ISL-PROJECT

Name: Abhinayreddy Polimera
ID: 1633633

PART-1: REGRESSION

A) Linear Regression

winequality_red

Filter

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	de
1	7.4	0.700	0.00	1.90	0.076	11	34	
2	7.8	0.880	0.00	2.60	0.098	25	67	
3	7.8	0.760	0.04	2.30	0.092	15	54	
4	11.2	0.280	0.56	1.90	0.075	17	60	
5	7.4	0.700	0.00	1.90	0.076	11	34	
6	7.4	0.660	0.00	1.80	0.075	13	40	
7	7.9	0.600	0.06	1.60	0.069	15	59	
8	7.3	0.650	0.00	1.20	0.065	15	21	
9	7.8	0.580	0.02	2.00	0.073	9	18	
10	7.5	0.500	0.36	6.10	0.071	17	102	
11	6.7	0.580	0.08	1.80	0.097	15	65	
12	7.5	0.500	0.36	6.10	0.071	17	102	
13	5.6	0.615	0.00	1.60	0.089	16	59	
14	7.8	0.610	0.29	1.60	0.114	9	29	

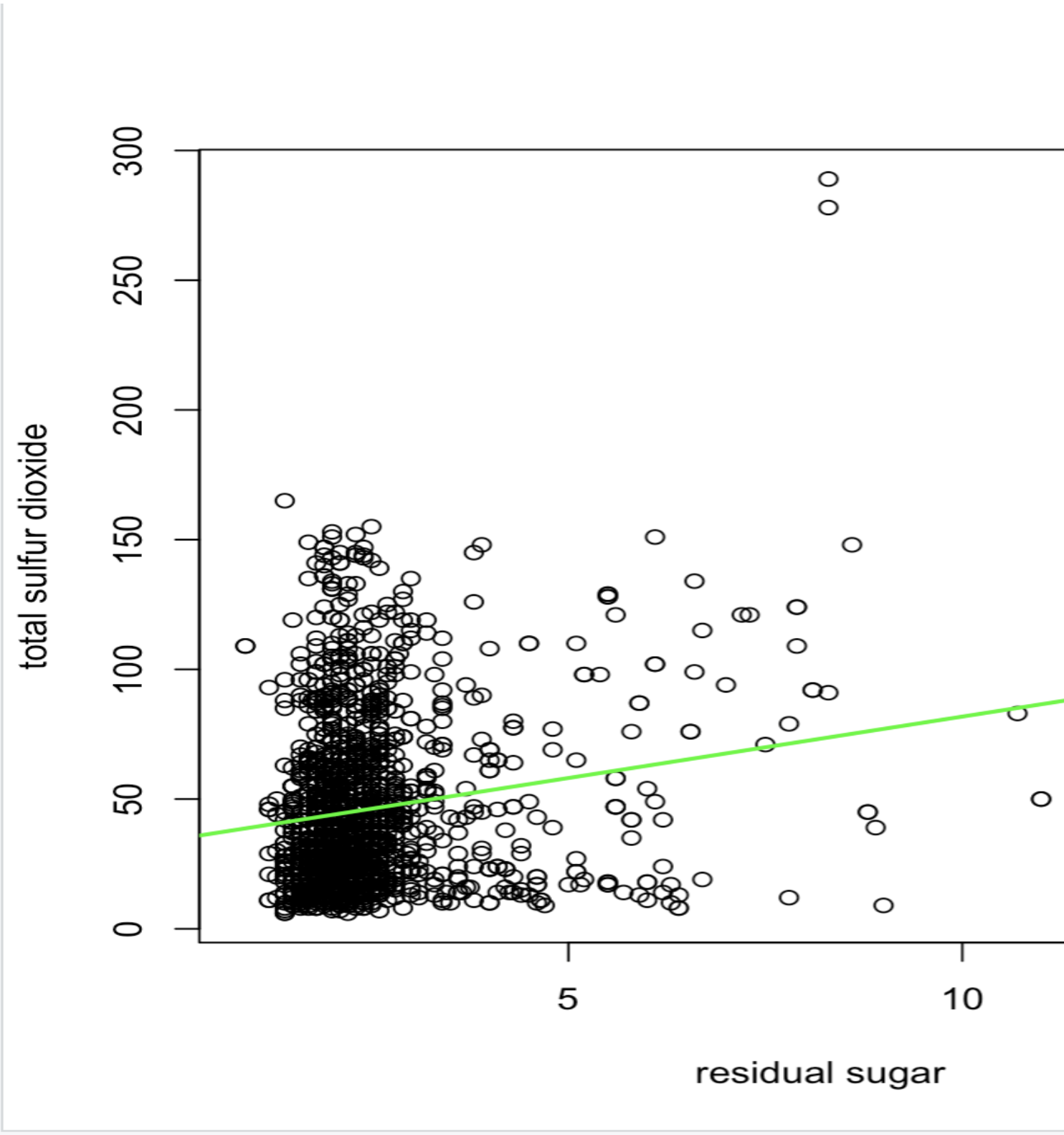
Showing 1 to 14 of 1,599 entries, 12 total columns

```
> library(readr)
> winequality_red <- read_csv("Downloads/winequality-red.csv")
Rows: 1599 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlor

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this mess
> View(winequality_red)
> library(MASS)
> library(ISLR2)

Attaching package: 'ISLR2'

The following object is masked from 'package:MASS':

    Boston

> head(winequality_red)
# A tibble: 6 × 12
  `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
            <dbl>              <dbl>         <dbl>             <dbl>     <dbl>
1             7.4                0.7             0               1.9     0.076
2             7.8                0.88            0               2.6     0.098
3             7.8                0.76            0.04            2.3     0.092
4            11.2                0.28            0.56            1.9     0.075
5             7.4                0.7             0               1.9     0.076
6             7.4                0.66            0               1.8     0.075
# ℹ 6 more variables: `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>,
#   alcohol <dbl>, quality <dbl>
>
```

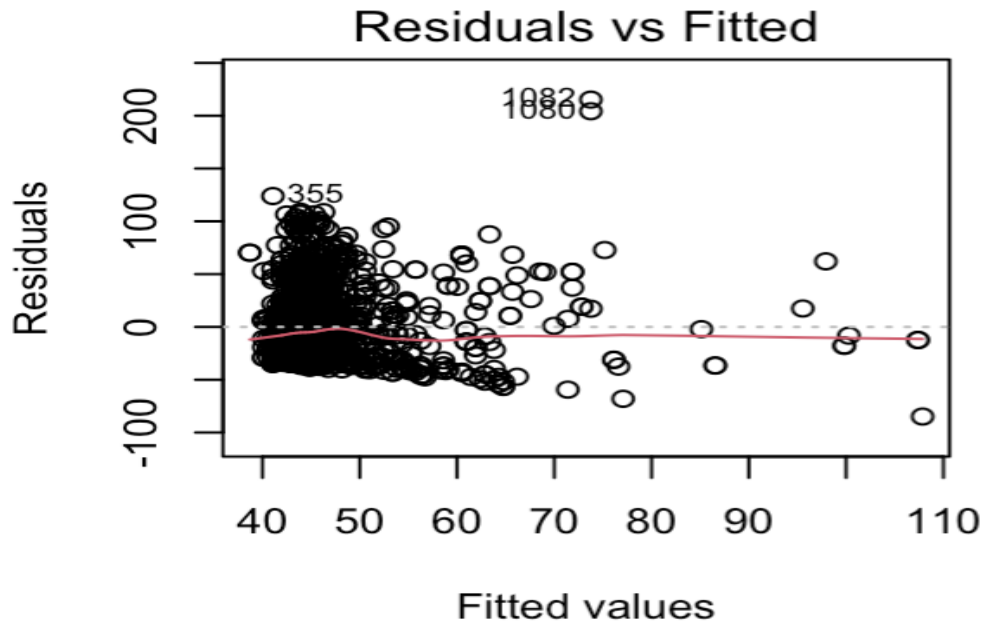
```
> attach(winequality_red)
> lm.fit <- lm(`total sulfur dioxide` ~ `residual sugar`, data = winequality_
> lm.fit <- lm(`total sulfur dioxide` ~ `residual sugar`)
> lm.fit

Call:
lm(formula = `total sulfur dioxide` ~ `residual sugar`)

Coefficients:
     (Intercept)  `residual sugar`
          34.442             4.737

> summary(lm.fit)

Call:
lm(formula = `total sulfur dioxide` ~ `residual sugar`)

Residuals:
    Min      1Q  Median      3Q     Max
-84.863 -23.021  -8.337  16.032 215.242

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       34.4418     1.6600  20.748  < 2e-16 ***
`residual sugar`   4.7369     0.5717   8.286 2.45e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 32.22 on 1597 degrees of freedom
Multiple R-squared:  0.04122,   Adjusted R-squared:  0.04062
F-statistic: 68.66 on 1 and 1597 DF,  p-value: 2.449e-16
```

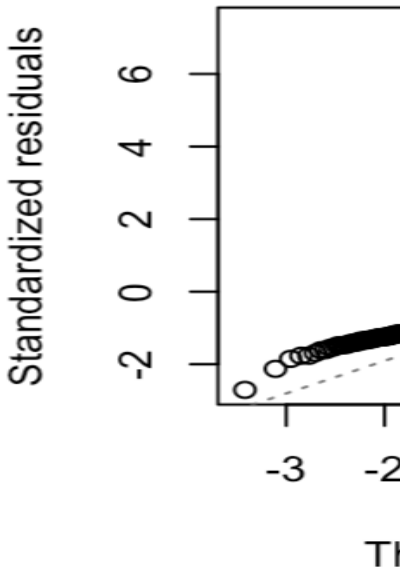
```
> names(lm.fit)
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.
 [7] "qr"            "df.residual"   "xlevels"       "call"          "terms"
> coef(lm.fit)
     (Intercept) `residual sugar`
       34.441761         4.736886
> confint(lm.fit)
                     2.5 %    97.5 %
(Intercept)      31.185706 37.697815
`residual sugar`  3.615586  5.858185
> predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
+         interval = "confidence")
           fit      lwr       upr
1     43.44184 41.70664  45.17704
2     46.75766 45.17572  48.33961
3     45.33660 43.73362  46.93958
4     43.44184 41.70664  45.17704
5     43.44184 41.70664  45.17704
6     42.96816 41.18374  44.75257
7     42.02078 40.12184  43.91972
8     40.12602 37.94624  42.30581
9     43.91553 42.22353  45.60753
10    63.33676 59.04221  67.63132
```

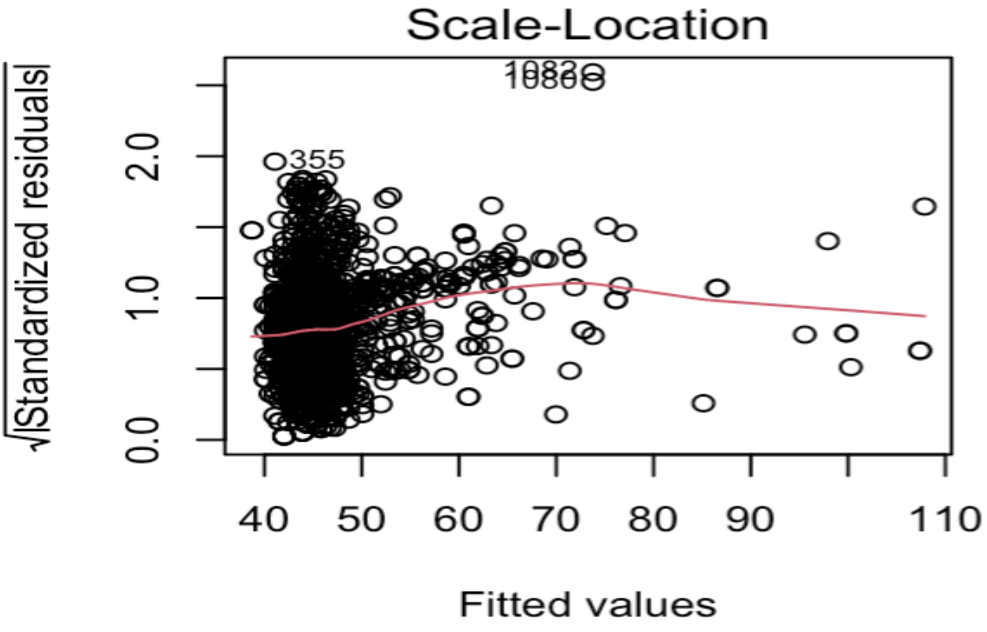
```
> predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
+         interval = "prediction")
           fit           lwr      upr
1     43.44184 -19.780488448 106.6642
2     46.75766 -16.460647875 109.9760
3     45.33660 -17.882243440 108.5554
4     43.44184 -19.780488448 106.6642
5     43.44184 -19.780488448 106.6642
6     42.96816 -20.255546839 106.1919
7     42.02078 -21.206260107 105.2478
8     40.12602 -23.110072098 103.3621
9     43.91553 -19.305628902 107.1367
10    63.33676  -0.007498758 126.6810
11    42.96816 -20.255546839 106.1919
12    63.33676  -0.007498758 126.6810
13    42.02078 -21.206260107 105.2478
14    42.02078 -21.206260107 105.2478
15    52.44193 -10.792163213 115.6760
16    52.91562 -10.321081698 116.1523
17    42.96816 -20.255546839 106.1919
```

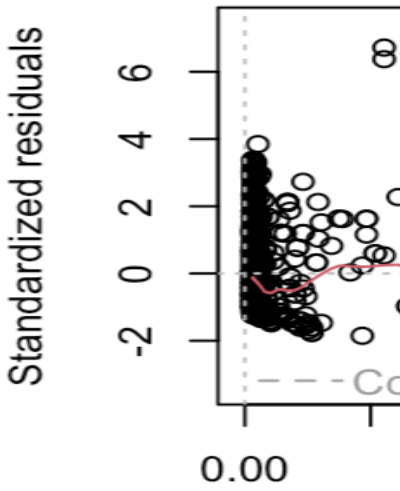

total sulfur dioxide
0
50
100
150
200
250
300
5
10
residual sugar

```
> plot(`residual sugar`, `total sulfur dioxide`)
> abline(lm.fit, lwd = 2, col = "green")
>
> par(mfrow = c(2, 2))
> plot(lm.fit)
>
```


Residuals vs Fitted
Residuals
Fitted values
355
1082
1086
-100
0
100
200
40
50
60
70
80
90
110


Standardized residuals
-2
0
2
4
6
-3


Scale-Location
√|Standardized residuals|
Fitted values
355
1082
1086
0.0
1.0
2.0
40
50
60
70
80
90
110


Standardized residuals
-2
0
2
4
6
0.00

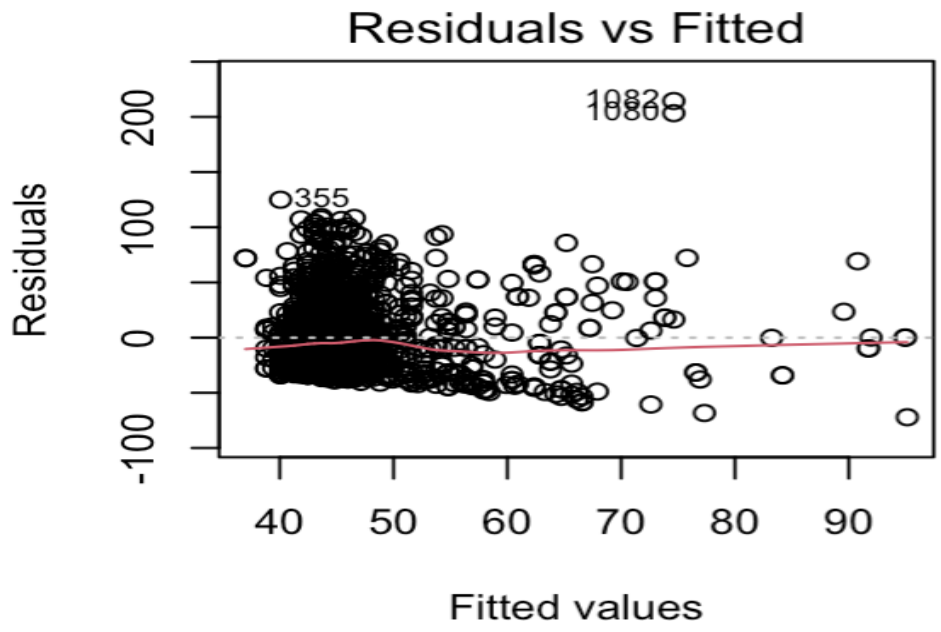
```
> lm.fit <- lm(`total sulfur dioxide` ~ ., data = winequality_red)
> summary(lm.fit)

Call:
lm(formula = `total sulfur dioxide` ~ ., data = winequality_red)

Residuals:
    Min      1Q  Median      3Q     Max
-55.355 -13.868  -4.457   8.186 177.013

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)          -1.969e+03  7.241e+02  -2.719  0.00663 **
`fixed acidity`      -8.052e+00  8.652e-01  -9.307  < 2e-16 ***
`volatile acidity`    3.099e+01  4.177e+00   7.420 1.90e-13 ***
`citric acid`         5.215e+01  4.867e+00  10.714  < 2e-16 ***
`residual sugar`      8.005e-01  5.133e-01   1.559  0.11912
chlorides            -8.682e+01  1.428e+01  -6.081 1.50e-09 ***
`free sulfur dioxide` 1.967e+00  5.567e-02  35.335  < 2e-16 ***
density               2.241e+03  7.385e+02   3.035  0.00245 **
pH                   -5.193e+01  6.438e+00  -8.066 1.42e-15 ***
sulphates             8.628e+00  3.986e+00   2.164  0.03059 *
alcohol              -1.938e+00  9.358e-01  -2.071  0.03855 *
quality              -3.825e+00  8.539e-01  -4.480 8.00e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.18 on 1587 degrees of freedom
Multiple R-squared:  0.5484,    Adjusted R-squared:  0.5453
F-statistic: 175.2 on 11 and 1587 DF,  p-value: < 2.2e-16
```

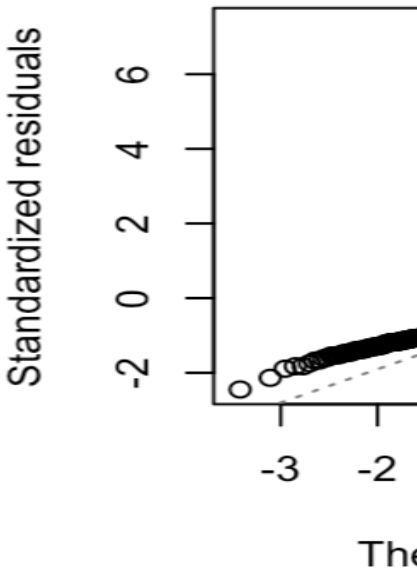
```
> lm.fit1 <- lm(`total sulfur dioxide` ~ . - sulphates, data = winequality_
> summary(lm.fit)

Call:
lm(formula = `total sulfur dioxide` ~ ., data = winequality_red)

Residuals:
    Min      1Q  Median      3Q     Max
-55.355 -13.868  -4.457   8.186 177.013

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)          -1.969e+03  7.241e+02  -2.719  0.00663 **
`fixed acidity`      -8.052e+00  8.652e-01  -9.307  < 2e-16 ***
`volatile acidity`    3.099e+01  4.177e+00   7.420 1.90e-13 ***
`citric acid`         5.215e+01  4.867e+00  10.714  < 2e-16 ***
`residual sugar`      8.005e-01  5.133e-01   1.559  0.11912
chlorides            -8.682e+01  1.428e+01  -6.081 1.50e-09 ***
`free sulfur dioxide` 1.967e+00  5.567e-02  35.335  < 2e-16 ***
density               2.241e+03  7.385e+02   3.035  0.00245 **
pH                   -5.193e+01  6.438e+00  -8.066 1.42e-15 ***
sulphates             8.628e+00  3.986e+00   2.164  0.03059 *
alcohol              -1.938e+00  9.358e-01  -2.071  0.03855 *
quality              -3.825e+00  8.539e-01  -4.480 8.00e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.18 on 1587 degrees of freedom
Multiple R-squared:  0.5484,    Adjusted R-squared:  0.5453
F-statistic: 175.2 on 11 and 1587 DF,  p-value: < 2.2e-16
```

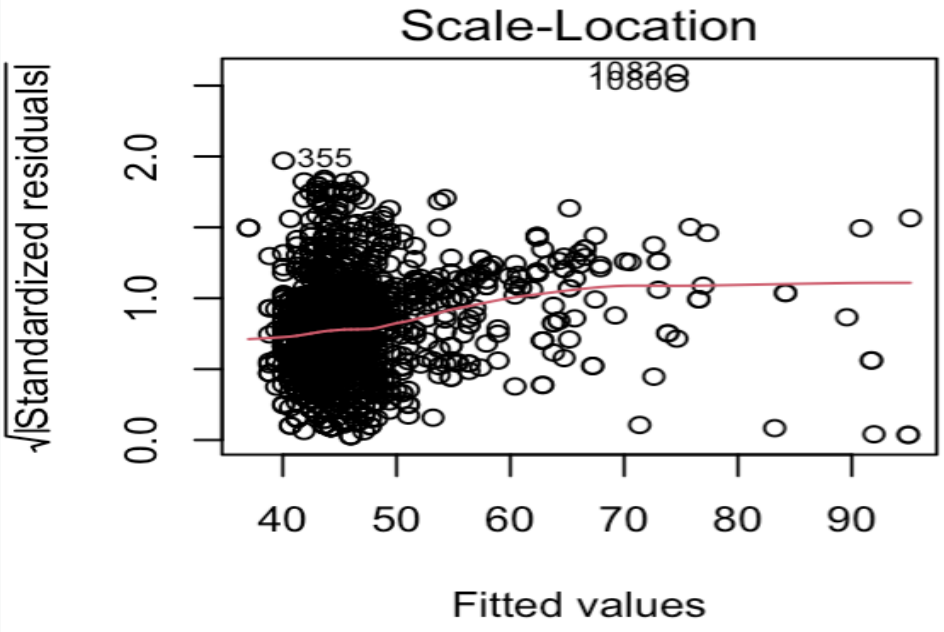
B) Multi-Linear Regression

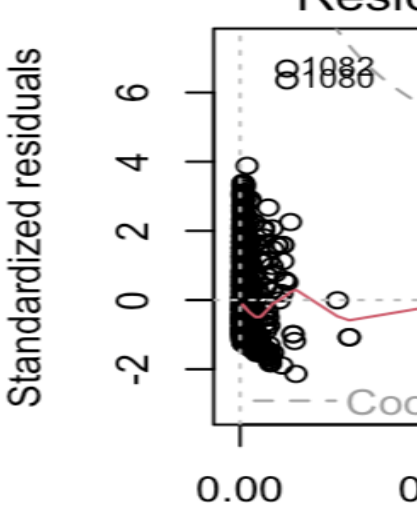
```
> summary(lm(`total sulfur dioxide` ~ `residual sugar` * sulphates, data =

Call:
lm(formula = `total sulfur dioxide` ~ `residual sugar` * sulphates,
    data = winequality_red)

Residuals:
    Min      1Q  Median      3Q     Max
-84.594 -23.360  -7.872  15.719 209.351

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                  16.314      7.419   2.199 0.028017 *
`residual sugar`             10.087      2.790   3.616 0.000308 ***
sulphates                    27.588     11.004   2.507 0.012270 *
`residual sugar`:sulphates   -8.141      4.151  -1.961 0.049996 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 32.17 on 1595 degrees of freedom
Multiple R-squared:  0.04527,   Adjusted R-squared:  0.04348
F-statistic: 25.21 on 3 and 1595 DF,  p-value: 6.178e-16

> lm.fit1 <- lm(`total sulfur dioxide` ~ `residual sugar` + I(`residual sug
> summary(lm.fit1)

Call:
lm(formula = `total sulfur dioxide` ~ `residual sugar` + I(`residual sugar`

Residuals:
    Min      1Q  Median      3Q     Max
-72.136 -23.048  -8.398  15.879 214.380

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            31.2739     3.1361   9.972  < 2e-16 ***
`residual sugar`        6.4931     1.5820   4.104 4.26e-05 ***
I(`residual sugar`^2)  -0.1531     0.1286  -1.191    0.234
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> anova(lm.fit, lm.fit2)
Error in anova.lm(lm.fit, lm.fit2) : object 'lm.fit2' not found
> anova(lm.fit, lm.fit1)
Analysis of Variance Table

Model 1: `total sulfur dioxide` ~ `fixed acidity` + `volatile acidity` +
    `citric acid` + `residual sugar` + chlorides + `free sulfur dioxide` +
    density + pH + sulphates + alcohol + quality
Model 2: `total sulfur dioxide` ~ `residual sugar` + I(`residual sugar`^2)
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1   1587  780865
2   1596 1656450 -9   -875585 197.72 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> par(mfrow = c(2, 2))
> plot(lm.fit1)
>
```


Residuals vs Fitted
Residuals
Fitted values
355
1082
1086
-100
0
100
200
40
50
60
70
80
90

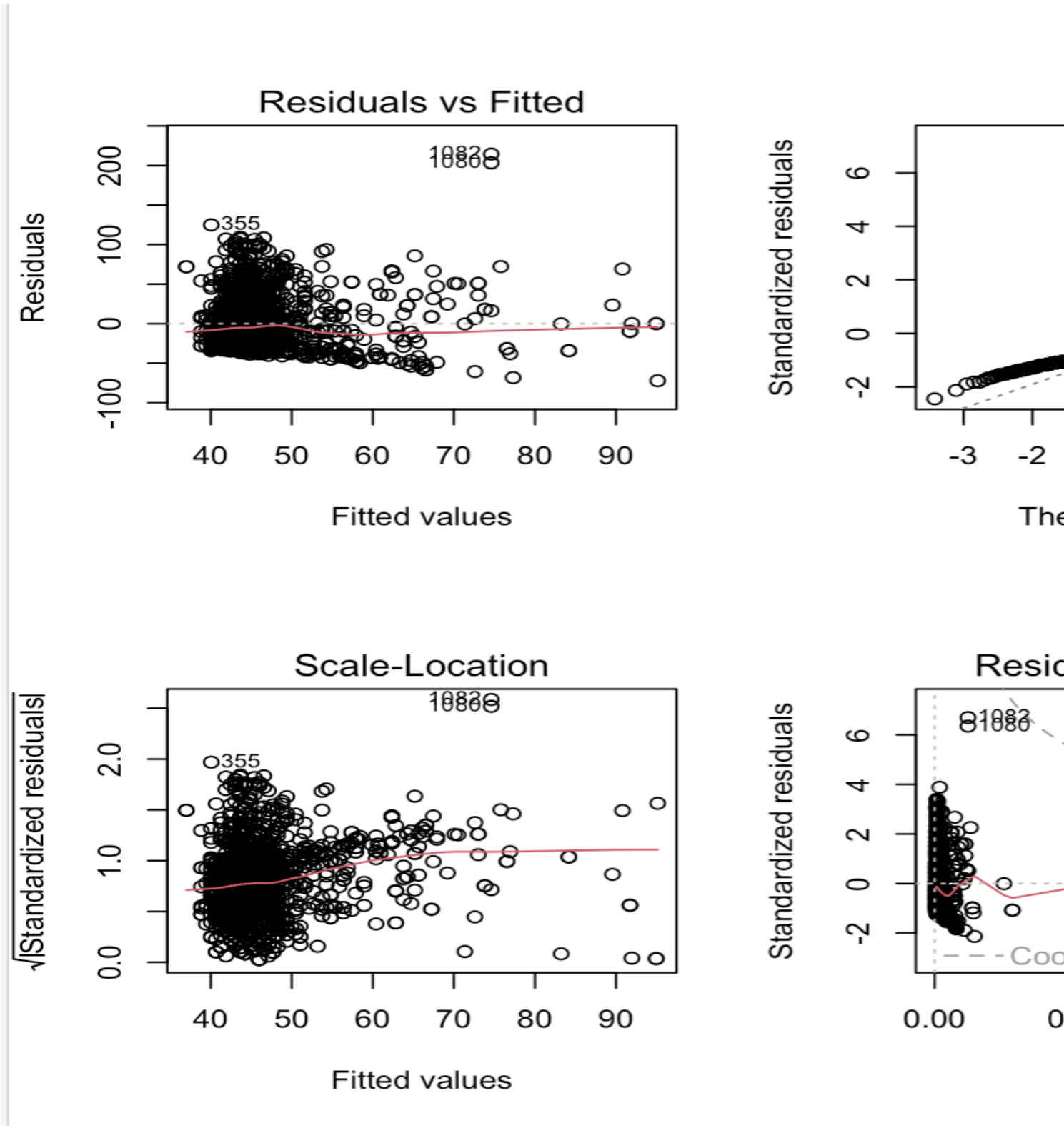

Standardized residuals
-2
0
2
4
6
-3
-2


Scale-Location
√|Standardized residuals|
Fitted values
355
1082
1086
0.0
1.0
2.0
40
50
60
70
80
90


Standardized residuals
1082
1086
-2
0
2
4
6
0.00

```
> summary(lm(`total sulfur dioxide` ~ log(`volatile acidity`),

Call:
lm(formula = `total sulfur dioxide` ~ log(`volatile acidity`),
    data = winequality_red)

Residuals:
    Min      1Q  Median      3Q     Max
-47.446 -24.117  -8.779  15.330 246.895

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)                52.498      1.814  28.936  < 2e-16 *
log(`volatile acidity`)     8.632      2.317   3.725 0.000202 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 32.76 on 1597 degrees of freedom
Multiple R-squared:  0.008614,  Adjusted R-squared:  0.007993
F-statistic: 13.88 on 1 and 1597 DF,  p-value: 0.000202
```

```
> lm.fit5 <- lm(`total sulfur dioxide` ~ poly(`residual suga
> summary(lm.fit5)

Call:
lm(formula = `total sulfur dioxide` ~ poly(`residual sugar`,
    5))

Residuals:
    Min      1Q  Median      3Q     Max
-90.677 -23.027  -8.536  16.041 198.493

Coefficients:
                              Estimate Std. Error t value Pr(>|t
(Intercept)                    46.4678     0.8029  57.872   <2e
poly(`residual sugar`, 5)1 266.9797    32.1073   8.315   <2e
poly(`residual sugar`, 5)2 -38.3549    32.1073  -1.195   0.2
poly(`residual sugar`, 5)3 -51.2376    32.1073  -1.596   0.1
poly(`residual sugar`, 5)4 -74.9902    32.1073  -2.336   0.0
poly(`residual sugar`, 5)5  77.5085    32.1073   2.414   0.0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 32.11 on 1593 degrees of freedom
Multiple R-squared:  0.05032,   Adjusted R-squared:  0.04733
F-statistic: 16.88 on 5 and 1593 DF,  p-value: 2.748e-16
```



PART-2 FEATURE SELECTION/MODEL OPTIMIZATION METHODS

1) **Forward and Backward Stepwise Selection**:

```
> library(readr)
> Life_Expectancy_Data <- read_csv("Downloads/Life Expectancy Data.csv")
Rows: 2938 Columns: 22
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (2): Country, Status
dbl (20): Year, Life expectancy, Adult Mortality, infant deaths, Alcohol, per

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this mess
> View(Life_Expectancy_Data)
> library(ISLR2)
> names(Life_Expectancy_Data)
 [1] "Country"                        "Year"
 [3] "Status"                         "Life expectancy"
 [5] "Adult Mortality"                "infant deaths"
 [7] "Alcohol"                        "percentage expenditure"
 [9] "Hepatitis B"                    "Measles"
[11] "BMI"                            "under-five deaths"
[13] "Polio"                          "Total expenditure"
[15] "Diphtheria"                     "HIV/AIDS"
[17] "GDP"                            "Population"
[19] "thinness  1-19 years"           "thinness 5-9 years"
[21] "Income composition of resources" "Schooling"
> dim(Life_Expectancy_Data)
[1] 2938   22
>
```

```
> library(readr)
> heart_disease <- read_csv("Downloads/heart_disease.csv")
Rows: 909 Columns: 15
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (6): sex, dataset, cp, restecg, slope, thal
dbl (7): id, age, trestbps, chol, thalch, oldpeak, ca
lgl (2): fbs, exang

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this mes
> View(heart_disease)
> regfit.fwd <- regsubsets(heart_disease$chol ~ ., data = heart_disease, nvr
> summary(regfit.fwd)
Subset selection object
Call: regsubsets.formula(heart_disease$chol ~ ., data = heart_disease,
    nvmax = 19, method = "forward")
20 Variables  (and intercept)
                         Forced in Forced out
id                           FALSE      FALSE
age                          FALSE      FALSE
sexMale                      FALSE      FALSE
datasetHungary               FALSE      FALSE
datasetVA Long Beach         FALSE      FALSE
cpatypical angina            FALSE      FALSE
cpnon-anginal                FALSE      FALSE
cptypical angina             FALSE      FALSE
trestbps                     FALSE      FALSE
fbsTRUE                      FALSE      FALSE
restecgnormal                FALSE      FALSE
```

Filter

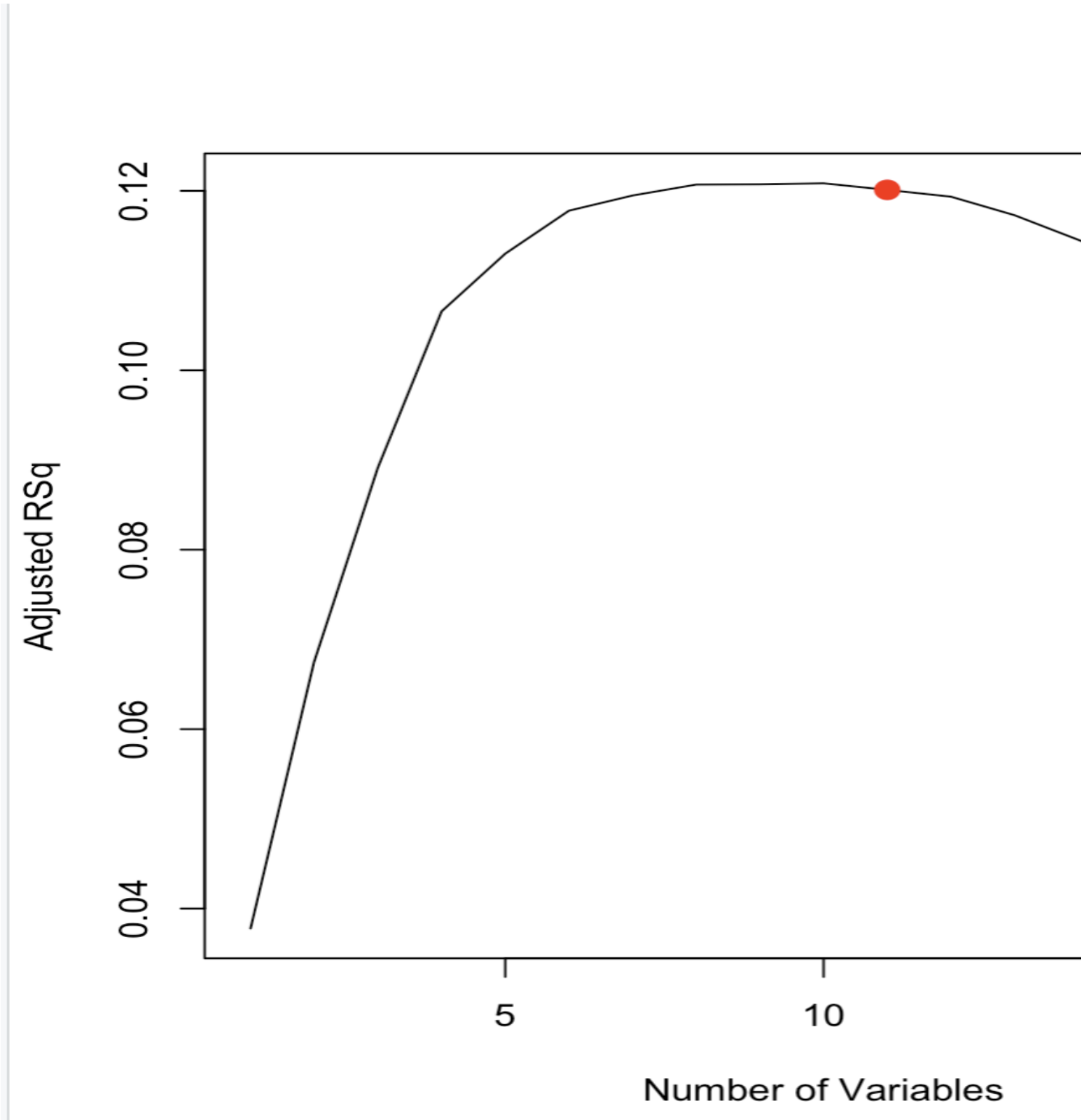
	id	age	sex	dataset	cp	trestbps	chol	fbs	reste
1	1	63	Male	Cleveland	typical angina	145	233	TRUE	lv hyp
2	2	67	Male	Cleveland	asymptomatic	160	286	FALSE	lv hyp
3	3	67	Male	Cleveland	asymptomatic	120	229	FALSE	lv hyp
4	4	37	Male	Cleveland	non-anginal	130	250	FALSE	norm
5	5	41	Female	Cleveland	atypical angina	130	204	FALSE	lv hyp
6	6	56	Male	Cleveland	atypical angina	120	236	FALSE	norm
7	7	62	Female	Cleveland	asymptomatic	140	268	FALSE	lv hyp
8	8	57	Female	Cleveland	asymptomatic	120	354	FALSE	norm

Showing 1 to 9 of 909 entries, 15 total columns

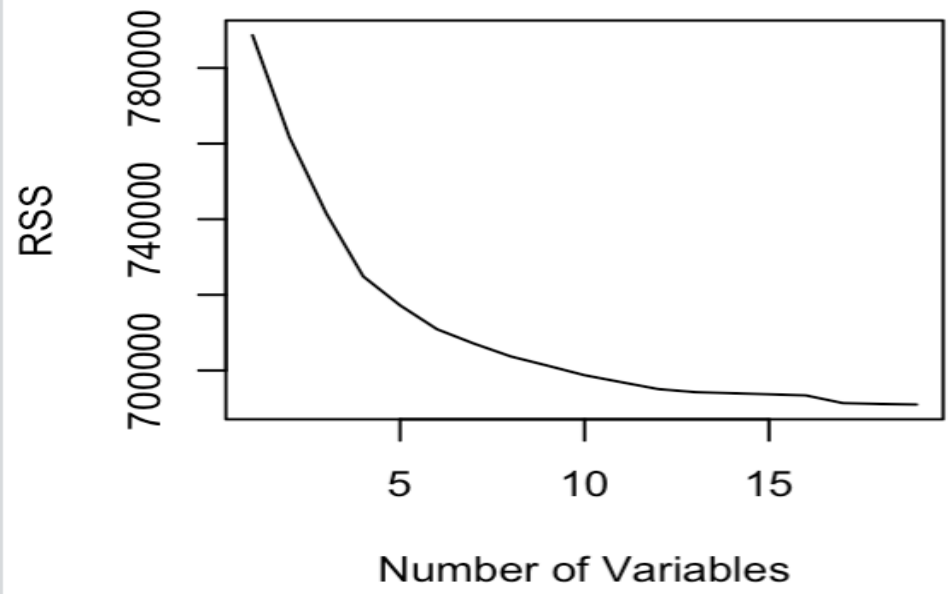
```
> regfit.bwd <- regsubsets(heart_disease$chol ~ ., data = heart_disease,
+                          nvmax = 19, method = "backward")
> summary(regfit.bwd)
Subset selection object
Call: regsubsets.formula(heart_disease$chol ~ ., data = heart_disease,
    nvmax = 19, method = "backward")
20 Variables  (and intercept)
                         Forced in Forced out
id                           FALSE      FALSE
age                          FALSE      FALSE
sexMale                      FALSE      FALSE
datasetHungary               FALSE      FALSE
datasetVA Long Beach         FALSE      FALSE
cpatypical angina            FALSE      FALSE
cpnon-anginal                FALSE      FALSE
cptypical angina             FALSE      FALSE
trestbps                     FALSE      FALSE
fbsTRUE                      FALSE      FALSE
restecgnormal                FALSE      FALSE
restecgst-t abnormality      FALSE      FALSE
thalch                       FALSE      FALSE
exangTRUE                    FALSE      FALSE
oldpeak                      FALSE      FALSE
slopeflat                    FALSE      FALSE
slopeupsloping               FALSE      FALSE
ca                           FALSE      FALSE
```

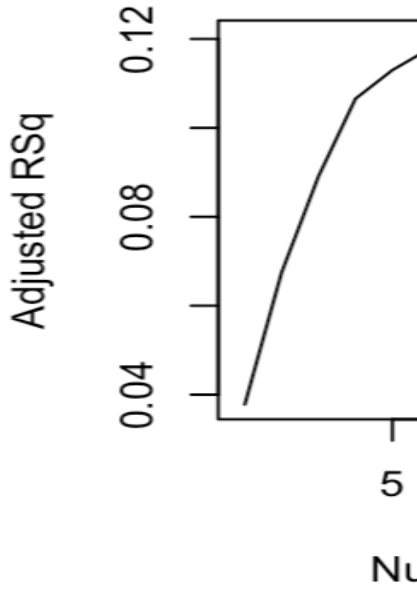
2) **Forward and Backward Features**

```
> par(mfrow = c(1, 1))
> plot(reg.summaryfwd$adjr2, xlab = "Number of Variables",
+      ylab = "Adjusted RSq", type = "l")
> points(11, reg.summaryfwd$adjr2[11], col = "red", cex = 2
+        pch = 20)
>
```


Adjusted RSq
0.04
0.06
0.08
0.10
0.12
5
10
Number of Variables

```
> reg.summarybwd <- summary(regfit.bwd)
> plot(reg.summarybwd$rss, xlab = "Number of Variables",
+      ylab = "RSS", type = "l")
> plot(reg.summarybwd$adjr2, xlab = "Number of Variables",
+      ylab = "Adjusted RSq", type = "l")
> par(mfrow = c(2, 2))
> plot(reg.summarybwd$rss, xlab = "Number of Variables",
+      ylab = "RSS", type = "l")
> plot(reg.summarybwd$adjr2, xlab = "Number of Variables",
+      ylab = "Adjusted RSq", type = "l")
>
```





3) PCR

```
> Heart_filtered <- heart_disease[complete.cases(heart_disease), ]
> pcr.fit <- pcr(Heart_filtered$age ~ ., data= Heart_filtered, scale = TRUE)
> summary(pcr.fit)
Data:   X dimension: 299 20
        Y dimension: 299 1
Fit method: svdpc
Number of components considered: 20
TRAINING: % variance explained
                    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
X                    18.883    27.74    36.38    43.02    49.25    55.28
Heart_filtered$age    9.306    16.51    16.62    17.71    18.86    22.27
                    10 comps  11 comps  12 comps  13 comps  14 comps  15 com
X                      74.74     78.95     82.99     86.58     89.73     92.
Heart_filtered$age     24.08     25.86     26.14     26.51     27.76     28.
                    18 comps  19 comps  20 comps
X                       99.0     99.55    100.00
Heart_filtered$age      33.7     33.70     33.71
>
```

PART-3: CLASSIFICATION

1) **Logistic Regression**

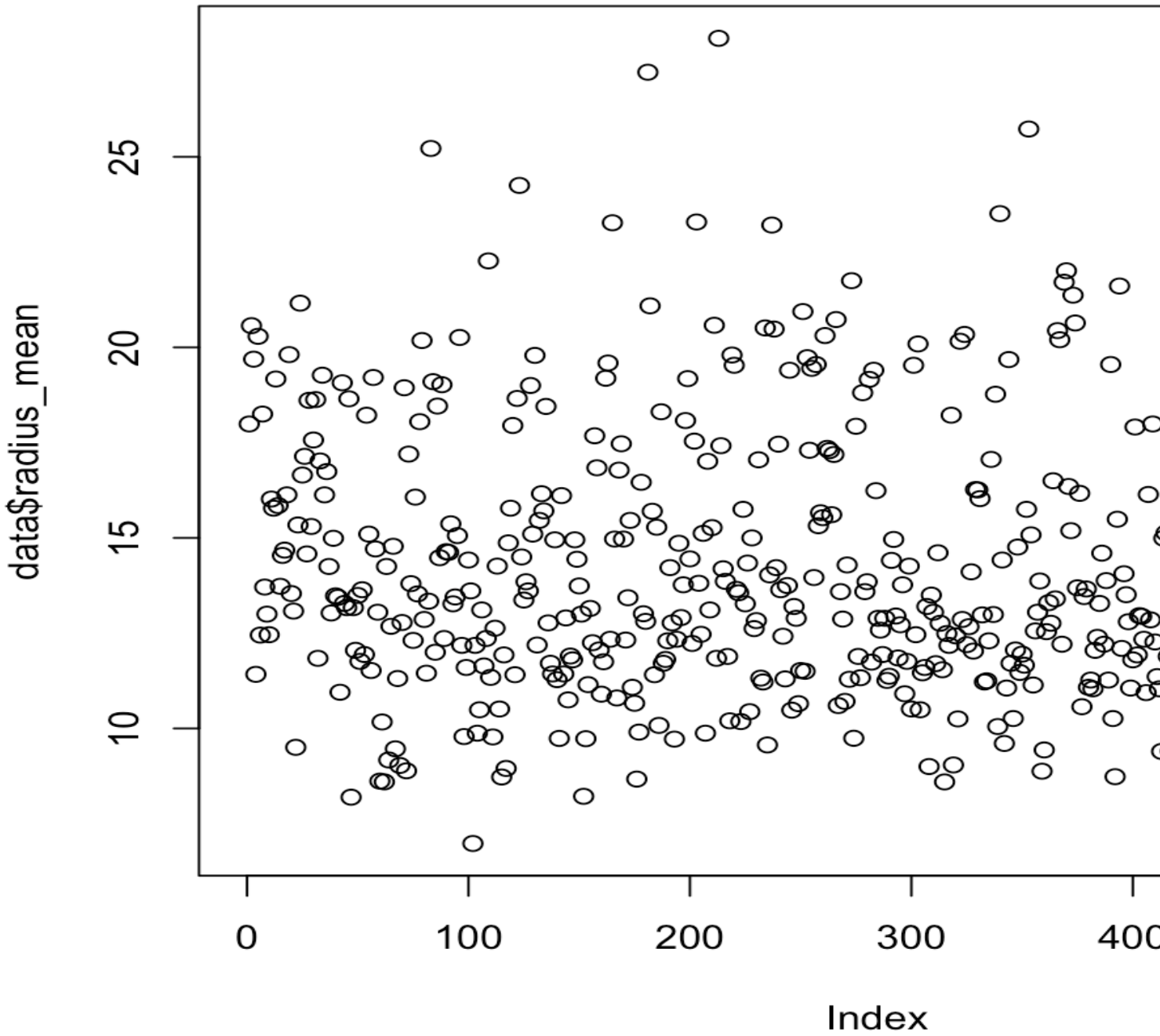
```
> library(readr)
> data <- read_csv("Downloads/data.csv")
New names:
• `` -> `...33`
Rows: 568 Columns: 33
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): diagnosis
dbl (31): id, radius_mean, texture_mean, perimeter_mean, area_mean, smoothnes
lgl  (1): ...33

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this mess
Warning message:
One or more parsing issues, call `problems()` on your data frame for details,
  dat <- vroom(...)
  problems(dat)
> View(data)
> library(ISLR2)
> names(data)
 [1] "id"                     "diagnosis"              "radius_mean"
 [4] "texture_mean"           "perimeter_mean"         "area_mean"
 [7] "smoothness_mean"        "compactness_mean"       "concavity_mean"
[10] "concave points_mean"    "symmetry_mean"          "fractal_dimension_m
[13] "radius_se"              "texture_se"             "perimeter_se"
[16] "area_se"                "smoothness_se"          "compactness_se"
[19] "concavity_se"           "concave points_se"      "symmetry_se"
[22] "fractal_dimension_se"   "radius_worst"           "texture_worst"
[25] "perimeter_worst"        "area_worst"             "smoothness_worst"
[28] "compactness_worst"      "concavity_worst"        "concave points_wors
```

Filter

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_
1	842302	M	17.990	10.38	122.80	
2	842517	M	20.570	17.77	132.90	
3	84300903	M	19.690	21.25	130.00	
4	84348301	M	11.420	20.38	77.58	
5	84358402	M	20.290	14.34	135.10	
6	843786	M	12.450	15.70	82.57	
7	844359	M	18.250	19.98	119.60	
8	84458202	M	13.710	20.83	90.20	
9	844981	M	13.000	21.82	87.50	
10	84501001	M	12.460	24.04	83.97	
11	845636	M	16.020	23.24	102.70	

Showing 1 to 12 of 568 entries, 33 total columns

```
> dim(data)
[1] 568  33
> summary(data)
       id              diagnosis          radius_mean      texture_mean   peri
 Min.   :     8670   Length:568         Min.   : 6.981   Min.   : 9.71   Min.
 1st Qu.:   869222   Class :character   1st Qu.:11.707   1st Qu.:16.17   1st
 Median :   906157   Mode  :character   Median :13.375   Median :18.84   Medi
 Mean   : 30425140                      Mean   :14.139   Mean   :19.28   Mean
 3rd Qu.:  8825022                      3rd Qu.:15.797   3rd Qu.:21.79   3rd
 Max.   :911320502                      Max.   :28.110   Max.   :39.28   Max.
   area_mean      smoothness_mean   compactness_mean  concavity_mean    conca
 Min.   : 143.5   Min.   :0.06251   Min.   :0.01938   Min.   :0.00000   Min.
 1st Qu.: 420.3   1st Qu.:0.08640   1st Qu.:0.06517   1st Qu.:0.02958   1st Q
 Median : 551.4   Median :0.09589   Median :0.09312   Median :0.06155   Media
 Mean   : 655.7   Mean   :0.09644   Mean   :0.10445   Mean   :0.08896   Mean
 3rd Qu.: 784.1   3rd Qu.:0.10533   3rd Qu.:0.13043   3rd Qu.:0.13100   3rd Q
 Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.
 symmetry_mean    fractal_dimension_mean   radius_se        texture_se      p
 Min.   :0.1060   Min.   :0.04996        Min.   :0.1115   Min.   :0.3602   Mi
 1st Qu.:0.1620   1st Qu.:0.05770        1st Qu.:0.2324   1st Qu.:0.8331   1s
> par(mfrow = c(1, 1))
> plot(data$radius_mean)
>
```

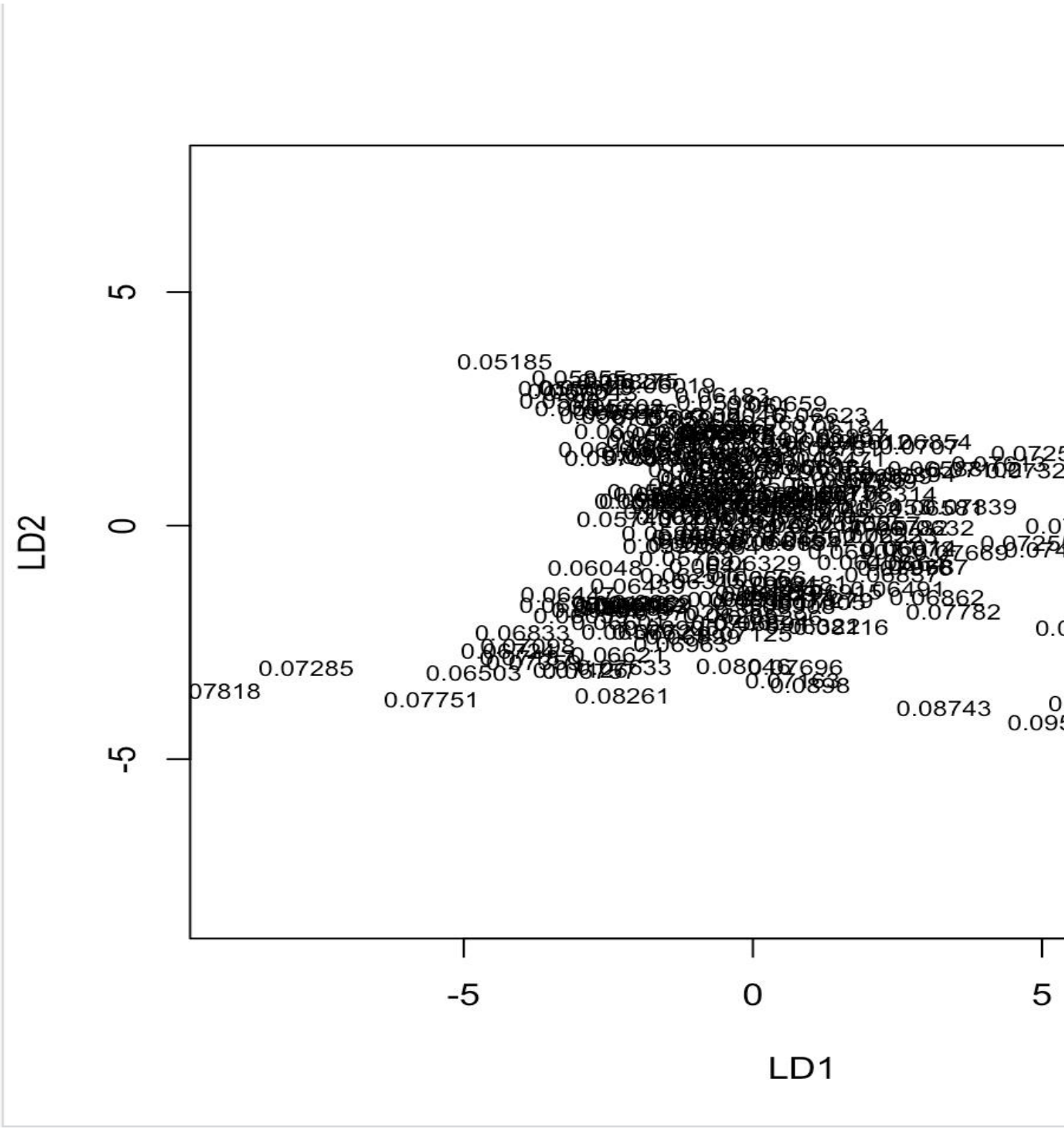

data$radius_mean
10
15
20
25
0
100
200
300
400
Index

```
> glm.fits <- glm(
+     data$fractal_dimension_mean  ~ data$smoothness_mean + data$area_mean
$concavity_mean,
+     data = data, family = binomial
+ )
Warning message:
In eval(family$initialize) : non-integer #successes in a binomial glm!
> summary(glm.fits)

Call:
glm(formula = data$fractal_dimension_mean ~ data$smoothness_mean +
    data$area_mean + data$compactness_mean + data$concavity_mean,
    family = binomial, data = data)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-0.033228  -0.008941  -0.002016   0.006243   0.058832

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)           -2.8645878  1.4594023  -1.963   0.0497 *
data$smoothness_mean   1.5933671 16.6624872   0.096   0.9238
data$area_mean        -0.0002607  0.0007198  -0.362   0.7172
data$compactness_mean  1.4549449  7.9225094   0.184   0.8543
data$concavity_mean    0.2519569  5.5507531   0.045   0.9638
```

```
> coef(glm.fits)
          (Intercept)  data$smoothness_mean         data$area_mean data$comp
        -2.8645878231          1.5933670854          -0.0002606788
  data$concavity_mean
         0.2519569260
> summary(glm.fits)$coef
                           Estimate   Std. Error     z value   Pr(>|z|)
(Intercept)           -2.8645878231 1.459402e+00 -1.96285000 0.04966361
data$smoothness_mean   1.5933670854 1.666249e+01  0.09562600 0.92381761
data$area_mean        -0.0002606788 7.198111e-04 -0.36214886 0.71724080
data$compactness_mean  1.4549449416 7.922509e+00  0.18364698 0.85429040
data$concavity_mean    0.2519569260 5.550753e+00  0.04539149 0.96379527
> summary(glm.fits)$coef[, 4]
          (Intercept)  data$smoothness_mean         data$area_mean data$comp
           0.04966361            0.92381761             0.71724080
  data$concavity_mean
           0.96379527
> glm.probs <- predict(glm.fits, type = "response")
> glm.probs[1:10]
         1          2          3          4          5          6
0.07890144 0.05026309 0.06173570 0.09412311 0.05733664 0.07598264 0.0574362
        10
0.08364694
>
>
```

1-b) Linear Discriminant Analysis:

```
> train <- (data$area_mean < 500)
> data.500 <- data[!train, ]
> dim(data.500)
[1] 339  33
>
> library(MASS)
> lda.fit <- lda(data$fractal_dimension_mean ~ data$perimeter_mean + data$a
+                subset = train)
> lda.fit
Call:
lda(data$fractal_dimension_mean ~ data$perimeter_mean + data$area_mean,
    data = data, subset = train)

Prior probabilities of groups:
    0.05185     0.05474     0.05502     0.05541     0.05561     0.05597
0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.0
    0.05667     0.05673     0.05677     0.05688     0.05698     0.05708
0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.0
    0.05743     0.05754     0.05768      0.0578     0.05808     0.05821
0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.0
    0.05855     0.05859     0.05865     0.05875     0.05878     0.05888
0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.0
    0.05914     0.05916     0.05934     0.05945     0.05948     0.05952
0.004366812 0.004366812 0.004366812 0.008733624 0.004366812 0.004366812 0.0
    0.05968     0.05975     0.05976     0.05977     0.05984     0.06013
0.004366812 0.004366812 0.004366812 0.004366812 0.004366812 0.008733624 0.0
    0.06031     0.06043     0.06046     0.06048     0.06057     0.06059
0.004366812 0.004366812 0.004366812 0.008733624 0.004366812 0.004366812 0.0
    0.06072     0.06081     0.06083     0.06095       0.061     0.06104
0.004366812 0.004366812 0.004366812 0.004366812 0.008733624 0.004366812 0.0

Proportion of trace:
   LD1    LD2
0.6593 0.3407
> plot(lda.fit)
>
```

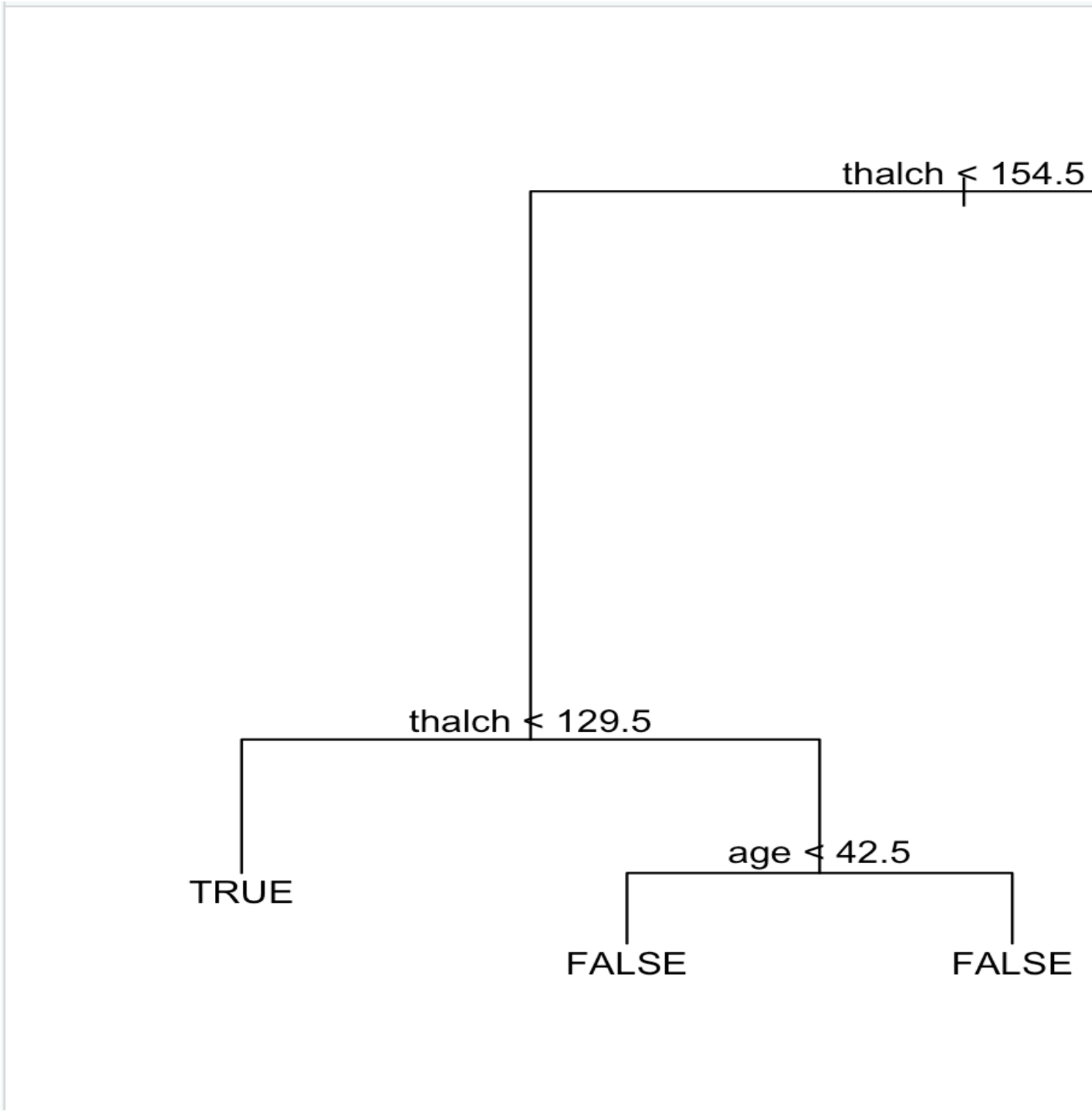

LD2
5
0
-5
LD1
-5
0
5
0.05185
0.07285
0.07818
0.07751
0.06503
0.08261
0.08743
0.06833
0.07782
0.06862
0.06048

```
> sum(lda.pred$posterior[, 1] >= .5)
[1] 202
> sum(lda.pred$posterior[, 1] < .5)
[1] 366
> sum(lda.pred$posterior[, 1] < .5)
[1] 366
> lda.pred$posterior[1:20, 1]
           1            2            3            4            5
9.999999e-01 9.985155e-01 1.000000e+00 5.975192e-59 1.000000e+00 1.642111e-
           8            9           10           11           12
2.205818e-06 3.750840e-24 9.073119e-31 9.999999e-01 9.999715e-01 1.000000e+
          15           16           17           18           19
1.821610e-27 2.034506e-02 9.998300e-01 9.742847e-01 1.000000e+00 4.070321e-
> lda.class[1:20]
 [1] 0.05185 0.05185 0.05185 0.09744 0.05185 0.07613 0.05185 0.0659  0.0725
[13] 0.05185 0.05185 0.08243 0.06275 0.05185 0.05185 0.05185 0.06275
217 Levels: 0.05185 0.05474 0.05502 0.05541 0.05561 0.05597 0.05628 0.05649
> sum(lda.pred$posterior[, 1] > .9)
[1] 179
>
```

2) **Tree Classifier**

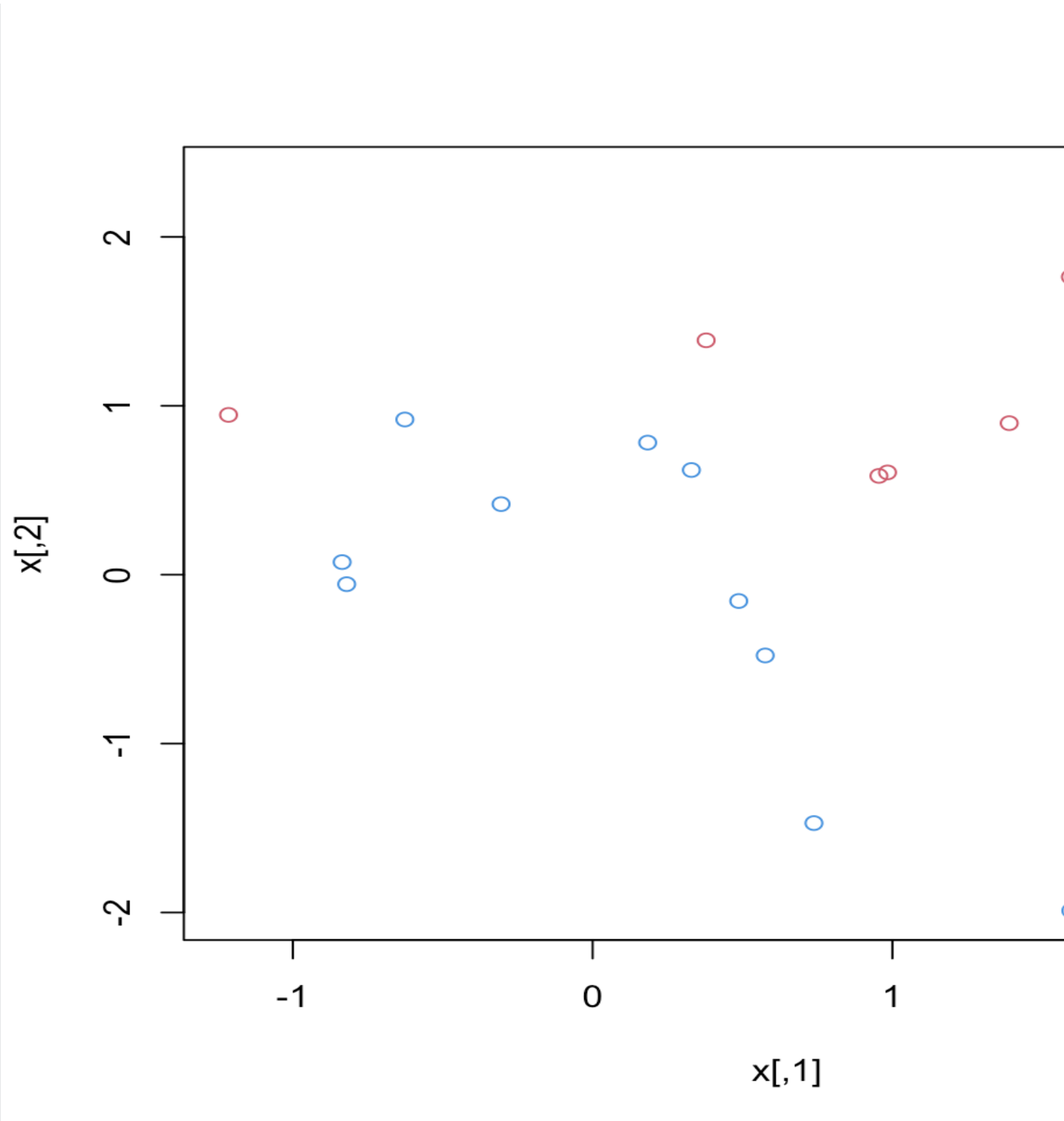
```
> attach(heart_disease)
The following object is masked from data:

    id

> library(caTools)
> library(ggplot2)
> library(Metrics)
> library(e1071)
> head(heart_disease)
# A tibble: 6 × 15
     id   age sex    dataset   cp       trestbps  chol fbs   restecg thalch
  <dbl> <dbl> <chr>  <chr>     <chr>       <dbl> <dbl> <lgl> <chr>    <dbl>
1     1    63 Male   Cleveland typica…       145   233 TRUE  lv hyp…    150
2     2    67 Male   Cleveland asympt…       160   286 FALSE lv hyp…    108
3     3    67 Male   Cleveland asympt…       120   229 FALSE lv hyp…    129
4     4    37 Male   Cleveland non-an…       130   250 FALSE normal     187
5     5    41 Female Cleveland atypic…       130   204 FALSE lv hyp…    172
6     6    56 Male   Cleveland atypic…       120   236 FALSE normal     178
# ℹ 1 more variable: thal <chr>
> split = sample.split(heart_disease$cp, SplitRatio = 0.7)
> trainingset = subset(heart_disease, split == TRUE)
> testset= subset(heart_disease, split == FALSE)
> tree.exang <- tree(as.factor(exang) ~ cp+age+sex+chol+fbs+thalch, trainin
Warning message:
In tree(as.factor(exang) ~ cp + age + sex + chol + fbs + thalch,  :
  NAs introduced by coercion
> summary(tree.exang)

Classification tree:
tree(formula = as.factor(exang) ~ cp + age + sex + chol + fbs +
    thalch, data = trainingset)
Variables actually used in tree construction:
[1] "thalch" "age"
Number of terminal nodes:  4
Residual mean deviance:  1.142 = 590.3 / 517
Misclassification error rate: 0.3052 = 159 / 521
```

```
> plot(tree.exang)
> test(tree.exang, pretty = 0)
Error in test(tree.exang, pretty = 0) : could not find function "test"
> text(tree.exang, pretty = 0)
> tree.pred <- predict(tree.exang, testset, type = "class")
Warning message:
In pred1.tree(object, tree.matrix(newdata)) : NAs introduced by coercion
> table(tree.pred, testset$exang)

tree.pred FALSE TRUE
    FALSE   116   46
    TRUE     44   58
```

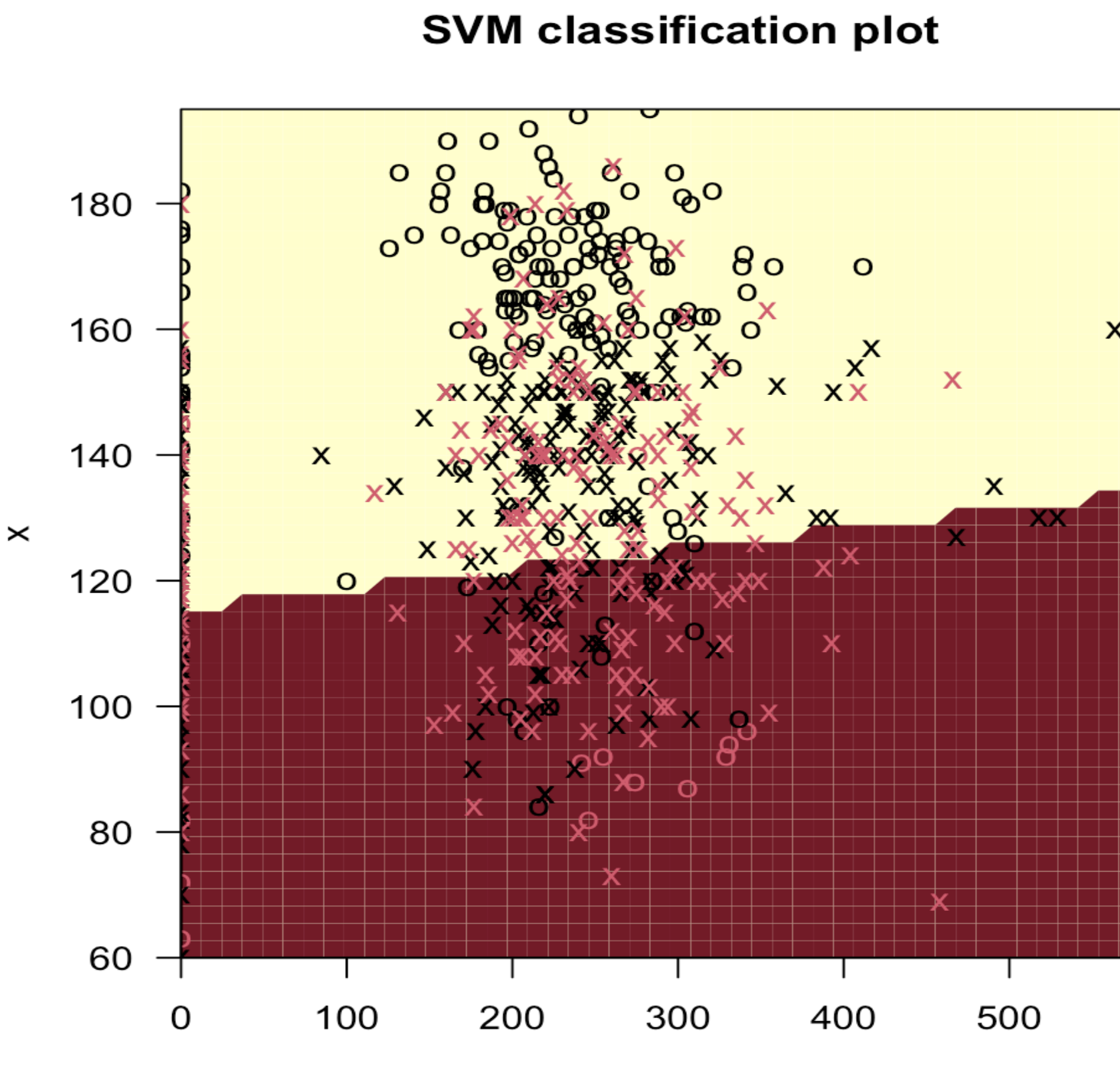


3) Support Vector Classifier

```
> set.seed(1)
> x <- matrix(rnorm(20 * 2), ncol = 2)
> y <- c(rep(-1, 10), rep(1, 10))
> x[y == 1, ] <- x[y == 1, ] + 1
> plot(x, col = (3 - y))
>
```


x[,2]
2
1
0
-1
-2
-1
0
1
x[,1]

```
> data = data.frame(x = heart_disease$thalch, y = as.factor(heart_disease$ex
> df = data.frame(x = heart_disease$thalch, y = as.factor(heart_disease$exan
> split = sample.split(df$z, SplitRatio = 0.7)
> trainset = subset(df, split == TRUE)
> tset = subset(data, split == FALSE)
> svmfit <- svm(y ~ x+z, data = trainset, kernel = "linear", cost = 1, scale
> summary(svmfit)

Call:
svm(formula = y ~ x + z, data = trainset, kernel = "linear", cost = 1, scale


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  446

 ( 224 222 )


Number of Classes:  2

Levels:
 FALSE TRUE



> plot(svmfit, trainset)
>
```


SVM classification plot
x
180
160
140
120
100
80
60
0
100
200
300
400
500

```
> tune.out <- tune(svm, y ~ ., data = dat, kernel = "linear",
+                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10,
Error in tune(svm, y ~ ., data = dat, kernel = "linear", ranges = lis
  object 'dat' not found
> tune.out <- tune(svm, y ~ ., data = trainset, kernel = "linear",
+                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10,
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.3523952

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.3959590 0.06330543
2 1e-02 0.3993243 0.06890859
3 1e-01 0.3523952 0.07846156
4 1e+00 0.3555636 0.07424868
5 5e+00 0.3555636 0.07424868
6 1e+01 0.3555636 0.07424868
7 1e+02 0.3555636 0.07424868

> bestmod <- tune.out$best.model
> summary(bestmod)

Call:
best.tune(METHOD = svm, train.x = y ~ ., data = trainset, ranges = li
    0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
```

PART-4: MODELS

a) There are four indicators (such as total inventory, number of employees, annual operating budget, and total earnings). It appears that every variable is linearly dependent. To determine the likelihood that his company concept would succeed, I would advise using multiple linear regression analysis.

b) The likelihood that a customer will visit the client's store can be determined using logistic regression. We can determine the precise regions where mailing can have the most impact based on overall results.

c) We need to convert the vast amounts of data we have into more straightforward features. We can utilize PCA to identify linear combinations of variables, and after that, we can choose the best regression techniques to uncover further information.

d) As previously indicated, there is significant overlap between the classes in this categorization problem. If it were possible to easily separate the characteristics, I would have recommended a multi-dimensional SVM. However, it appears that the random forest classifier can address these overlaps more precisely.