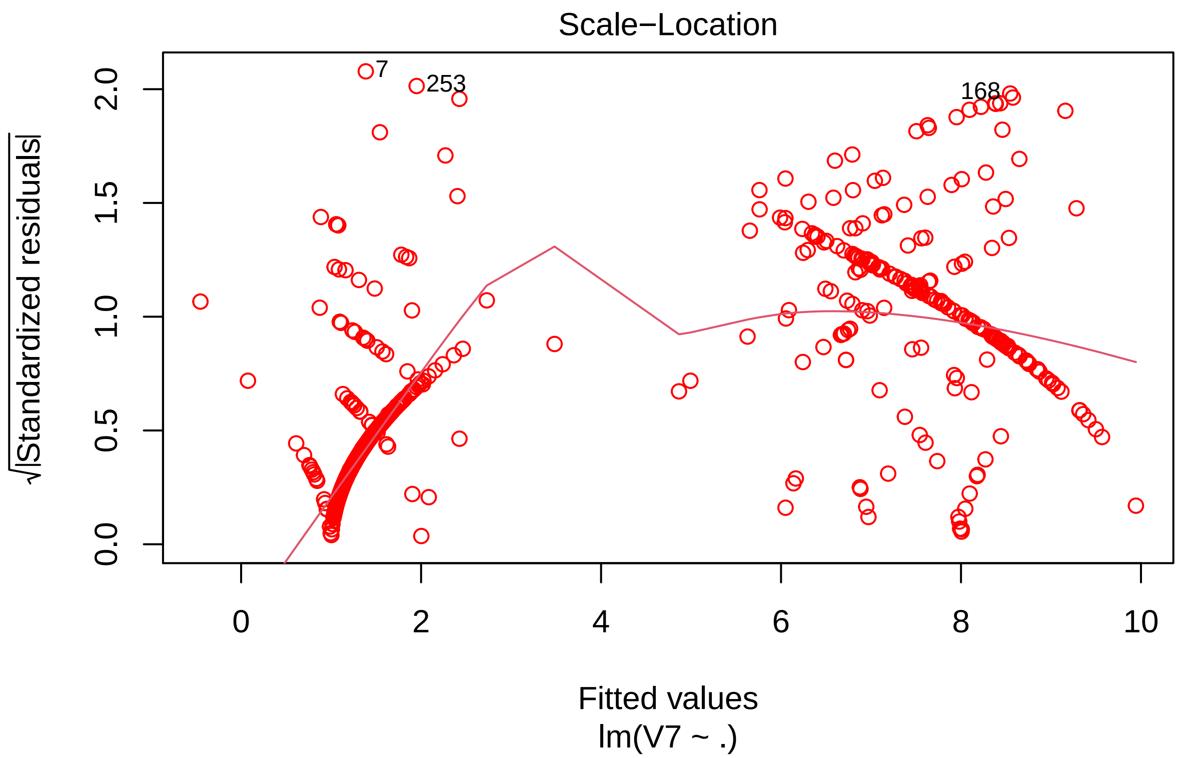
```
## V4           1.863e-01  6.548e-02   2.844  0.00459 **
## V5           2.194e-01  4.124e-02   5.320 1.42e-07 ***
## V6           1.872e-02  5.520e-02   0.339  0.73457
## V8           1.505e-01  5.327e-02   2.825  0.00487 **
## V9          -8.724e-02  3.967e-02  -2.199  0.02821 *
## V10         -6.365e-02  5.215e-02  -1.220  0.22272
## V11          2.495e+00  1.784e-01  13.990  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.999 on 672 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.7035, Adjusted R-squared:  0.6991
## F-statistic: 159.4 on 10 and 672 DF,  p-value: < 2.2e-16
```

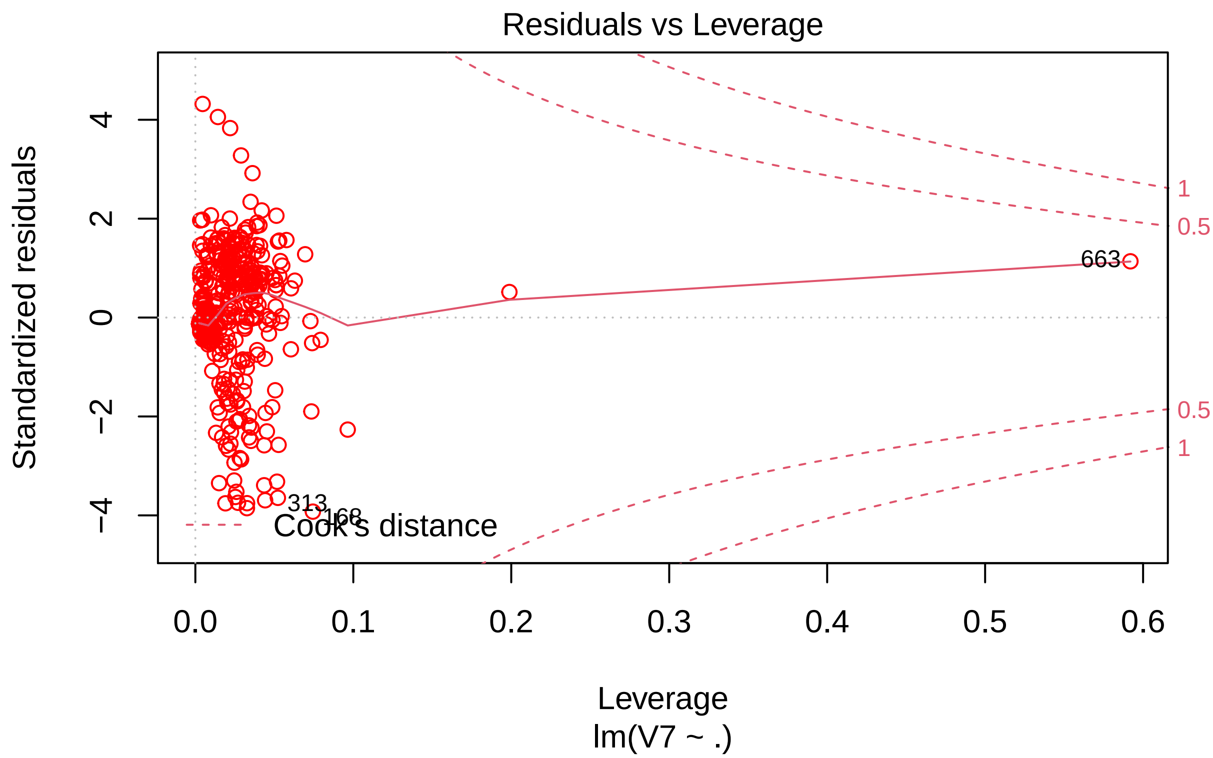
```r
plot(mod, col="Red")
```





Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(V7 ~ .)
253
7
168
−4
−2
0
2
4
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
7
253
168
0
2
4
6
8
10
Fitted values
lm(V7 ~ .)




```r
r_test_data = cancer_data[is.na(cancer_data[,7]),]
r_test_data2 = r_test_data[,c(1,2,3,4,5,6,8,9,10,11)]


cancer_data[is.na(cancer_data[,7]),7] = predict(mod,r_test_data2)
print("Summary of data after imputing data using Regression")
```

```
## [1] "Summary of data after imputing data using Regression"
```

```r
summary(cancer_data)
```

```
##        V1                  V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.515   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
```

```
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```r
#####Regression with pertrubtaion

noise = runif(nrow(r_test_data2), min=1, max=10)


cancer_data_noise[is.na(cancer_data_noise[,7]),7] = predict(mod, r_test_data2) + noise
print("Summary of data after adding noise to the regression predictions")
```

```
## [1] "Summary of data after adding noise to the regression predictions"
```

```r
summary(cancer_data_noise)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.622   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 7.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :12.759   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```r
var=""
for(i in 1:5)
  {
    data = data.frame()
    if (i == 1)
    {
      data = cancer_data_mean
      var = "using mean"
```

```
    }
    else if (i == 2)
    {
      data = cancer_data_mode
      var = "using mode"

    }
    else if (i == 3)
    {
      data = data.frame(cancer_data)
      var = "using Regression"
    }
    else if ( i == 4)
    {
      data = cancer_data_noise
      var = "using Regression and noise"
    }
    else
    {
      data = cancer_data_deleted
      var = "where missing data is deleted"
    }

    samples = sort(sample(nrow(data), nrow(data)*0.70))

    train_data = data[samples,]
    test_data = data[-samples,]
    model = ksvm(as.matrix(train_data[,1:10]), as.factor(train_data[,11]), type="C-svc", kernel="vanill

    pred = predict(model, test_data[,1:10])

    cat(" SVM test accuracy in case of", var, "is ", sum(pred == test_data[,11])/nrow(test_data), "\n")
    cat("\n")


    model = knn(as.matrix(train_data[,1:10]), test_data[,1:10], as.factor(train_data[,11]), k=6)
    tab = table(model,test_data[,11])
    print("Confusion Matrix")
    print(tab)
    accuracy = sum(diag(tab))*1.0/(nrow(test_data))
    cat("\n")
    cat(" Knn test accuracy in case of", var, "is ", accuracy, "\n")
}
```

```
##  Setting default kernel parameters
##  SVM test accuracy in case of using mean is  0.9571429
##
## [1] "Confusion Matrix"
##
## model   2   4
##     2 110  43
##     4  35  22
##
```

```
##  Knn test accuracy in case of using mean is  0.6285714
##  Setting default kernel parameters
##  SVM test accuracy in case of using mode is  0.9714286
##
## [1] "Confusion Matrix"
##
## model   2   4
##     2 111  61
##     4  25  13
##
##  Knn test accuracy in case of using mode is  0.5904762
##  Setting default kernel parameters
##  SVM test accuracy in case of using Regression is  0.9666667
##
## [1] "Confusion Matrix"
##
## model   2   4
##     2 105  61
##     4  24  20
##
##  Knn test accuracy in case of using Regression is  0.5952381
##  Setting default kernel parameters
##  SVM test accuracy in case of using Regression and noise is  0.9761905
##
## [1] "Confusion Matrix"
##
## model   2   4
##     2 108  48
##     4  31  23
##
##  Knn test accuracy in case of using Regression and noise is  0.6238095
##  Setting default kernel parameters
##  SVM test accuracy in case of where missing data is deleted is  0.9609756
##
## [1] "Confusion Matrix"
##
## model   2   4
##     2 103  50
##     4  26  26
##
##  Knn test accuracy in case of where missing data is deleted is  0.6292683
```