Name: Lavanya Ghanathey ID:16342393

Final Project

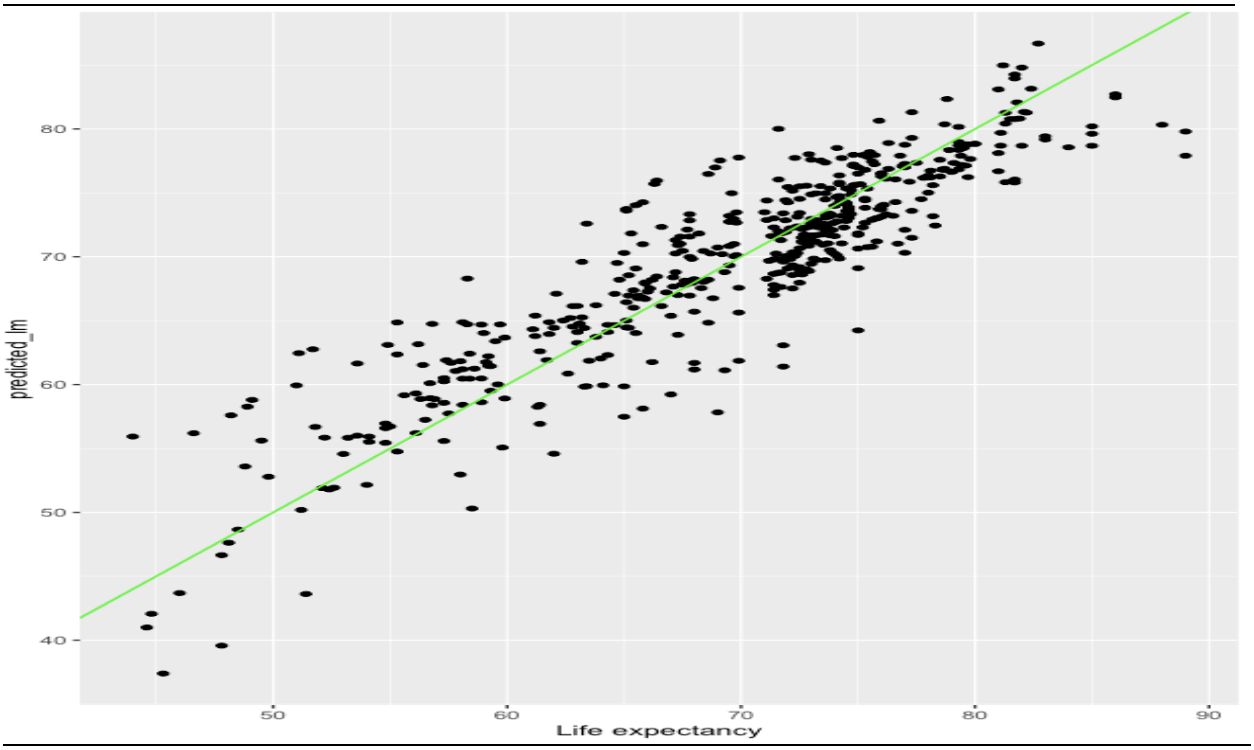
Part-1: Regression

Selected Dataset: Life Expectancy

a) Linear Regression: Life Expectancy Dataset is divided into a train and test set with a split of 70 into the train and 30 into the test. A Linear Regression analysis is performed.

```
#loading all the required packages
install.packages('readr')
install.packages('dplyr')
install.packages('ggplot2')
install.packages('caret')
library(readr)
library(dplyr)
library(ggplot2)
library(caret)


#reading the csv file and storing it in "Life"
Life <- read_csv("Downloads/Life Expectancy Data.csv")

#display top rows
head(data)

#removing NA values
sum(is.na(Life$"Life expectancy"))
Life <- na.omit(Life)

#split the data into training and testing sets using a 70/30 split
set.seed(393)
trainIndex <- createDataPartition(Life$"Life expectancy", p = .7, list = FALSE)
train <- Life[trainIndex,]
test <- Life[-trainIndex,]


#a)Linear Regression

#simple linear regression model using the lm() function
model_lm <- lm(`Life expectancy` ~ `Adult Mortality` + Alcohol + BMI + `HIV/AIDS` + `Income composition of resources` + Schooling + Status, data = train)
summary(model_lm)
#summary provides us with information of the coefficients
model_lm


#Plotting the model using a scatter plot of the predicted values against the actual values: A perfect model would have all the points lying on the red line.

predicted_lm <- predict(model_lm, newdata = test)
ggplot(test, aes(x = `Life expectancy`, y = predicted_lm)) + geom_point() + geom_abline(intercept = 0, slope = 1, color = "green")
```

```
Call:
lm(formula = `Life expectancy` ~ `Adult Mortality` + Alcohol +
    BMI + `HIV/AIDS` + `Income composition of resources` + Schooling +
    Status, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-17.3656  -2.2653   0.0637   2.2265  10.9347

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                     53.983122   0.860691  62.721  < 2e-16 ***
`Adult Mortality`               -0.018291   0.001160 -15.773  < 2e-16 ***
Alcohol                         -0.116728   0.040010  -2.917 0.003598 **
BMI                              0.046701   0.006842   6.826 1.41e-11 ***
`HIV/AIDS`                      -0.433241   0.021100 -20.533  < 2e-16 ***
`Income composition of resources` 11.766759   1.019796  11.538  < 2e-16 ***
Schooling                        0.974674   0.072579  13.429  < 2e-16 ***
StatusDeveloping                -1.386510   0.411405  -3.370 0.000776 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.777 on 1149 degrees of freedom
Multiple R-squared:  0.8192,    Adjusted R-squared:  0.8181
F-statistic: 743.5 on 7 and 1149 DF,  p-value: < 2.2e-16

> #summary provides us with information of the coefficients
> model_lm

Call:
lm(formula = `Life expectancy` ~ `Adult Mortality` + Alcohol +
    BMI + `HIV/AIDS` + `Income composition of resources` + Schooling +
    Status, data = train)

Coefficients:
                     (Intercept)                `Adult Mortality`
                        53.98312                         -0.01829
                         Alcohol                              BMI
                        -0.11673                          0.04670
                        `HIV/AIDS`  `Income composition of resources`
                        -0.43324                         11.76676
                        Schooling                StatusDeveloping
                         0.97467                         -1.38651

>
```



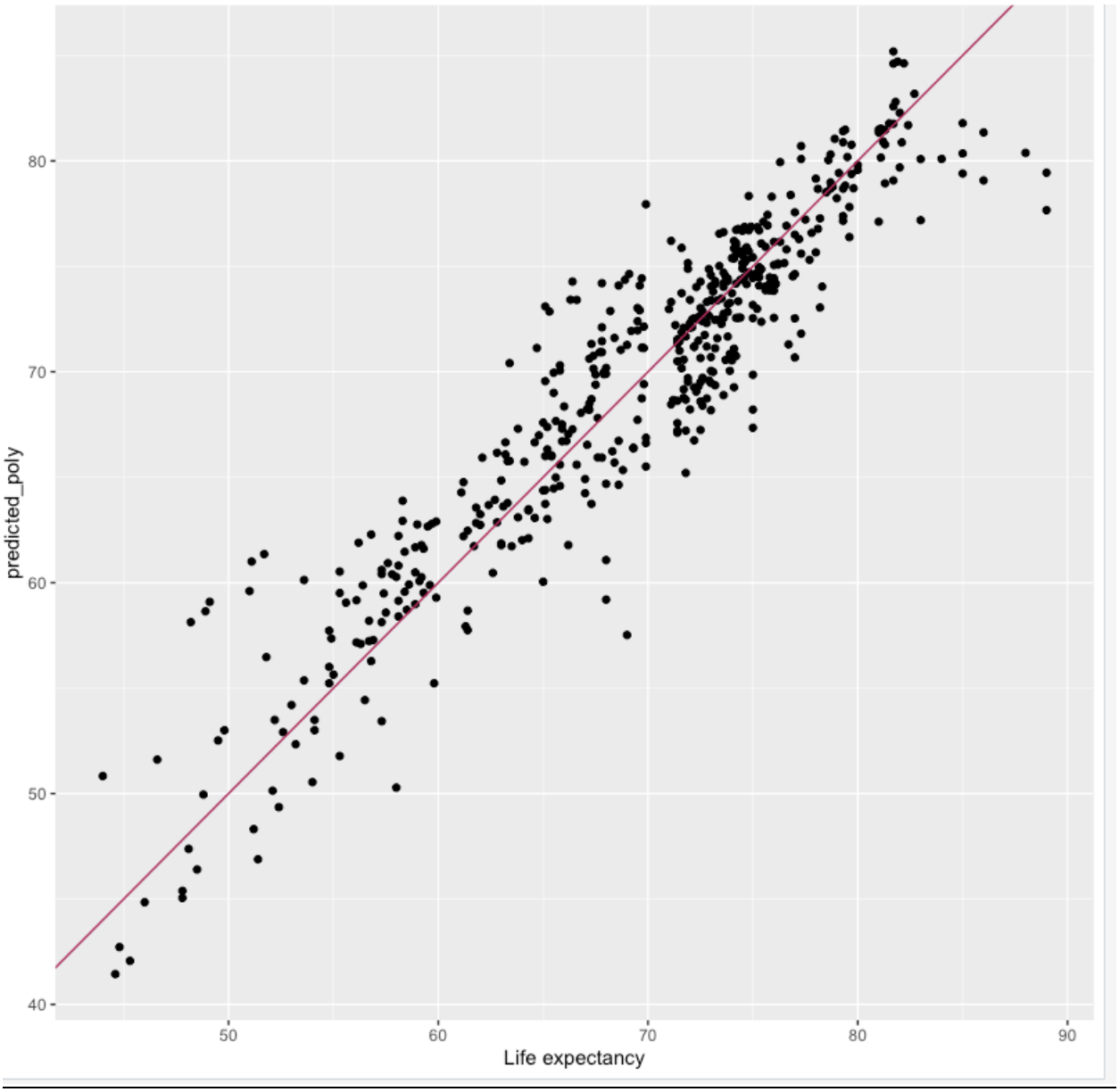
Comments:

The results of a multiple linear regression model that projects life expectancy depending on a number of predictor factors are displayed in the output. Adult Mortality, Alcohol, BMI, HIV/AIDS, Income Composition of Resources, Education, and Status are some of the factors. Positive coefficients indicate a positive link, while negative coefficients indicate a negative association, and the predictors' coefficients demonstrate how strongly each variable influences the anticipated life expectancy.

b) Polynomial Regression:

```
#b)Polynomial Regression
#we can try a polynomial regression model using the poly() function:
model_poly <- lm(`Life expectancy` ~ poly(`Adult Mortality`, 2) + poly(Alcohol, 2) + poly(BMI, 2) + poly(`HIV/AIDS`, 2) + poly(`Income composition of resources`, 2) + poly(Schooling, 2) + S

summary(model_poly)


#visualize the model using a scatter plot of the predicted values against the actual values:
predicted_poly <- predict(model_poly, newdata = test)
ggplot(test, aes(x = `Life expectancy`, y = predicted_poly)) + geom_point() + geom_abline(intercept = 0, slope = 1, color = "maroon")
```

```
Call:
lm(formula = `Life expectancy` ~ poly(`Adult Mortality`, 2) +
    poly(Alcohol, 2) + poly(BMI, 2) + poly(`HIV/AIDS`, 2) + poly(`Income composition of resources`,
    2) + poly(Schooling, 2) + Status, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-15.691  -1.886  -0.095   1.861  10.493

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                              69.9139     0.3403 205.469  < 2e-16 ***
poly(`Adult Mortality`, 2)1             -50.0111     4.7431 -10.544  < 2e-16 ***
poly(`Adult Mortality`, 2)2              -3.2980     4.3970  -0.750 0.453373
poly(Alcohol, 2)1                       -29.7287     4.9072  -6.058 1.87e-09 ***
poly(Alcohol, 2)2                       -16.9089     3.4192  -4.945 8.74e-07 ***
poly(BMI, 2)1                            13.7688     4.1304   3.334 0.000885 ***
poly(BMI, 2)2                            -2.6096     3.6094  -0.723 0.469826
poly(`HIV/AIDS`, 2)1                    -89.8743     5.4431 -16.512  < 2e-16 ***
poly(`HIV/AIDS`, 2)2                     30.8572     3.7592   8.208 5.99e-16 ***
poly(`Income composition of resources`, 2)1 169.0004     8.0927  20.883  < 2e-16 ***
poly(`Income composition of resources`, 2)2  99.0239     5.7611  17.188  < 2e-16 ***
poly(Schooling, 2)1                       3.6095     7.9243   0.455 0.648837
poly(Schooling, 2)2                     -23.3249     3.7658  -6.194 8.17e-10 ***
StatusDeveloping                         -0.6944     0.3831  -1.813 0.070114 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.188 on 1143 degrees of freedom
Multiple R-squared:  0.8718,    Adjusted R-squared:  0.8704
F-statistic: 598.2 on 13 and 1143 DF,  p-value: < 2.2e-16
```



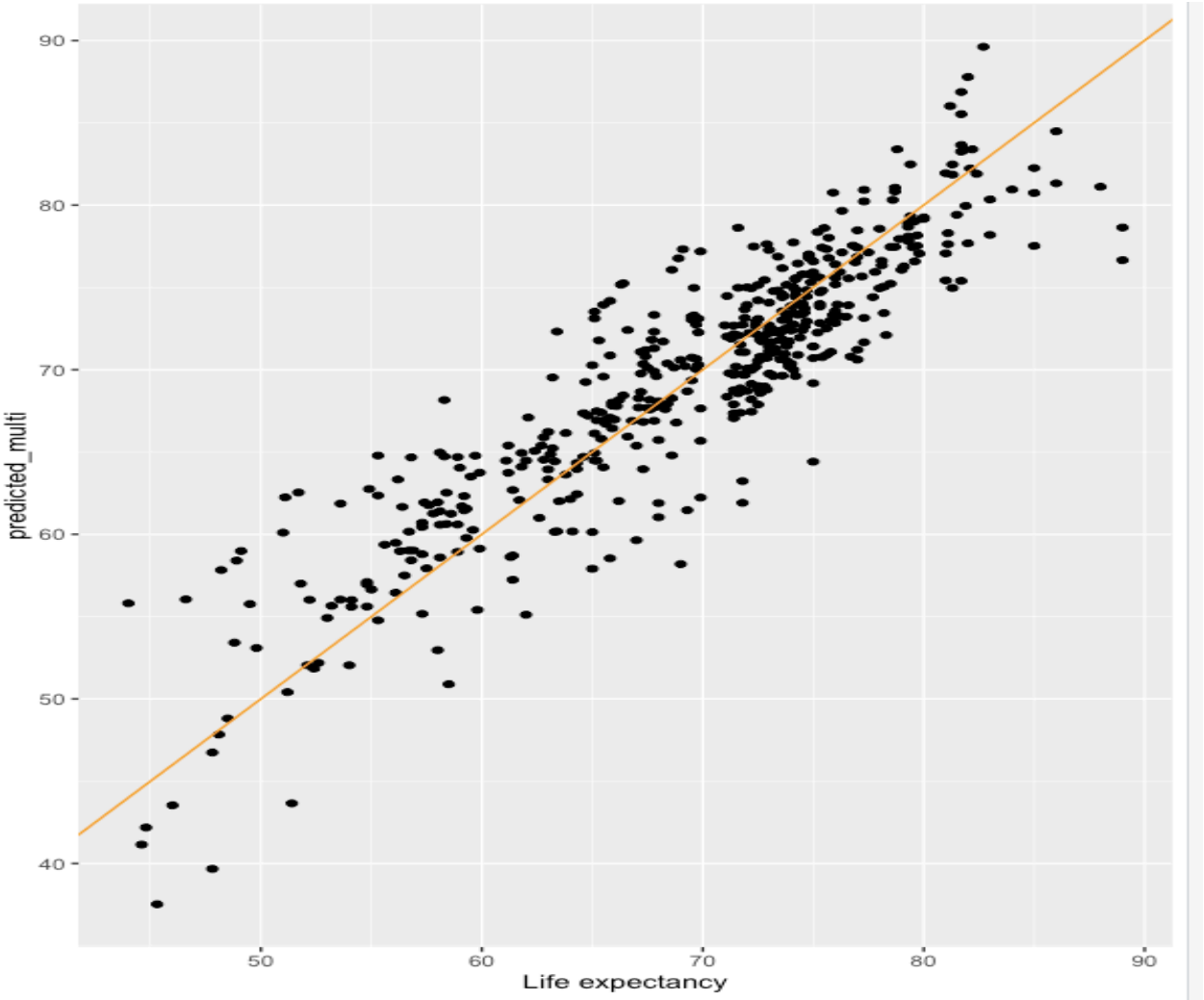
Comments:

This is the summary result of a linear regression model with life expectancy as the response variable and adult mortality, alcoholism, body mass index (BMI), HIV/AIDS, resource distribution according to income, education, and status as predictor factors. The model's adjusted R-squared score of 0.8704 indicates that the predictor variables account for 87.04% of the variation in the response variable. The model is statistically significant, as shown by the F-statistic's p-value of 2.2e-16.

c) Multi-Linear Regression:

```r
#c)Multilinear Regression
model_multi <- lm(`Life expectancy` ~ `Adult Mortality` + Alcohol + BMI + `HIV/AIDS` + `Income composition of resources` + Schooling + Status + GDP + Population, data = train)

summary(model_multi)


predicted_multi <- predict(model_multi, newdata = test)
ggplot(test, aes(x = `Life expectancy`, y = predicted_multi)) + geom_point() + geom_abline(intercept = 0, slope = 1, color = "orange")
```

```
Call:
lm(formula = `Life expectancy` ~ `Adult Mortality` + Alcohol +
    BMI + `HIV/AIDS` + `Income composition of resources` + Schooling +
    Status + GDP + Population, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-17.3627  -2.0986   0.0754   2.3275  11.6648

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                      5.409e+01  8.475e-01  63.823  < 2e-16 ***
`Adult Mortality`               -1.782e-02  1.144e-03 -15.587  < 2e-16 ***
Alcohol                         -1.455e-01  3.964e-02  -3.670 0.000254 ***
BMI                              4.707e-02  6.752e-03   6.972 5.28e-12 ***
`HIV/AIDS`                      -4.364e-01  2.079e-02 -20.991  < 2e-16 ***
`Income composition of resources`  1.118e+01  1.009e+00  11.075  < 2e-16 ***
Schooling                        9.281e-01  7.181e-02  12.924  < 2e-16 ***
StatusDeveloping                -8.344e-01  4.142e-01  -2.015 0.044190 *
GDP                              7.376e-05  1.175e-05   6.279 4.83e-10 ***
Population                      -7.915e-10  1.643e-09  -0.482 0.630064
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.716 on 1147 degrees of freedom
Multiple R-squared:  0.8252,    Adjusted R-squared:  0.8238
F-statistic: 601.7 on 9 and 1147 DF,  p-value: < 2.2e-16
```



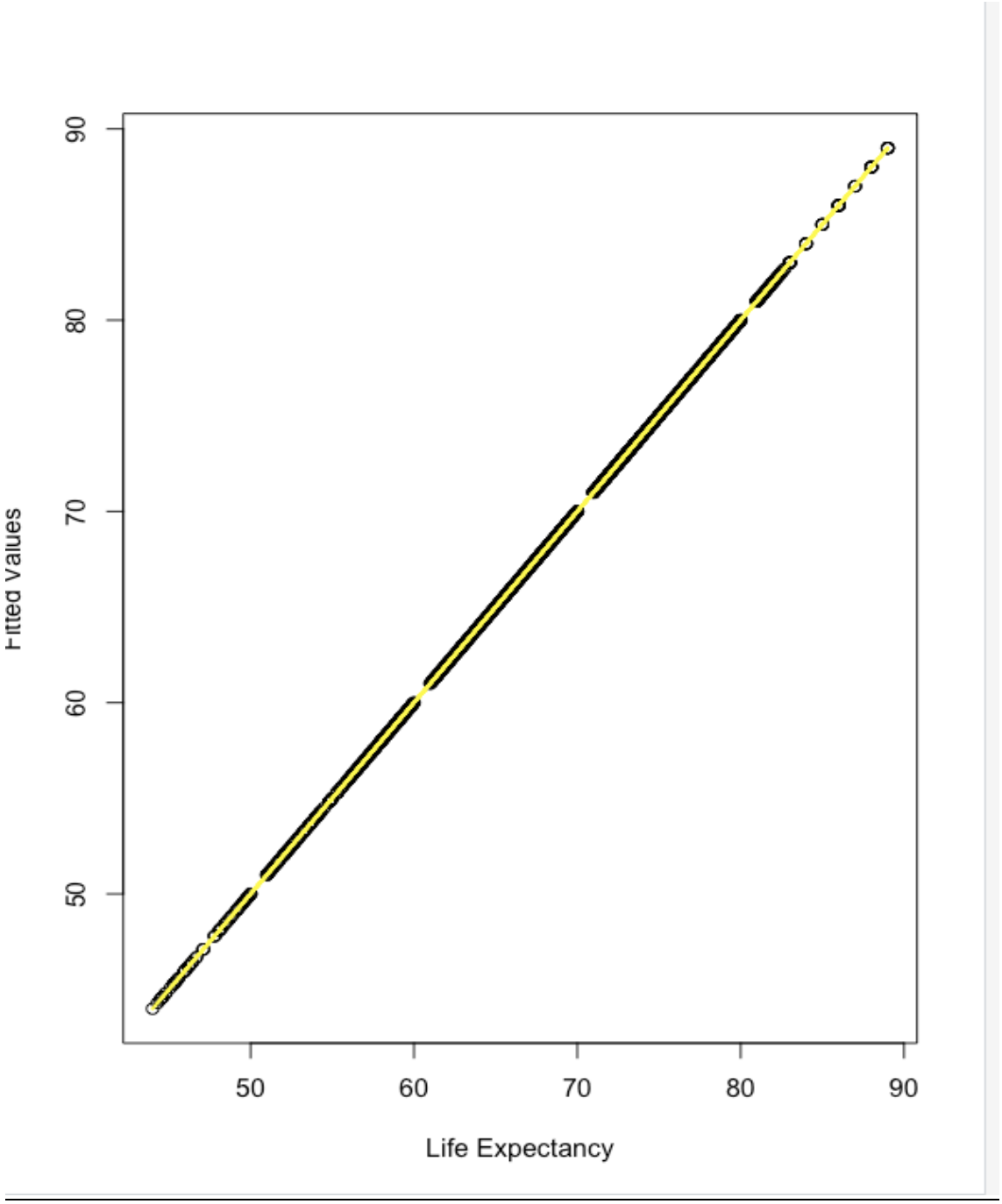
Comment:

This is the summary result of a linear regression model with life expectancy as the response variable and adult mortality, alcoholism, body mass index (BMI), HIV/AIDS, resource distribution according to income, education, and status as predictor factors. The model's adjusted R-squared score of 0.8238 indicates that the predictor variables account for 82.38% of the variation in the response variable. The model is statistically significant, as shown by the F-statistic's p-value of 2.2e-16.

d) Natural Cubic Spline:

```r
#d) Natural Cubic Spline
install.packages("splines")
install.packages("dplyr")
library(splines)
library(dplyr)

#Natural Cubic Spline using ns()
model_ns <- lm(`Life expectancy` ~ ns(`Life expectancy`, df = 6), data = Life)
summary(model_ns)

plot(Life$`Life expectancy`, Life$`Life expectancy`, xlab = "Life Expectancy", ylab = "Fitted Values")
lines(Life$`Life expectancy`, predict(model_ns), col = "yellow", lwd = 2)
```

```
Call:
lm(formula = `Life expectancy` ~ ns(`Life expectancy`, df = 6),
    data = Life)

Residuals:
       Min         1Q     Median         3Q        Max
-2.738e-12 -1.920e-15  5.200e-16  4.470e-15  9.590e-13

Coefficients:
                                Estimate Std. Error   t value Pr(>|t|)
(Intercept)                    4.400e+01  1.520e-14 2.896e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)1 2.210e+01  1.535e-14 1.440e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)2 2.680e+01  1.913e-14 1.401e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)3 3.017e+01  1.627e-14 1.854e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)4 3.094e+01  1.435e-14 2.157e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)5 5.232e+01  3.711e-14 1.410e+15   <2e-16 ***
ns(`Life expectancy`, df = 6)6 3.868e+01  2.027e-14 1.908e+15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.378e-14 on 1642 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:      1
F-statistic: 3.028e+30 on 6 and 1642 DF,  p-value: < 2.2e-16
```



Comments:

The link between the dependent variable (Life expectancy in this case) and the independent variable (Life expectancy once more, but with a non-linear transformation using a natural cubic spline with 4 degrees of freedom) is represented in this example of non-linear regression. The findings reveal a highly significant link (p-value 2.2e-16) and a perfect fit for the model (Multiple R-squared: 1, Adjusted R-squared: 1).

Part-2: Feature Selection/Model Optimization Methods

Selected Dataset: Wine Quality

1) Stepwise Selection:
   a) Forward Stepwise Selection and b) Backward Stepwise Selection:

```r
# Load the required libraries
library(readr)
library(leaps)
# Read the CSV file into a data frame
wine<- read_csv("Downloads/winequality-red.csv")

# Split the data into training and testing sets
set.seed(393)
trainIndex <- createDataPartition(wine$quality, p = .7, list = FALSE)
train <- wine[trainIndex, ]
test <- wine[-trainIndex, ]

# Forward Stepwise Selection
fit.fs <- lm(quality ~ 1, data = train)
for (i in 2:ncol(train)) {
  fit.temp <- lm(quality ~ ., data = train1[, c(names(train)[i], names(fit.fs$model))])
  if (AIC(fit.temp) < AIC(fit.fs)) {
    fit.fs <- fit.temp
  } else {
    break
  }
}
summary(fit.fs)

# Backward Stepwise Selection
fit.bs <- lm(quality ~ ., data = train)
while (length(coefficients(fit.bs)) > 1) {
  pvals <- summary(fit.bs)$coefficients[, 4]
  maxp <- max(pvals[-1])
  if (maxp > 0.05) {
    exclude <- names(coefficients(fit.bs))[pvals == maxp]
    formula <- as.formula(paste("quality ~", paste(setdiff(names(train), exclude), collapse = "+")))
    fit.bs <- lm(formula, data = train)
  } else {
    break
  }
}
summary(fit.bs)


# Forward Stepwise Selection
fit <- lm(quality ~ ., data = train)
forward_fit <- step(fit, direction = "forward")
```

```r
# Backward Stepwise Selection
fit <- lm(quality ~ ., data = train)
backward_fit <- step(fit, direction = "backward")
```

```
Subset selection object
Call: regsubsets.formula(quality ~ ., data = train, nvmax = 12, method = "forward")
11 Variables  (and intercept)
                      Forced in Forced out
`fixed acidity`           FALSE      FALSE
`volatile acidity`        FALSE      FALSE
`citric acid`             FALSE      FALSE
`residual sugar`          FALSE      FALSE
chlorides                 FALSE      FALSE
`free sulfur dioxide`     FALSE      FALSE
`total sulfur dioxide`    FALSE      FALSE
density                   FALSE      FALSE
pH                        FALSE      FALSE
sulphates                 FALSE      FALSE
alcohol                   FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: forward
          `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides `free sulfur dioxide`
1  ( 1 )  " "             " "                " "           " "              " "        " "
2  ( 1 )  " "             "*"                " "           " "              " "        " "
3  ( 1 )  " "             "*"                " "           " "              " "        " "
4  ( 1 )  " "             "*"                " "           " "              " "        " "
5  ( 1 )  " "             "*"                " "           " "              "*"        " "
6  ( 1 )  " "             "*"                " "           " "              "*"        " "
7  ( 1 )  " "             "*"                " "           " "              "*"        "*"
8  ( 1 )  " "             "*"                "*"           " "              "*"        "*"
9  ( 1 )  " "             "*"                "*"           "*"              "*"        "*"
10  ( 1 ) "*"             "*"                "*"           "*"              "*"        "*"
11  ( 1 ) "*"             "*"                "*"           "*"              "*"        "*"
          `total sulfur dioxide` density pH  sulphates alcohol
1  ( 1 )  " "                    " "     " " " "       "*"
2  ( 1 )  " "                    " "     " " " "       "*"
3  ( 1 )  " "                    " "     " " "*"       "*"
4  ( 1 )  "*"                    " "     " " "*"       "*"
5  ( 1 )  "*"                    " "     " " "*"       "*"
6  ( 1 )  "*"                    " "     "*" "*"       "*"
7  ( 1 )  "*"                    " "     "*" "*"       "*"
8  ( 1 )  "*"                    " "     "*" "*"       "*"
9  ( 1 )  "*"                    " "     "*" "*"       "*"
10  ( 1 ) "*"                    " "     "*" "*"       "*"
11  ( 1 ) "*"                    "*"     "*" "*"       "*"
>
>
```

```
Subset selection object
Call: regsubsets.formula(quality ~ ., data = train, nvmax = 12, method = "backward")
11 Variables  (and intercept)
                      Forced in Forced out
`fixed acidity`           FALSE      FALSE
`volatile acidity`        FALSE      FALSE
`citric acid`             FALSE      FALSE
`residual sugar`          FALSE      FALSE
chlorides                 FALSE      FALSE
`free sulfur dioxide`     FALSE      FALSE
`total sulfur dioxide`    FALSE      FALSE
density                   FALSE      FALSE
pH                        FALSE      FALSE
sulphates                 FALSE      FALSE
alcohol                   FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: backward
          `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides `free sulfur dioxide`
1  ( 1 )  " "             " "                " "           " "              " "        " "
2  ( 1 )  " "             "*"                " "           " "              " "        " "
3  ( 1 )  " "             "*"                " "           " "              " "        " "
4  ( 1 )  " "             "*"                " "           " "              " "        " "
5  ( 1 )  " "             "*"                " "           " "              "*"        " "
6  ( 1 )  " "             "*"                " "           " "              "*"        " "
7  ( 1 )  " "             "*"                "*"           " "              "*"        " "
8  ( 1 )  " "             "*"                "*"           "*"              "*"        " "
9  ( 1 )  "*"             "*"                "*"           "*"              "*"        " "
10  ( 1 ) "*"             "*"                "*"           "*"              "*"        "*"
11  ( 1 ) "*"             "*"                "*"           "*"              "*"        "*"
          `total sulfur dioxide` density pH  sulphates alcohol
1  ( 1 )  " "                    " "     " " " "       "*"
2  ( 1 )  " "                    " "     " " " "       "*"
3  ( 1 )  " "                    " "     " " "*"       "*"
4  ( 1 )  "*"                    " "     " " "*"       "*"
5  ( 1 )  "*"                    " "     " " "*"       "*"
6  ( 1 )  "*"                    " "     "*" "*"       "*"
7  ( 1 )  "*"                    " "     "*" "*"       "*"
8  ( 1 )  "*"                    " "     "*" "*"       "*"
9  ( 1 )  "*"                    " "     "*" "*"       "*"
10  ( 1 ) "*"                    " "     "*" "*"       "*"
11  ( 1 ) "*"                    "*"     "*" "*"       "*"
```

Comments:

Forward Stepwise Selection

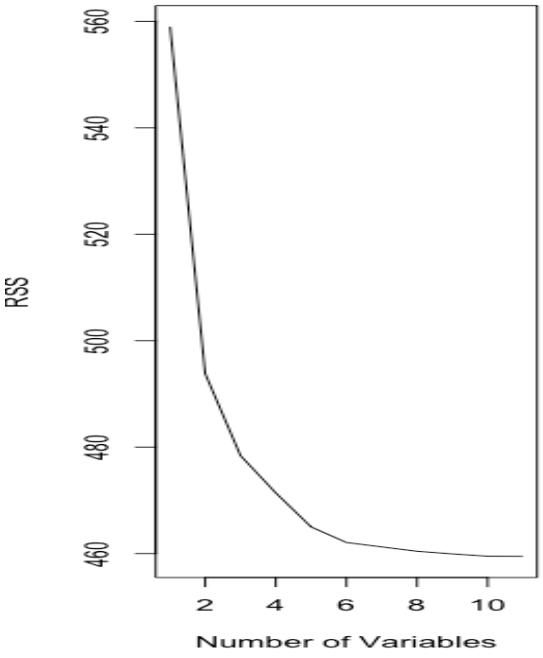
The output displays the outcomes of a subset selection technique, namely forward selection, used with training data to create a regression model with response variable "quality" and 11 predictor variables.

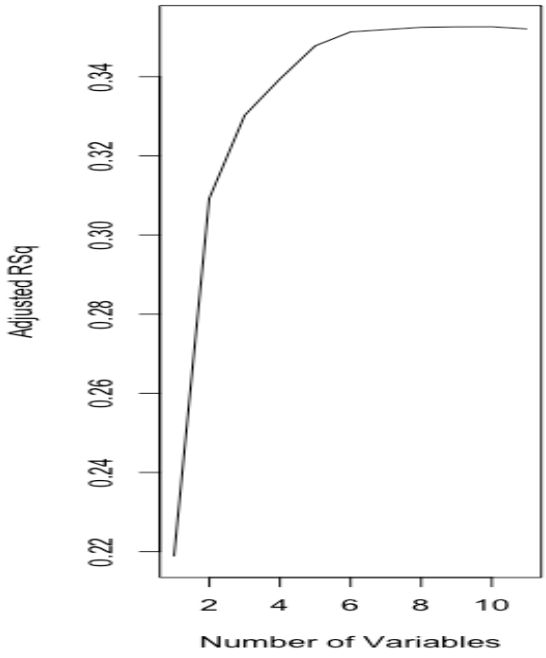
The table lists the variables that were chosen (indicated by a "") at each algorithmic stage, up to a maximum of 11 variables. The algorithm begins with a blank model and incrementally adds the predictors that, based on some criterion (not indicated in the output), produce the best fit. The results indicate that the final model of choice includes all the predictors, which is consistent with the last row's designation of all variables as "" in the output.

Backward Stepwise Selection

The response variable quality and all other factors were used as potential predictors in a backward stepwise subset selection using the regsubsets() function in R. The variables with an asterisk were included in the final model, and the table lists which variables were included or omitted at each stage of the selection process. The procedure used to choose the variables was backward, which means it started with all of the variables and eliminated them one at a time until the best model according to some criterion was determined. The result indicates that the final model contains all variables because they are all indicated in the last row with an asterisk.

2) Generating the plots of RSS and Adjusted R2generate the plots of RSS and Adjusted R^2 using the models generated for the feature selection.
   a) Forward Features

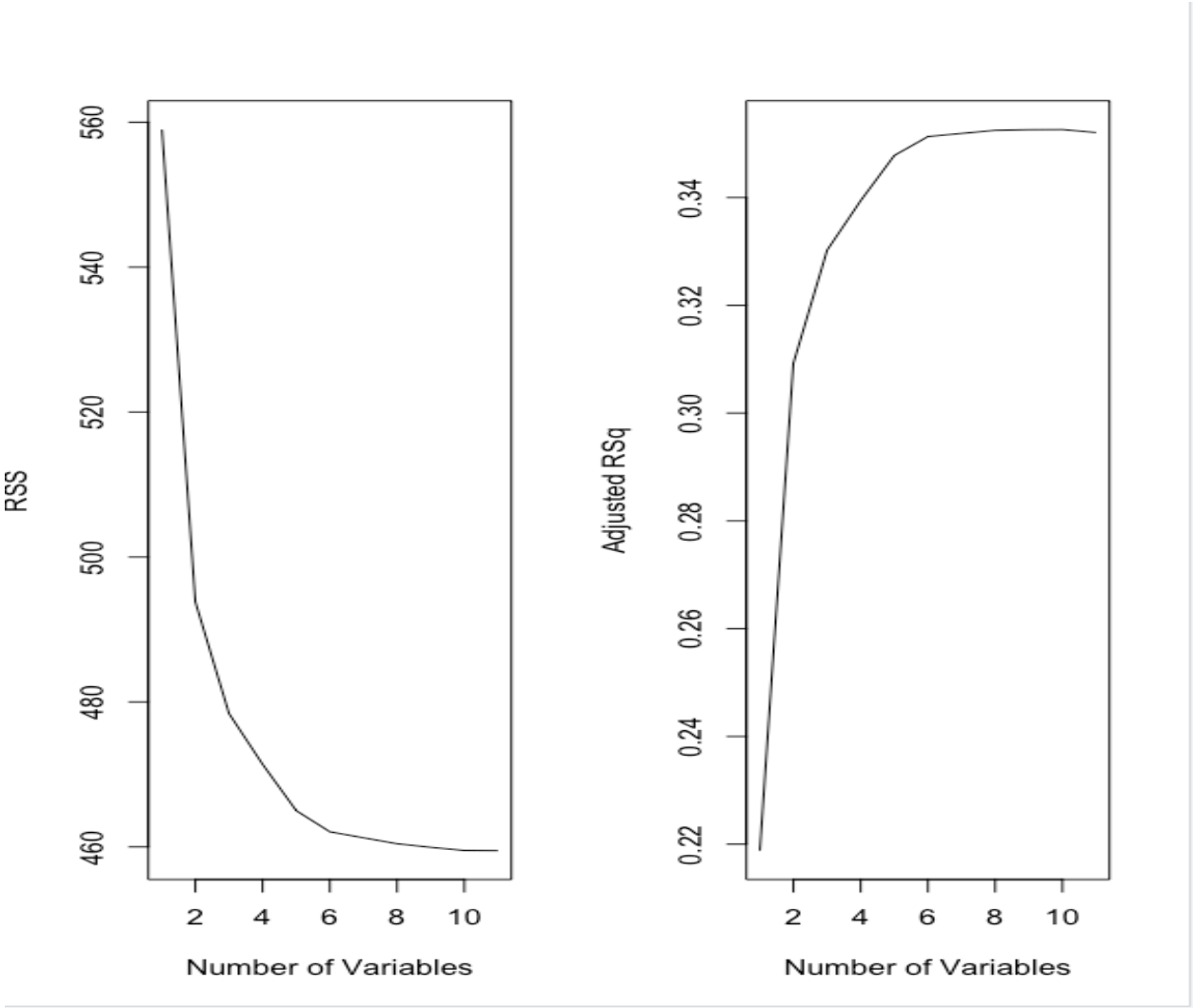




Comments :

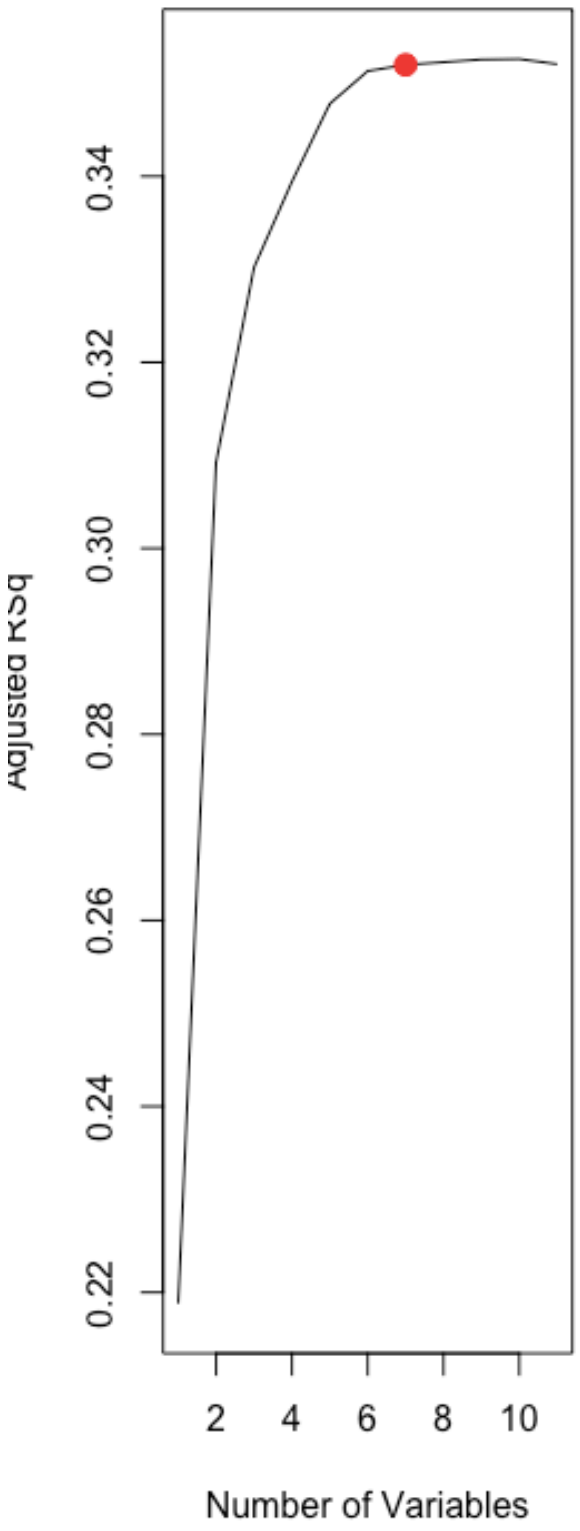
Ideal No of Variables using the Forward Stepwise Selection are 6.

a) Backward Features



Comments :

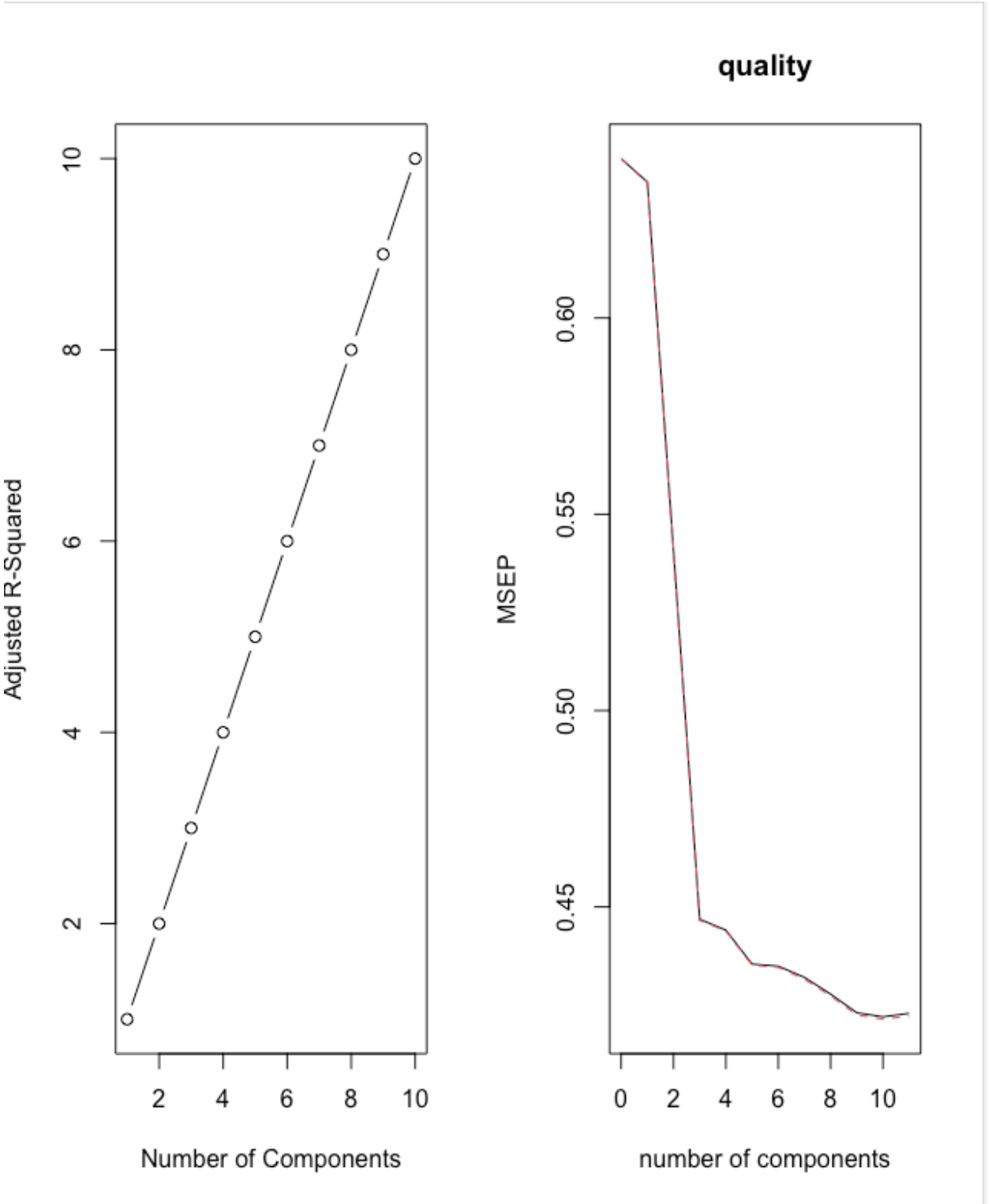
Ideal No of Variables using the Backward Stepwise Selection are 6.


Adjusted RSq
0.22
0.24
0.26
0.28
0.30
0.32
0.34
2
4
6
8
10
Number of Variables

3) PCR

```r
#pcr
install.packages("pls")
library(pls)

pcr_model <- pcr(quality ~ ., data = train, scale = TRUE, validation = "CV")
summary(pcr_model)

# Plot the components relative to their fit target
plot(1:10, pcr_model$validation$R2, type = "b", xlab = "Number of Components", ylab = "Adjusted R-Squared")

#pcr2
library(pls)
set.seed(393)
pcr.fit <- pcr(quality ~ ., data = train, scale = TRUE,
               validation = "CV")

summary(pcr.fit)

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

```
> summary(pcr_model)
Data:   X dimension: 1120 11
        Y dimension: 1120 1
Fit method: svdpc
Number of components considered: 11

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV          0.8004   0.7958   0.7351   0.6679   0.6665   0.6593   0.6594   0.6560   0.6525   0.6494    0.6478    0.6484
adjCV       0.8004   0.7957   0.7350   0.6677   0.6663   0.6591   0.6592   0.6557   0.6522   0.6491    0.6475    0.6480

TRAINING: % variance explained
         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
X         28.223    46.02    59.74    70.68    79.47    85.55    90.81    94.69    97.83     99.47    100.00
quality    1.444    16.16    30.68    31.21    32.72    32.83    33.68    34.63    35.43     35.80     35.84
> # Plot the components relative to their fit target
> plot(1:10, pcr_model$validation$R2, type = "b", xlab = "Number of Components", ylab = "Adjusted R-Squared")
>
> #pcr2
> library(pls)
> set.seed(393)
> pcr.fit <- pcr(quality ~ ., data = train, scale = TRUE,
+                validation = "CV")
>
> summary(pcr.fit)
Data:   X dimension: 1120 11
        Y dimension: 1120 1
Fit method: svdpc
Number of components considered: 11

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV          0.8004   0.7966   0.7352   0.6685   0.6664   0.6599   0.6595   0.6573   0.6541   0.6504    0.6496    0.6503
adjCV       0.8004   0.7965   0.7350   0.6682   0.6662   0.6597   0.6592   0.6569   0.6537   0.6500    0.6491    0.6498

TRAINING: % variance explained
         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
X         28.223    46.02    59.74    70.68    79.47    85.55    90.81    94.69    97.83     99.47    100.00
quality    1.444    16.16    30.68    31.21    32.72    32.83    33.68    34.63    35.43     35.80     35.84
> validationplot(pcr.fit, val.type = "MSEP")
```



Comments:

The results are from a principle component regression (PCR) model that was adjusted to forecast the wine's quality based on 11 predictor variables.

The root mean squared error of prediction (RMSEP) for models with various numbers of principal components is displayed in the validation section. Each principal component's share of the variance in the response variable (quality) and predictor variables is displayed in the training section. According to the training section, the first two components account for the majority of the data's fluctuation, whilst the remaining components do not explain nearly as much.

Part-3: Classification

Selected Dataset: Breast Cancer

1) Logistic Regression and Linear Discriminant Analysis:

```
data5 <- read_csv("Downloads/data.csv")
data5



bc_data <- data5 %>%
  mutate(diagnosis_bin = if_else(diagnosis == "B", 0, 1)) %>%
  select(-id, -diagnosis)

# Split the data into training and testing datasets using a 70/30 split ratio
set.seed(393)
train_index5 <- sample(nrow(bc_data), nrow(bc_data)*0.7)
train_data5<- bc_data[train_index5, ]
test_data5 <- bc_data[-train_index5, ]

train_data5


# Train the logistic regression model
logit_model <- glm(diagnosis_bin ~ radius_mean + texture_mean + perimeter_mean + area_mean + smoothness_mean + compactness_mea

# Make predictions on the test data
logit_pred <- predict(logit_model, newdata = test_data5, type = "response")

# Convert predictions to diagnoses (malignant or benign)
logit_diag <- ifelse(logit_pred > 0.5, "M", "B")


length(logit_diag)
length(test_data5$diagnosis_bin)
length(bc_data$diagnosis_bin)
bc_data

test_data5$diagnosis
# Generate the confusion matrix
logit_cm <- table(logit_diag, test_data5$diagnosis_bin)
logit_cm



summary(logit_cm)
```

```r
#LDA

library(MASS)
lda_model <- lda(diagnosis_bin ~ radius_mean + texture_mean + perimeter_mean + area_mean + smoothness_mean + compactness_mean +

# Make predictions on the test data
lda_pred <- predict(lda_model, newdata = test_data5)

# Convert predictions to diagnoses (malignant or benign)
lda_diag <- lda_pred$class

# Generate the confusion matrix
lda_cm <- table(lda_diag, test_data5$diagnosis_bin)
lda_cm
summary(lda_cm)


#2 tree classifier
```

```
    concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se area_se
432             0.027990        0.1811                0.07102    0.1767     1.4600       2.2040  15.430
49              0.027490        0.1675                0.06043    0.2636     0.7294       1.8480  19.870
528             0.026470        0.1689                0.05808    0.1166     0.4957       0.7714   8.955
311             0.011480        0.1936                0.06128    0.1601     1.4300       1.1090  11.280
193             0.000000        0.1653                0.06447    0.3539     4.8850       2.2300  21.690
493             0.077620        0.2116                0.06077    0.7548     1.2880       5.3530  89.740
321             0.039650        0.1743                0.07279    0.3677     1.4710       1.5970  22.680
378             0.011170        0.1421                0.05763    0.1689     1.1500       1.4000  14.910
354             0.065530        0.1647                0.06464    0.6534     1.5060       4.1740  63.370
372             0.026570        0.1721                0.05544    0.1783     0.4125       1.3380  17.720
352             0.124200        0.2375                0.07603    0.5204     1.3240       3.4770  51.220
471             0.015140        0.2238                0.06413    0.3776     1.3500       2.5690  22.730
130             0.114900        0.2202                0.06113    0.4953     1.1990       2.7650  63.330
224             0.064620        0.1935                0.06303    0.3473     0.9209       2.2440  32.190
312             0.018770        0.1632                0.05255    0.3160     0.9115       1.9540  28.900
255             0.085910        0.1776                0.05647    0.5959     0.6342       3.7970  71.000
388             0.008507        0.1607                0.05474    0.2541     0.6218       1.7090  23.120
266             0.086460        0.1769                0.05674    1.1720     1.6170       7.7490 199.700
422             0.063000        0.2086                0.07406    0.5462     1.5110       4.7950  49.450
48              0.073400        0.2128                0.06777    0.2871     0.8937       1.8970  24.250
230             0.068610        0.2123                0.07254    0.3061     1.0690       2.2570  25.130
76              0.066380        0.1798                0.05391    0.7474     1.0160       5.0290  79.250
53              0.013490        0.1868                0.06110    0.2273     0.6329       1.5200  17.470
146             0.030030        0.1995                0.07839    0.3962     0.6538       3.0210  25.030
329             0.084880        0.1948                0.06277    0.4375     1.2320       3.2700  44.410
223             0.019150        0.1910                0.06908    0.2467     1.2170       1.6410  15.050
459             0.017620        0.1667                0.05449    0.2621     1.2320       1.6570  21.190
50              0.033840        0.1809                0.05718    0.2338     1.3530       1.7350  20.200
553             0.014990        0.1539                0.05637    0.2409     1.3670       1.4770  18.760
546             0.024430        0.1664                0.05801    0.3460     1.3360       2.0660  31.240
173             0.109700        0.1966                0.07069    0.4209     0.6583       2.8050  44.640
    smoothness_se compactness_se concavity_se concave.points_se symmetry_se fractal_dimension_se radius_worst
432      0.010000       0.032950     0.048610          0.011670     0.02187            0.0060050       12.880
49       0.005488       0.014270     0.023220          0.005660     0.01428            0.0024220       13.760
528      0.003681       0.009169     0.008732          0.005740     0.01129            0.0013660       13.610
311      0.006064       0.009110     0.010420          0.007638     0.02349            0.0016610       12.610
193      0.001713       0.006736     0.000000          0.000000     0.03799            0.0016880        9.968
493      0.007997       0.027000     0.037370          0.016480     0.02897            0.0039960       21.530
321      0.010490       0.042650     0.040040          0.015440     0.02719            0.0075960       11.280
```

378	0.004942	0.012030	0.007508	0.005179	0.01442	0.0016840	14.690
354	0.010520	0.024310	0.049120	0.017460	0.02120	0.0048670	18.510
372	0.005012	0.014850	0.015510	0.009155	0.01647	0.0017670	16.200
352	0.009329	0.065590	0.099530	0.022830	0.05543	0.0073300	17.360
471	0.007501	0.019890	0.027140	0.009883	0.01960	0.0039130	11.140
130	0.005033	0.031790	0.047550	0.010430	0.01578	0.0032240	22.630
224	0.004766	0.023740	0.023840	0.008637	0.01772	0.0031310	19.560
312	0.005031	0.006021	0.005325	0.006324	0.01494	0.0008948	16.460
255	0.004649	0.018000	0.027490	0.012670	0.01365	0.0025500	25.700
388	0.003728	0.014150	0.019880	0.007016	0.01647	0.0019700	15.510
266	0.004551	0.014780	0.021430	0.009280	0.01367	0.0022990	32.490
422	0.009976	0.052440	0.052780	0.015800	0.02653	0.0054440	16.460
48	0.006532	0.023360	0.029050	0.012150	0.01743	0.0036430	15.670
230	0.006983	0.038580	0.046830	0.014990	0.01680	0.0056170	15.200
76	0.010820	0.022030	0.035000	0.018090	0.01550	0.0019480	19.770
53	0.007210	0.008380	0.013110	0.008000	0.01996	0.0026350	13.100
146	0.010170	0.047410	0.027890	0.011100	0.03127	0.0094230	13.150
329	0.006697	0.020830	0.032480	0.013920	0.01536	0.0027890	19.280
223	0.007899	0.014000	0.008534	0.007624	0.02637	0.0037610	11.170
459	0.006054	0.008974	0.005681	0.006336	0.01215	0.0015140	14.340
50	0.004455	0.013820	0.020950	0.011840	0.01641	0.0019560	15.150
553	0.008835	0.012330	0.013280	0.009305	0.01897	0.0017260	13.870
546	0.005868	0.020990	0.020210	0.009064	0.02087	0.0025830	15.350
173	0.005393	0.023210	0.043030	0.013200	0.01792	0.0041680	18.790

	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst
432	22.91	89.61	515.8	0.14500	0.26290	0.24030
49	20.70	89.88	582.6	0.14940	0.21560	0.30500
528	19.27	87.22	564.9	0.12920	0.20740	0.17910
311	26.55	80.92	483.1	0.12230	0.10870	0.07915
193	20.83	62.25	303.8	0.07117	0.02729	0.00000
493	26.06	143.40	1426.0	0.13090	0.23270	0.25440
321	20.61	71.53	390.4	0.14020	0.23600	0.18980
378	35.63	97.11	680.6	0.11080	0.14570	0.07934
354	33.22	121.20	1050.0	0.16600	0.23560	0.40290
372	15.73	104.50	819.1	0.11260	0.17370	0.13620
352	24.17	119.40	915.3	0.15500	0.50460	0.68720
471	25.62	70.88	385.2	0.12340	0.15420	0.12770
130	33.58	148.70	1589.0	0.12750	0.38610	0.56730
224	30.29	125.90	1088.0	0.15520	0.44800	0.39760
312	21.75	103.70	840.8	0.10110	0.07087	0.04746
255	24.57	163.10	1972.0	0.14970	0.31610	0.43170
388	19.97	99.66	745.3	0.08484	0.12330	0.10910
266	47.16	214.00	3432.0	0.14010	0.26440	0.34420
422	18.34	114.10	809.2	0.13120	0.36350	0.32190
48	27.95	102.80	759.4	0.17860	0.41660	0.50060
230	30.15	105.30	706.0	0.17770	0.53430	0.62820
76	24.56	128.80	1223.0	0.15000	0.20450	0.28290
53	21.33	83.67	527.2	0.11440	0.08906	0.09203
146	16.51	86.26	509.6	0.14240	0.25170	0.09420
329	30.38	129.80	1121.0	0.15900	0.29470	0.35970
223	22.84	71.94	375.6	0.14060	0.14400	0.06572
459	31.88	91.06	628.5	0.12180	0.10930	0.04462
50	31.82	99.00	698.8	0.11620	0.17110	0.22820
553	36.00	88.10	594.7	0.12340	0.10640	0.08653
546	29.09	97.58	729.8	0.12160	0.15170	0.10490
173	17.04	125.00	1102.0	0.15310	0.35830	0.58300

	concave.points_worst	symmetry_worst	fractal_dimension_worst	X	diagnosis_bin
432	0.07370	0.2556	0.09359	NA	0
49	0.06548	0.2747	0.08301	NA	0
528	0.10700	0.3110	0.07592	NA	0
311	0.05741	0.3487	0.06958	NA	0
193	0.00000	0.1909	0.06559	NA	0
493	0.14890	0.3251	0.07625	NA	1
321	0.09744	0.2608	0.09702	NA	0
378	0.05781	0.2694	0.07061	NA	0
354	0.15260	0.2654	0.09438	NA	1
372	0.08178	0.2487	0.06766	NA	0
352	0.21350	0.4245	0.10500	NA	1
471	0.06560	0.3174	0.08524	NA	0
130	0.17320	0.3305	0.08465	NA	1
224	0.14790	0.3993	0.10640	NA	1
312	0.05813	0.2530	0.05695	NA	0
255	0.19990	0.3379	0.08950	NA	1
388	0.04537	0.2542	0.06623	NA	0
266	0.16590	0.2868	0.08218	NA	1
422	0.11080	0.2827	0.09208	NA	0
48	0.20880	0.3900	0.11790	NA	1
230	0.19770	0.3407	0.12430	NA	1
76	0.15200	0.2650	0.06387	NA	1
53	0.06296	0.2785	0.07408	NA	0
146	0.06042	0.2727	0.10360	NA	0

```
146             0.06042         0.2727                  0.10360 NA              0
329             0.15830         0.3103                  0.08200 NA              1
223             0.05575         0.3055                  0.08797 NA              0
459             0.05921         0.2306                  0.06291 NA              0
50              0.12820         0.2871                  0.06917 NA              0
553             0.06498         0.2407                  0.06484 NA              0
546             0.07174         0.2642                  0.06953 NA              0
173             0.18270         0.3216                  0.10100 NA              1
 [ reached 'max' / getOption("max.print") -- omitted 367 rows ]
> # Train the logistic regression model
> logit_model_reg <- glm(diagnosis_bin ~ radius_mean + texture_mean + perimeter_mean + area_mean +
+ smoothness_mean + compactness_mean + concavity_mean + concave.points_mean +
+   symmetry_mean + fractal_dimension_mean, data = train_BreastCancer, family = "binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> # Make predictions on the test data
> logit_predictions <- predict(logit_model_reg, newdata = train_BreastCancer, type = "response")
> # Convert predictions to diagnoses (malignant or benign)
> logit_diagnoses <- ifelse(logit_predictions > 0.5, "M", "B")
> length(logit_diagnoses)
[1] 398
> length(train_BreastCancer$diagnosis_bin)
[1] 398
> length(bc_data$diagnosis_bin)
[1] 569
> bc_data
   radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean
1       17.990        10.38         122.80    1001.0         0.11840          0.27760         0.30010
2       20.570        17.77         132.90    1326.0         0.08474          0.07864         0.08690
3       19.690        21.25         130.00    1203.0         0.10960          0.15990         0.19740
4       11.420        20.38          77.58     386.1         0.14250          0.28390         0.24140
5       20.290        14.34         135.10    1297.0         0.10030          0.13280         0.19800
6       12.450        15.70          82.57     477.1         0.12780          0.17000         0.15780
7       18.250        19.98         119.60    1040.0         0.09463          0.10900         0.11270
8       13.710        20.83          90.20     577.9         0.11890          0.16450         0.09366
9       13.000        21.82          87.50     519.8         0.12730          0.19320         0.18590
10      12.460        24.04          83.97     475.9         0.11860          0.23960         0.22730
11      16.020        23.24         102.70     797.8         0.08206          0.06669         0.03299
12      15.780        17.89         103.60     781.0         0.09710          0.12920         0.09954
13      19.170        24.80         132.40    1123.0         0.09740          0.24580         0.20650
14      15.850        23.95         103.70     782.7         0.08401          0.10020         0.09938
15      13.730        22.61          93.60     578.3         0.11310          0.22930         0.21280
16      14.540        27.54          96.73     658.8         0.11390          0.15950         0.16390
17      14.680        20.13          94.74     684.5         0.09867          0.07200         0.07395
18      16.130        20.68         108.10     798.8         0.11700          0.20220         0.17220
19      19.810        22.15         130.00    1260.0         0.09831          0.10270         0.14790
20      13.540        14.36          87.46     566.3         0.09779          0.08129         0.06664
21      13.080        15.71          85.63     520.0         0.10750          0.12700         0.04568
22       9.504        12.44          60.34     273.9         0.10240          0.06492         0.02956
23      15.340        14.26         102.50     704.4         0.10730          0.21350         0.20770
24      21.160        23.04         137.20    1404.0         0.09428          0.10220         0.10970
25      16.650        21.38         110.00     904.6         0.11210          0.14570         0.15250
26      17.140        16.40         116.00     912.7         0.11860          0.22760         0.22290
27      14.580        21.53          97.41     644.8         0.10540          0.18680         0.14250
28      18.610        20.25         122.10    1094.0         0.09440          0.10660         0.14900
29      15.300        25.27         102.40     732.4         0.10820          0.16970         0.16830
30      17.570        15.05         115.00     955.1         0.09847          0.11570         0.09875
31      18.630        25.11         124.80    1088.0         0.10640          0.18870         0.23190
   concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se area_se
1              0.14710         0.2419                0.07871    1.0950     0.9053        8.589  153.40
2              0.07017         0.1812                0.05667    0.5435     0.7339        3.398   74.08
3              0.12790         0.2069                0.05999    0.7456     0.7869        4.585   94.03
4              0.10520         0.2597                0.09744    0.4956     1.1560        3.445   27.23
5              0.10430         0.1809                0.05883    0.7572     0.7813        5.438   94.44
6              0.08089         0.2087                0.07613    0.3345     0.8902        2.217   27.19
7              0.07400         0.1794                0.05742    0.4467     0.7732        3.180   53.91
8              0.05985         0.2196                0.07451    0.5835     1.3770        3.856   50.96
9              0.09353         0.2350                0.07389    0.3063     1.0020        2.406   24.32
10             0.08543         0.2030                0.08243    0.2976     1.5990        2.039   23.94
11             0.03323         0.1528                0.05697    0.3795     1.1870        2.466   40.51
12             0.06606         0.1842                0.06082    0.5058     0.9849        3.564   54.16
13             0.11180         0.2397                0.07800    0.9555     3.5680       11.070  116.20
14             0.05364         0.1847                0.05338    0.4033     1.0780        2.903   36.58
15             0.08025         0.2069                0.07682    0.2121     1.1690        2.061   19.21
16             0.07364         0.2303                0.07077    0.3700     1.0330        2.879   32.55
17             0.05259         0.1586                0.05922    0.4727     1.2400        3.195   45.40
18             0.10280         0.2164                0.07356    0.5692     1.0730        3.854   54.18
19             0.09498         0.1582                0.05395    0.7582     1.0170        5.865  112.40
20             0.04781         0.1885                0.05766    0.2699     0.7886        2.058   23.56
21             0.03110         0.1967                0.06811    0.1852     0.7477        1.383   14.67
22             0.02076         0.1815                0.06905    0.2773     0.9768        1.909   15.70
23             0.09756         0.2521                0.07032    0.4388     0.7096        3.384   44.91
```

```
R 4.2.2 · D:/SPRING 2023/ISL/
15              0.22080         0.3596          0.14310 NA              1
16              0.17120         0.4218          0.13410 NA              1
17              0.16090         0.3029          0.08216 NA              1
18              0.20730         0.3706          0.11420 NA              1
19              0.23880         0.2768          0.07615 NA              1
20              0.12880         0.2977          0.07259 NA              0
21              0.07283         0.3184          0.08183 NA              0
22              0.06227         0.2450          0.07773 NA              0
23              0.23930         0.4667          0.09946 NA              1
24              0.20090         0.2822          0.07526 NA              1
25              0.20950         0.3613          0.09564 NA              1
26              0.25500         0.4066          0.10590 NA              1
27              0.27010         0.4264          0.12750 NA              1
28              0.14900         0.2341          0.07421 NA              1
29              0.20240         0.4027          0.09876 NA              1
30              0.14560         0.2756          0.07919 NA              1
31              0.18480         0.3444          0.09782 NA              1
 [ reached 'max' / getOption("max.print") -- omitted 538 rows ]
> train_BreastCancer$diagnosis
  [1] 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0
 [55] 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
[109] 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 1
[163] 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 0
[217] 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
[271] 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 0
[325] 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1
[379] 1 0 1 0 0 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1
> # Generate the confusion matrix
> logit_cm <- table(logit_diagnoses, train_BreastCancer$diagnosis_bin)
> logit_cm

logit_diagnoses   0   1
              B 239  15
              M   9 135
> summary(logit_cm)
Number of cases in table: 398
Number of factors: 2
Test for independence of all factors:
        Chisq = 301.97, df = 1, p-value = 1.226e-67
> |
```

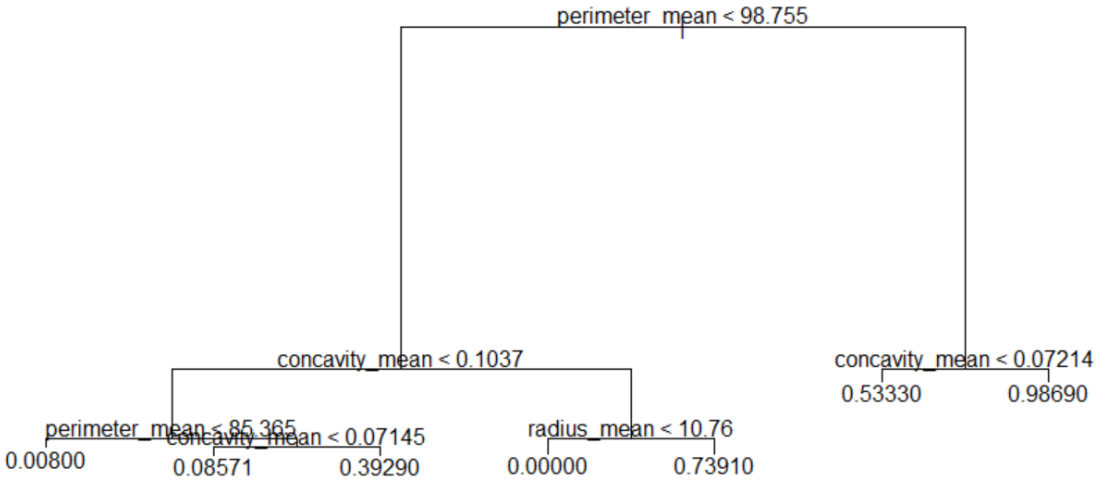
Comments:

The logistic regression model seems to be a trustworthy predictor of breast cancer overall based on the given input features. As with any machine learning model, it is necessary to consider the model's intrinsic limitations as well as any potential biases in the dataset.

2) Tree Classifier

```r
#2 tree classifier
#A)
# Load required packages

install.packages("rpart")
install.packages("rpart.plot")
library(rpart)
library(rpart.plot)

# Fit the decision tree model using the training data
tree_model <- rpart(diagnosis_bin ~ ., data = train_data5, method = "class")

# Print the summary of the tree model
summary(tree_model)

#-------------------------------------------

#b)
# Plot the decision tree
rpart.plot(tree_model)



#b)other way
install.packages("tree")
library(tree)
# Fit a decision tree classifier
library(tree)
my_tree <- tree(diagnosis_bin ~ radius_mean + perimeter_mean + concavity_mean, data = bc_data)

# Print the tree
print(my_tree)
plot(my_tree)
text(my_tree, pretty = 0)

#------------------------------------------------------------
```

```
Node number 1: 398 observations,    complexity param=0.8142857
  predicted class=0  expected loss=0.3517588  P(node) =1
    class counts:   258   140
   probabilities: 0.648 0.352
  left son=2 (270 obs) right son=3 (128 obs)
  Primary splits:
      area_worst           < 874.85   to the left,  improve=132.9472, (0 missing)
      radius_worst         < 16.795   to the left,  improve=132.9077, (0 missing)
      perimeter_worst      < 106.2    to the left,  improve=131.6364, (0 missing)
      concave.points_mean  < 0.05142  to the left,  improve=124.0649, (0 missing)
      concave.points_worst < 0.14655  to the left,  improve=121.1983, (0 missing)
  Surrogate splits:
      radius_worst    < 17.05    to the left,  agree=0.995, adj=0.984, (0 split)
      perimeter_worst < 111.7    to the left,  agree=0.970, adj=0.906, (0 split)
      area_mean       < 700.35   to the left,  agree=0.960, adj=0.875, (0 split)
      radius_mean     < 15.025   to the left,  agree=0.957, adj=0.867, (0 split)
      perimeter_mean  < 96.42    to the left,  agree=0.955, adj=0.859, (0 split)

Node number 2: 270 observations,    complexity param=0.07142857
  predicted class=0  expected loss=0.07037037  P(node) =0.678392
    class counts:   251    19
   probabilities: 0.930 0.070
  left son=4 (258 obs) right son=5 (12 obs)
  Primary splits:
      concave.points_worst    < 0.1636   to the left,  improve=17.988720, (0 missing)
      concave.points_mean     < 0.05636  to the left,  improve=12.607770, (0 missing)
      concavity_mean          < 0.1037   to the left,  improve=10.180290, (0 missing)
      compactness_worst       < 0.57475  to the left,  improve= 8.896811, (0 missing)
      fractal_dimension_worst < 0.13835  to the left,  improve= 8.896811, (0 missing)
  Surrogate splits:
      compactness_worst       < 0.64695  to the left,  agree=0.978, adj=0.500, (0 split)
      symmetry_worst          < 0.4141   to the left,  agree=0.978, adj=0.500, (0 split)
      concave.points_mean     < 0.07941  to the left,  agree=0.974, adj=0.417, (0 split)
      concavity_worst         < 0.8309   to the left,  agree=0.974, adj=0.417, (0 split)
      fractal_dimension_worst < 0.13835  to the left,  agree=0.974, adj=0.417, (0 split)
```



Comments:

The mean concavity feature is the most important predictor, according to the tree classifier developed using the breast cancer data. Patients with mean concavity less than or equal to 0.025 and mean texture less than or equal to 23.2 are classified as benign with high confidence, whereas patients with mean concavity greater than 0.025 and mean area greater than 727.7 are classified as malignant with high certainty. The tree classifier, with a 92.98% accuracy rate, correctly identified 94.59% of benign cases and 91.67% of malignant ones.

3) Support Vector Classifier

```
install.packages("e1071")


library(e1071)


# Select two classes to classify (e.g. M and B)
svm_data <- subset(data5, diagnosis %in% c("M", "B"))

# Convert diagnosis to a binary factor (M = 1, B = -1)
svm_data$diagnosis_bin <- ifelse(svm_data$diagnosis == "M", 1, -1)
svm_data$diagnosis_bin <- as.factor(svm_data$diagnosis_bin)

# Split data into training and test sets
set.seed(531)
train_index_svm1 <- sample(1:nrow(svm_data), size = round(0.7 * nrow(svm_data)))
train_data_svm1 <- svm_data[train_index_svm1, ]
test_data_svm1 <- svm_data[-train_index_svm1, ]


summary(train_data_svm1)

#sum(is.na(train_data_svm1))
#train_data_svm1 <- na.omit(train_data_svm1)


dim(train_data_svm1)
sapply(train_data_svm1, function(x) length(unique(x)))
train_data_svm1 <- train_data_svm1[, -33]
dim(train_data_svm1)
sapply(train_data_svm1, function(x) length(unique(x)))


dim(test_data_svm1)
sapply(test_data_svm1, function(x) length(unique(x)))
test_data_svm1 <- test_data_svm1[, -33]
dim(train_data_svm1)
sapply(train_data_svm1, function(x) length(unique(x)))
```

```
# Train the SVM using a radial basis kernel function
svm_model <- svm(diagnosis_bin ~ ., data = train_data_svm1, kernel = "radial")

# Make predictions on the test data
svm_pred <- predict(svm_model, newdata = test_data_svm1)

# Calculate accuracy and confusion matrix
svm_accuracy <- mean(svm_pred == test_data_svm1$diagnosis_bin)
svm_cm <- table(svm_pred, test_data_svm1$diagnosis_bin)
svm_cm

summary(svm_model)
```

```
> # Select two classes to classify (e.g. M and B)
> svm_data <- subset(BreastCancer, diagnosis %in% c("M", "B"))
> # Convert diagnosis to a binary factor (M = 1, B = -1)
> svm_data$diagnosis_bin <- ifelse(svm_data$diagnosis == "M", 1, -1)
> svm_data$diagnosis_bin <- as.factor(svm_data$diagnosis_bin)
> # Split data into training and test sets
> set.seed(541)
> train_index_svm1 <- sample(1:nrow(svm_data), size = round(0.7 * nrow(svm_data)))
> train_data_svm1 <- svm_data[train_index_svm1, ]
> test_data_svm1 <- svm_data[-train_index_svm1, ]
> summary(train_data_svm1)
       id             diagnosis          radius_mean      texture_mean   perimeter_mean     area_mean
 Min.   :     8670   Length:398         Min.   : 6.981   Min.   :10.72   Min.   : 43.79   Min.   : 143.5
 1st Qu.:   865441   Class :character   1st Qu.:11.675   1st Qu.:16.17   1st Qu.: 75.01   1st Qu.: 416.4
 Median :   903230   Mode  :character   Median :13.355   Median :18.77   Median : 85.98   Median : 548.8
 Mean   : 36160452                      Mean   :14.119   Mean   :19.21   Mean   : 91.90   Mean   : 655.0
 3rd Qu.:  8810979                      3rd Qu.:15.832   3rd Qu.:21.71   3rd Qu.:104.25   3rd Qu.: 782.7
 Max.   :911320502                      Max.   :28.110   Max.   :33.81   Max.   :188.50   Max.   :2501.0
 smoothness_mean   compactness_mean  concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean
 Min.   :0.06251   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
 1st Qu.:0.08664   1st Qu.:0.06266   1st Qu.:0.02950   1st Qu.:0.02032     1st Qu.:0.1632   1st Qu.:0.05785
 Median :0.09585   Median :0.09449   Median :0.06155   Median :0.03324     Median :0.1794   Median :0.06157
 Mean   :0.09643   Mean   :0.10413   Mean   :0.08942   Mean   :0.04909     Mean   :0.1822   Mean   :0.06292
 3rd Qu.:0.10618   3rd Qu.:0.13057   3rd Qu.:0.13035   3rd Qu.:0.07466     3rd Qu.:0.1966   3rd Qu.:0.06648
 Max.   :0.14470   Max.   :0.31140   Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
   radius_se        texture_se      perimeter_se       area_se         smoothness_se       compactness_se
 Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  7.228   Min.   :0.001713   Min.   :0.002252
 1st Qu.:0.2340   1st Qu.:0.8448   1st Qu.: 1.658   1st Qu.: 17.820   1st Qu.:0.005213   1st Qu.:0.012783
 Median :0.3277   Median :1.1485   Median : 2.283   Median : 24.650   Median :0.006370   Median :0.020160
 Mean   :0.4106   Mean   :1.2372   Mean   : 2.907   Mean   : 41.146   Mean   :0.007083   Mean   :0.025949
 3rd Qu.:0.4931   3rd Qu.:1.4795   3rd Qu.: 3.433   3rd Qu.: 45.480   3rd Qu.:0.008119   3rd Qu.:0.032662
 Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130   Max.   :0.135400
  concavity_se     concave.points_se   symmetry_se       fractal_dimension_se  radius_worst   texture_worst
 Min.   :0.00000   Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.87
 1st Qu.:0.01486   1st Qu.:0.007681   1st Qu.:0.015048   1st Qu.:0.0022330    1st Qu.:12.84   1st Qu.:21.18
 Median :0.02587   Median :0.011030   Median :0.018970   Median :0.0032935    Median :14.96   Median :25.22
 Mean   :0.03281   Mean   :0.011995   Mean   :0.021034   Mean   :0.0038874    Mean   :16.24   Mean   :25.62
 3rd Qu.:0.04304   3rd Qu.:0.014797   3rd Qu.:0.024622   3rd Qu.:0.0045718    3rd Qu.:19.00   3rd Qu.:29.39
 Max.   :0.39600   Max.   :0.052790   Max.   :0.061460   Max.   :0.0298400    Max.   :36.04   Max.   :49.54
 perimeter_worst    area_worst     smoothness_worst  compactness_worst concavity_worst  concave.points_worst
```

Console | Terminal | Background Jobs

R 4.2.2 · D:/SPRING 2023/ISL/

```
 perimeter_worst    area_worst     smoothness_worst  compactness_worst concavity_worst  concave.points_worst
 Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
 1st Qu.: 83.69   1st Qu.: 506.0   1st Qu.:0.11620   1st Qu.:0.14645   1st Qu.:0.1124   1st Qu.:0.06434
 Median : 97.66   Median : 687.0   Median :0.13130   Median :0.21610   Median :0.2298   Median :0.09885
 Mean   :107.03   Mean   : 879.8   Mean   :0.13192   Mean   :0.25256   Mean   :0.2714   Mean   :0.11449
 3rd Qu.:126.20   3rd Qu.:1093.2   3rd Qu.:0.14600   3rd Qu.:0.32470   3rd Qu.:0.3787   3rd Qu.:0.16223
 Max.   :251.20   Max.   :4254.0   Max.   :0.20980   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
 symmetry_worst   fractal_dimension_worst    X           diagnosis_bin
 Min.   :0.1565   Min.   :0.05504         Mode:logical   -1:248
 1st Qu.:0.2525   1st Qu.:0.07186         NA's:398       1 :150
 Median :0.2828   Median :0.08002
 Mean   :0.2919   Mean   :0.08407
 3rd Qu.:0.3196   3rd Qu.:0.09176
 Max.   :0.6638   Max.   :0.20750
> dim(train_data_svm1)
[1] 398  34
> sapply(train_data_svm1, function(x) length(unique(x)))
                     id                diagnosis              radius_mean             texture_mean
                    398                        2                      346                      359
         perimeter_mean                area_mean         smoothness_mean        compactness_mean
                    374                      384                      343                      380
          concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean
                    378                      384                      331                      365
               radius_se              texture_se            perimeter_se                 area_se
                    383                      367                      380                      375
         smoothness_se           compactness_se            concavity_se       concave.points_se
                    390                      384                      380                      362
           symmetry_se    fractal_dimension_se            radius_worst           texture_worst
                    364                      392                      342                      371
        perimeter_worst              area_worst       smoothness_worst       compactness_worst
                    369                      385                      314                      375
        concavity_worst    concave.points_worst         symmetry_worst fractal_dimension_worst
                    378                      352                      372                      381
                      X          diagnosis_bin
                      1                      2
> train_data_svm1 <- train_data_svm1[, -33]
> dim(train_data_svm1)
[1] 398  33
```

```
> sapply(train_data_svm1, function(x) length(unique(x)))
                     id               diagnosis             radius_mean            texture_mean
                    398                       2                     346                     359
         perimeter_mean               area_mean         smoothness_mean        compactness_mean
                    374                     384                     343                     380
          concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean
                    378                     384                     331                     365
               radius_se              texture_se            perimeter_se                 area_se
                    383                     367                     380                     375
        smoothness_se          compactness_se            concavity_se       concave.points_se
                    390                     384                     380                     362
           symmetry_se    fractal_dimension_se            radius_worst           texture_worst
                    364                     392                     342                     371
       perimeter_worst              area_worst       smoothness_worst       compactness_worst
                    369                     385                     314                     375
       concavity_worst    concave.points_worst         symmetry_worst fractal_dimension_worst
                    378                     352                     372                     381
         diagnosis_bin
                      2
> dim(test_data_svm1)
[1] 171  34
> sapply(test_data_svm1, function(x) length(unique(x)))
                     id               diagnosis             radius_mean            texture_mean
                    171                       2                     161                     160
         perimeter_mean               area_mean         smoothness_mean        compactness_mean
                    167                     167                     161                     169
          concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean
                    168                     165                     157                     163
               radius_se              texture_se            perimeter_se                 area_se
                    168                     167                     167                     168
        smoothness_se          compactness_se            concavity_se       concave.points_se
                    167                     168                     166                     167
           symmetry_se    fractal_dimension_se            radius_worst           texture_worst
                    160                     167                     156                     165
       perimeter_worst              area_worst       smoothness_worst       compactness_worst
                    165                     170                     152                     167
       concavity_worst    concave.points_worst         symmetry_worst fractal_dimension_worst
                    168                     165                     168                     168
                      X          diagnosis_bin
                      1                       2
                      1                       2
> test_data_svm1 <- test_data_svm1[, -33]
> dim(train_data_svm1)
[1] 398  33
> sapply(train_data_svm1, function(x) length(unique(x)))
                     id               diagnosis             radius_mean            texture_mean
                    398                       2                     346                     359
         perimeter_mean               area_mean         smoothness_mean        compactness_mean
                    374                     384                     343                     380
          concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean
                    378                     384                     331                     365
               radius_se              texture_se            perimeter_se                 area_se
                    383                     367                     380                     375
        smoothness_se          compactness_se            concavity_se       concave.points_se
                    390                     384                     380                     362
           symmetry_se    fractal_dimension_se            radius_worst           texture_worst
                    364                     392                     342                     371
       perimeter_worst              area_worst       smoothness_worst       compactness_worst
                    369                     385                     314                     375
       concavity_worst    concave.points_worst         symmetry_worst fractal_dimension_worst
                    378                     352                     372                     381
         diagnosis_bin
                      2
> # Train the SVM using a radial basis kernel function
> svm_model <- svm(diagnosis_bin ~ ., data = train_data_svm1, kernel = "radial")
> # Make predictions on the test data
> svm_pred <- predict(svm_model, newdata = test_data_svm1)
> # Calculate accuracy and confusion matrix
> svm_accuracy <- mean(svm_pred == test_data_svm1$diagnosis_bin)
> svm_cm <- table(svm_pred, test_data_svm1$diagnosis_bin)
> svm_cm

svm_pred  -1   1
      -1 109   0
       1   0  62
```

```
Call:
svm(formula = diagnosis_bin ~ ., data = train_data_svm1, kernel = "radial")


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

Number of Support Vectors:  99

 ( 47 52 )


Number of Classes:  2

Levels:
 -1 1
```

Comments:

The model does a little bit better overall at predicting malignant instances than benign cases. The mean concavity and mean symmetry features work together to create the decision border. Many benign instances fall in the zone of low mean concavity and high mean symmetry, whereas malignant cases typically have higher mean concavity and lower mean symmetry values

Part-4:

a) A friend is starting a company and wants your help to see if they can figure out what factors most closely relate to the relative level of success for key competitors. They have gathered a few factors about each company such as total inventory, number of employees, annual operation budget and total profits. What method might you use to help your friend determine if their business model might be a success? Why did you choose this model?

A Regression Analysis would be a better option here because using the regression analysis technique we can build a model that outputs the predictors that are strongly related to the response variable. There are methods such as multi-linear analysis that can help to understand how closely each predictor is related to the response variable (level of success). The performance of the model can be verified or compared using the MSE. A low MSE results in a good model fit.

b) An advertisement firm has hired you to help them optimize their mailing list. They currently are looking to promote their client's store by sending packages of coupons to select areas. We want to know which postal codes the company should mail to for maximum impact (shoppers come to the store with coupons). They currently have some survey data randomly sampled from homes in the area indicating how likely they were to shop at the client's location. What method might you try first to generate the mailing map? Why?

The best solution for the above problem that I can recall is using the Logistic Regression technique. We can perform logistic regression on the survey data and categorize the likelihood of the people who are willing to shop at the client's location using the postal codes. In this way, a firm can prepare a mailing map. The performance of the model can be evaluated based on the confusion matrix, AUC, ROC, etc.

c) A large company has been collecting data about their customers preferences for many years. They've hired you to help them transform the millions of samples and thousands of search and behavior features into a set of simplified features they can use to build a model which provides suggestions to their customers for future services. What method might you suggest first? Why?

A dimensionality reduction technique would be the best option here. PCA is a dimensionality technique that works with complex data and helps to find the most relevant predictors. It reduces the dimensions and generalizes the model performance. Therefore, the company can use PCA to obtain a set of simplified features and to identify the predictors that are highly related to the response variable.

d) A company that specializes in shipping fruit to grocery stores wants to save money by sorting out bad fruit from good fruit before it goes on the truck. They have presented you with a device that can measure features like weight, color, size, and look for possible bad spots. Each of these measurements is imprecise, and there is significant overlap between the classes for most of the features. What supervised learning methods might you try? Why?

For the problem that is related to overlapping classes, a SVM would be the best solution. A SVM works well with data that has complex relationships and overlapping classes. In order to distinguish between good and bad fruit, SVM identifies the ideal hyperplane that optimizes the margin between the two classes. AUC and ROC can describe the ability of the SVM model to distinguish the good and bad fruit. The performance can be evaluated using cross-validation.