INTRODUCTION TO STATISTICAL LEARNING

5565-0004

FINAL PROJECT REPORT

KONETI SRI GANESH

16341531

SKY4G@UMKC.EDU

Section 1:

Datasets used-

- Life Expectancy dataset: https://www.kaggle.com/kumarajarshi/life-expectancy-who
- Wine Quality Dataset: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009
- Breast Cancer Wisconsin (Diagnostic) Dataset: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

The above 3 datasets are used to perform multiple tasks such as Computation and Representation of Models for Complex Data.

- The primary focus of Data Set 1 will be on a **regression** issue involving a series or sequence of data in two or more dimensions.
- A **feature selection** task will be the main emphasis of data set 2.
- **Classification** tasks will be the main emphasis of data set 3.

Section 2:

Part 1: REGRESSION

- Life Expectancy dataset: https://www.kaggle.com/kumarajarshi/life-expectancy-who

a) Linear Regression

A dependent variable and one or more independent variables can have a linear connection, and this relationship can be modeled statistically using linear regression. In order to forecast the values of the dependent variable based on the values of the independent variable(s), it entails fitting a straight line through a series of data points.

CODE :

Packages and loading data:

projfinal.R

Source on Save

```
library(readr)
library(dplyr)
library(ggplot2)
library(caret)


#reading the csv file life expectency data
data <- read_csv("C:\\Users\\ganes\\Documents\\ISL\\ISL_PROJ\\dataset_1\\Life Expectancy Data.csv")
data

#display top rows
head(data)

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

#split the data into training and testing sets using a 70/30 split
set.seed(531)
trainIndex <- createDataPartition(data$"Life expectancy", p = .7, list = FALSE)
training <- data[trainIndex,]
testing <- data[-trainIndex,]
training
testing

```

Linear regression model:

```
#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, da
summary(model_lm)
#output of the summary provides us with information on the coefficients
model_lm


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

#This plot shows how well the predicted values match up with the actual values. A perfect model would have all the points lying on the red lin

```

Results:

Reviewing the data:

Console | Terminal | Background Jobs

R 4.2.2 · ~/

```
> data
# A tibble: 2,938 × 22
   Country      Year Status     `Life expectancy` `Adult Mortality` `infant deaths` Alcohol `percentage expenditure`
   <chr>       <dbl> <chr>                  <dbl>             <dbl>           <dbl>   <dbl>                    <dbl>
 1 Afghanistan  2015 Developing              65                263              62    0.01                    71.3
 2 Afghanistan  2014 Developing              59.9              271              64    0.01                    73.5
 3 Afghanistan  2013 Developing              59.9              268              66    0.01                    73.2
 4 Afghanistan  2012 Developing              59.5              272              69    0.01                    78.2
 5 Afghanistan  2011 Developing              59.2              275              71    0.01                     7.10
 6 Afghanistan  2010 Developing              58.8              279              74    0.01                    79.7
 7 Afghanistan  2009 Developing              58.6              281              77    0.01                    56.8
 8 Afghanistan  2008 Developing              58.1              287              80    0.03                    25.9
 9 Afghanistan  2007 Developing              57.5              295              82    0.02                    10.9
10 Afghanistan  2006 Developing              57.3              295              84    0.03                    17.2
# ℹ 2,928 more rows
# ℹ 14 more variables: `Hepatitis B` <dbl>, Measles <dbl>, BMI <dbl>, `under-five deaths` <dbl>, Polio <dbl>,
#   `Total expenditure` <dbl>, Diphtheria <dbl>, `HIV/AIDS` <dbl>, GDP <dbl>, Population <dbl>,
#   `thinness  1-19 years` <dbl>, `thinness 5-9 years` <dbl>, `Income composition of resources` <dbl>,
#   Schooling <dbl>
# ℹ Use `print(n = ...)` to see more rows
> #display top rows
> head(data)
# A tibble: 6 × 22
  Country      Year Status     `Life expectancy` `Adult Mortality` `infant deaths` Alcohol `percentage expenditure`
  <chr>       <dbl> <chr>                  <dbl>             <dbl>           <dbl>   <dbl>                    <dbl>
1 Afghanistan  2015 Developing              65                263              62    0.01                    71.3
2 Afghanistan  2014 Developing              59.9              271              64    0.01                    73.5
3 Afghanistan  2013 Developing              59.9              268              66    0.01                    73.2
4 Afghanistan  2012 Developing              59.5              272              69    0.01                    78.2
```

Simple linear regression using lm() function:

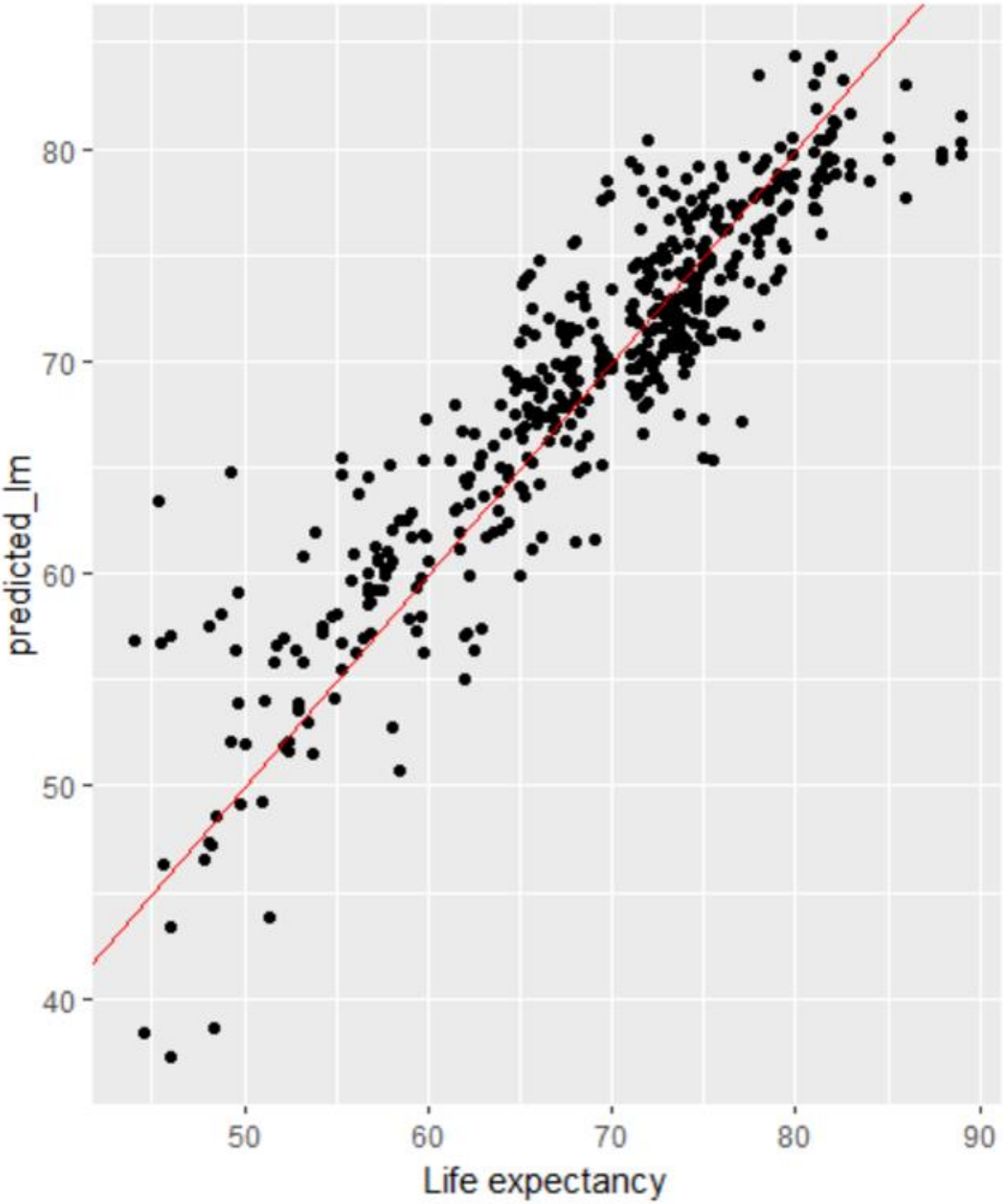
Console | Terminal | Background Jobs

R 4.2.2 · ~/

```
> #simple linear regression model using the lm() function
> model_lm <- lm(`Life expectancy` ~ `Adult Mortality` + Alcohol + BMI + `HIV/AIDS` + `Income composition of resources` + Schooling
training)
> summary(model_lm)

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

Residuals:
     Min       1Q   Median       3Q      Max
-12.9644  -2.2002  -0.0129   2.3163  11.2472

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                     55.023757   0.869875  63.255  < 2e-16 ***
`Adult Mortality`               -0.020157   0.001199 -16.810  < 2e-16 ***
Alcohol                         -0.106554   0.040279  -2.645 0.008271 **
BMI                              0.040277   0.006908   5.830 7.19e-09 ***
`HIV/AIDS`                      -0.416336   0.023012 -18.092  < 2e-16 ***
`Income composition of resources` 10.569576   0.964951  10.953  < 2e-16 ***
Schooling                        0.995960   0.070778  14.072  < 2e-16 ***
StatusDeveloping                -1.426304   0.411583  -3.465 0.000549 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.763 on 1149 degrees of freedom
Multiple R-squared:  0.8135,    Adjusted R-squared:  0.8124
F-statistic: 716.1 on 7 and 1149 DF,  p-value: < 2.2e-16
```

```
> #output of the summary provides us with information on the coefficients
> model_lm

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

Coefficients:
                      (Intercept)                `Adult Mortality`
                         55.02376                          -0.02016
                                                            Alcohol
                                                            -0.10655
                               BMI                         `HIV/AIDS`
                          0.04028                          -0.41634
`Income composition of resources`                        Schooling
                         10.56958                           0.99596
                  StatusDeveloping
                          -1.42630
```

visualize the model using a scatter plot.



Comments:

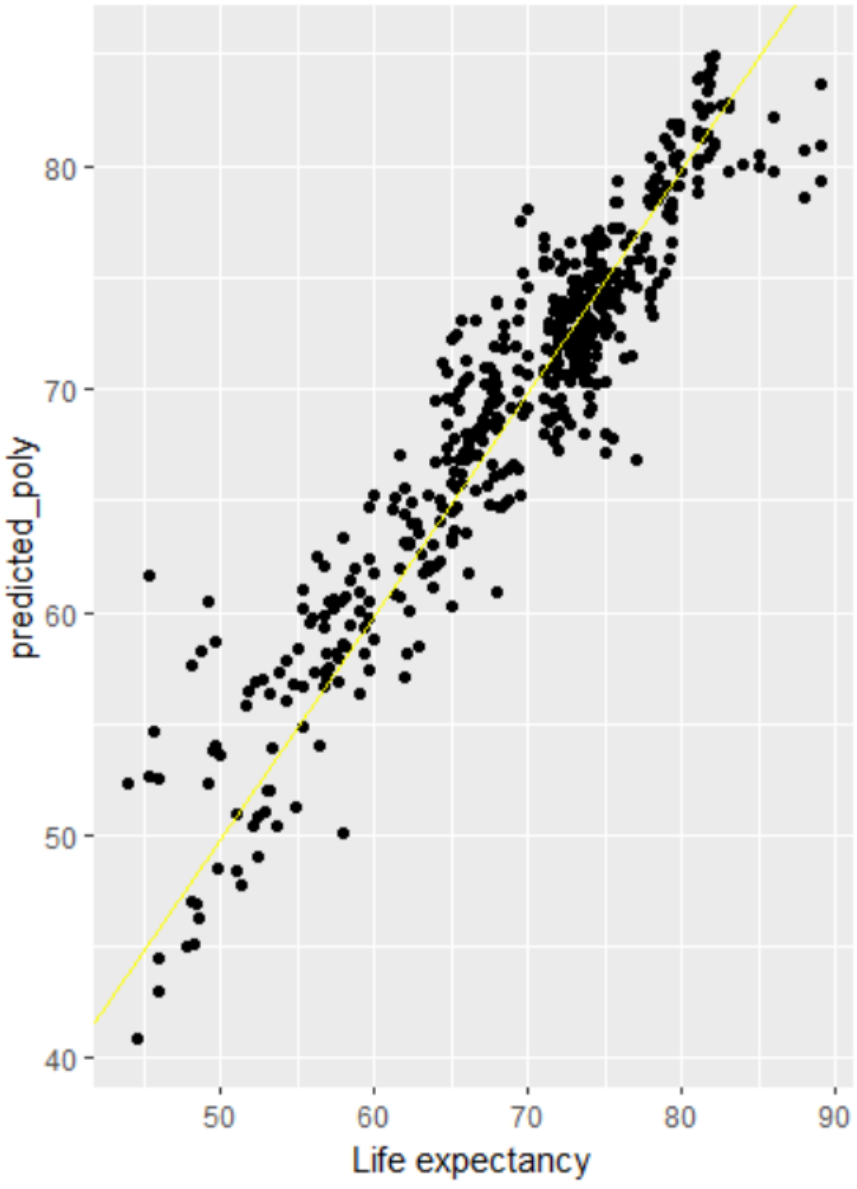
- The dataset consists of Adult Mortality Alcohol, BMI, HIV/AIDS, Income composition of resources, Schooling, Status (developed or developing). split the data into training and testing sets using a 70/30 split.
- simple linear regression model using the lm () function.
- The output of the summary provides us with information on the coefficients of the model, as well as the R-squared value which indicates how well the model fits the data.
- We can also visualize the model using a scatter plot of the predicted values against the actual values.
- The linear regression model has a good R-squared value of 0.812, indicating that about 81.29% of the variance in the dependent variable can be explained by the independent variables in the model. The mean absolute error of 3.76 means that on average, the model's predictions are off by about 3.089 years from the actual values. The RMSE of 3.763 is also relatively low, indicating that the model's predictions are generally accurate.

b) Polynomial Regression

CODE:

```
#b)Polynomial Regression
#we can try a polynomial regression model using the poly() function:
#The poly() function allows us to include polynomial terms in our model, which can help capture more complex relationships between the features an
model_poly <- lm(`Life expectancy` ~ poly(`Adult Mortality`, 2) + poly(Alcohol, 2) + poly(BMI, 2) + poly(`HIV/AIDS`, 2) + poly(`Income composition

summary(model_poly)


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

Results:



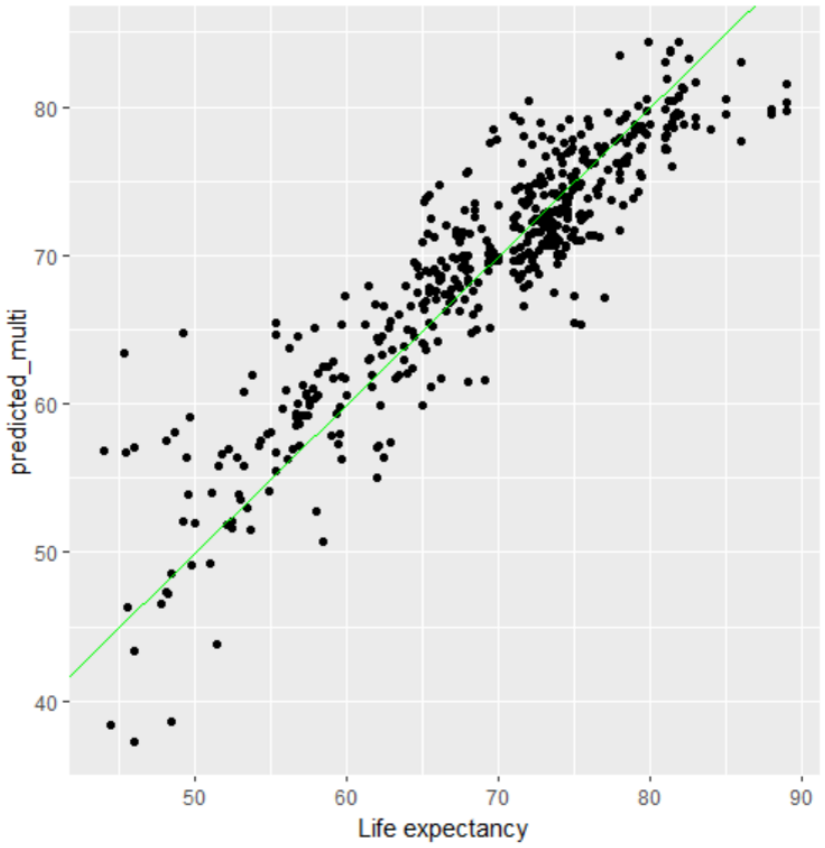
Comments:

- polynomial regression model using the poly () function.
- The poly () function allows us to include polynomial terms in our model, which can help capture more complex relationships between the features and the response.
- From the above results, we can state that the more of the variance in the dependent variable than the linear regression model.

c) Multilinear Regression

```
#c)Multilinear Regression
model_multi <- lm(`Life expectancy` ~ `Adult Mortality` + Alcohol + BMI + `HIV/AIDS` + `Income composition of resources` + Schooling + S

summary(model_multi)


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

Results



Comments:

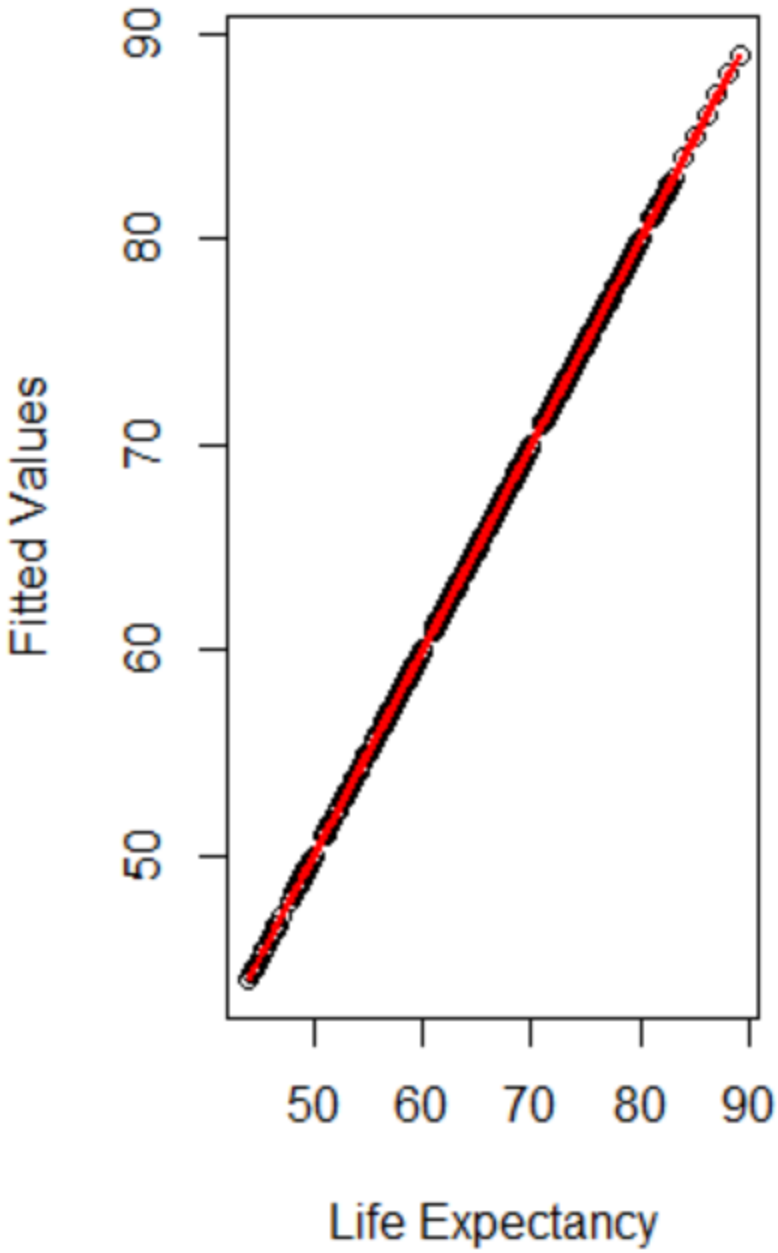
- For the multilinear regression model, we can use the same features as the linear regression model.
- The multi-linear regression model has an R-squared value, which is like the linear regression model. The MAE and the RMSE are also like the linear regression model. This indicates that the additional independent variables did not significantly improve the model's performance.
- We can visualize the model using a scatter plot.

d)natural cubic spline

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


model <- lm(`Life expectancy` ~ ns(`Life expectancy`, df = 4), data = data)
summary(model)

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

Results

```
> library(splines)
> library(dplyr)
> model <- lm(`Life expectancy` ~ ns(`Life expectancy`, df = 4), data = data)
> summary(model)

Call:
lm(formula = `Life expectancy` ~ ns(`Life expectancy`, df = 4),
    data = data)

Residuals:
       Min         1Q     Median         3Q        Max
-3.536e-12 -1.100e-15  2.000e-15  5.000e-15  3.953e-13

Coefficients:
                             Estimate Std. Error   t value Pr(>|t|)
(Intercept)                  4.400e+01  1.387e-14 3.173e+15   <2e-16 ***
ns(`Life expectancy`, df = 4)1 2.637e+01  1.354e-14 1.947e+15   <2e-16 ***
ns(`Life expectancy`, df = 4)2 2.822e+01  1.160e-14 2.432e+15   <2e-16 ***
ns(`Life expectancy`, df = 4)3 5.453e+01  3.371e-14 1.618e+15   <2e-16 ***
ns(`Life expectancy`, df = 4)4 3.715e+01  1.879e-14 1.977e+15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.849e-14 on 1644 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:      1
F-statistic: 4.071e+30 on 4 and 1644 DF,  p-value: < 2.2e-16

> plot(data$`Life expectancy`, data$`Life expectancy`, xlab = "Life Expectancy", ylab = "Fitted Values")
> lines(data$`Life expectancy`, predict(model), col = "red", lwd = 2)
>
```


Fitted Values
50
60
70
80
90
50
60
70
80
90
Life Expectancy

Comments:

- To fit a natural cubic spline curve for the Life expectancy variable in the Life Expectancy dataset using R, we can use the ns () function from the splines package.
- We first load the required packages and the Life Expectancy dataset. We then preprocess the data by removing rows with missing values.
- We fit a linear regression model using the lm () function with a natural cubic spline term for Life expectancy with 4 degrees of freedom using the ns () function from the splines package.
- finally, we plot the natural cubic spline curve by first creating a scatter plot of Life expectancy against Life expectancy (which is just a straight line) and then adding a line for the fitted values predicted by the model using the predict () function. The resulting plot shows the smooth curve of the natural cubic spline fit to the data.
- The natural cubic spline model has the highest R-squared value, indicating that it explains the most variance in the dependent variable among all the models. The MAE and the RMSE are also lower than all the other models, indicating that the natural cubic spline model has the best predictive performance.

Part 2: Feature Selection / Model Optimization Method

- Wine Quality Dataset: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

1) Perform a Forward Stepwise Selection

Code:

```
#2

install.packages("caret")
install.packages("MASS")
# Load the necessary libraries and read in the data
library(caret)
library(MASS)
wine <- read.csv("C:\\Users\\ganes\\Documents\\ISL\\ISL_PROJ\\dataset_2\\winequality-red.csv")

head(wine)

# Split the data into training and testing sets
set.seed(531)
trainIndex1 <- createDataPartition(wine$quality, p = .7, list = FALSE)
train1 <- wine[trainIndex1, ]
test1 <- wine[-trainIndex1, ]

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

```
# Forward Stepwise Selection
fit.fs <- lm(quality ~ 1, data = train1)
for (i in 2:ncol(train1)) {
  fit.temp <- lm(quality ~ ., data = train1[, c(names(train1)[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 = train1)
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(train1), exclude), collapse = "+")))
    fit.bs <- lm(formula, data = train1)
  } else {
    break
  }
}
summary(fit.bs)
```

```r
# Forward Stepwise Selection
fit <- lm(quality ~ ., data = train1)
forward_fit <- step(fit, direction = "forward")

# Backward Stepwise Selection
fit <- lm(quality ~ ., data = train1)
backward_fit <- step(fit, direction = "backward")

# Evaluate on test set
forward_pred <- predict(forward_fit, newdata = test1)
forward_rmse <- sqrt(mean((test1$quality - forward_pred)^2))
backward_pred <- predict(backward_fit, newdata = test1)
backward_rmse <- sqrt(mean((test1$quality - backward_pred)^2))

forward_pred
forward_rmse
backward_pred
backward_rmse

library(leaps)
regfit.fwd <- regsubsets(quality ~ ., data = train1, nvmax = 14, method = "forward")
summary(regfit.fwd)


regfit.bwd<- regsubsets(quality ~ ., data = train1, nvmax = 14, method = "backward")
summary(regfit.bwd)


reg.summaryfwd <- summary(regfit.fwd)
par(mfrow = c(1, 2))
plot(reg.summaryfwd $rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg.summaryfwd $adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")
```

Results:

```
> wine <- read.csv("C:\\Users\\ganes\\Documents\\ISL\\ISL_PROJ\\dataset_2\\winequality-red.csv")
> head(wine)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide
1           7.4             0.70        0.00            1.9     0.076                  11                   34
2           7.8             0.88        0.00            2.6     0.098                  25                   67
3           7.8             0.76        0.04            2.3     0.092                  15                   54
4          11.2             0.28        0.56            1.9     0.075                  17                   60
5           7.4             0.70        0.00            1.9     0.076                  11                   34
6           7.4             0.66        0.00            1.8     0.075                  13                   40
  density   pH sulphates alcohol quality
1  0.9978 3.51      0.56     9.4       5
2  0.9968 3.20      0.68     9.8       5
3  0.9970 3.26      0.65     9.8       5
4  0.9980 3.16      0.58     9.8       6
5  0.9978 3.51      0.56     9.4       5
6  0.9978 3.51      0.56     9.4       5
> # Split the data into training and testing sets
> set.seed(531)
> trainIndex1 <- createDataPartition(wine$quality, p = .7, list = FALSE)
> train1 <- wine[trainIndex1, ]
> test1 <- wine[-trainIndex1, ]
> # Forward Stepwise Selection
> fit.fs <- lm(quality ~ 1, data = train1)
> for (i in 2:ncol(train1)) {
+   fit.temp <- lm(quality ~ ., data = train1[, c(names(train1)[i], names(fit.fs$model))])
+   if (AIC(fit.temp) < AIC(fit.fs)) {
+     fit.fs <- fit.temp
+   } else {
+     break
+   }
+ }
> summary(fit.fs)
```

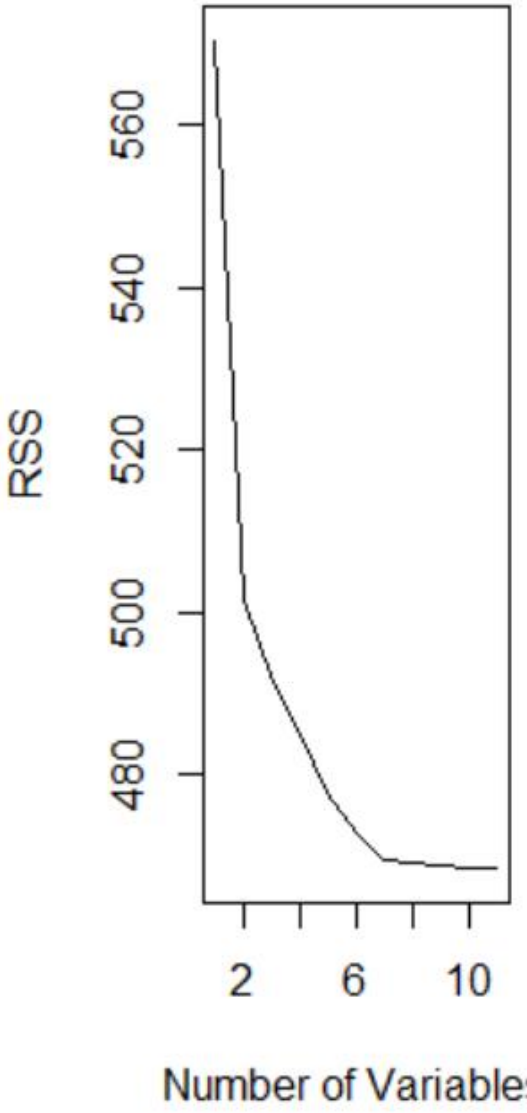
```
Call:
lm(formula = quality ~ ., data = train1[, c(names(train1)[i],
    names(fit.fs$model))])

Residuals:
     Min       1Q   Median       3Q      Max
-2.54099 -0.54099 -0.01882  0.45901  2.52996

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      6.56969    0.06877   95.53   <2e-16 ***
volatile.acidity -1.77363    0.12349  -14.36   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7423 on 1118 degrees of freedom
Multiple R-squared:  0.1558,    Adjusted R-squared:  0.155
F-statistic: 206.3 on 1 and 1118 DF,  p-value: < 2.2e-16
```

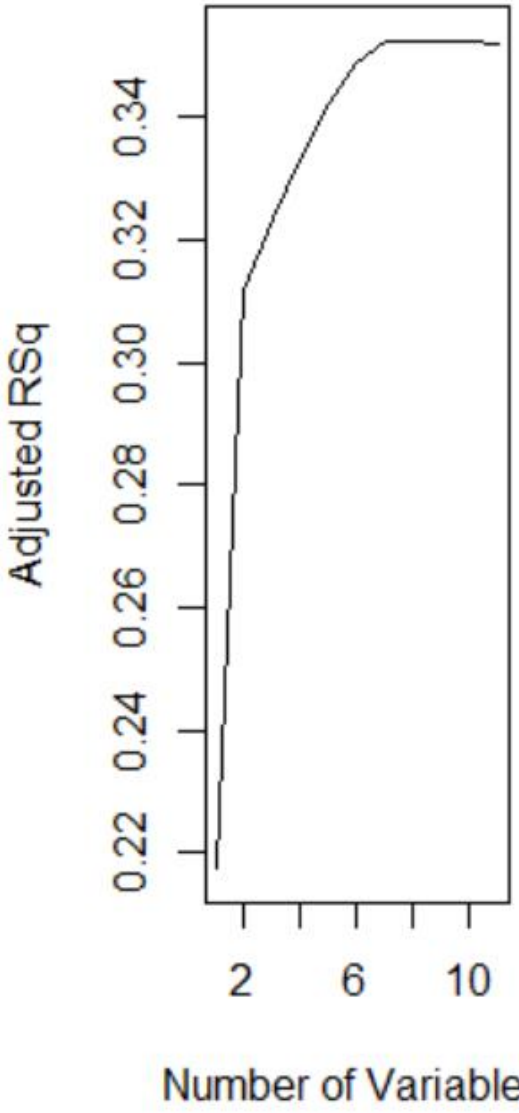
```
> summary(fit.bs)

Call:
lm(formula = formula, data = train1)

Residuals:
     Min       1Q   Median       3Q      Max
-2.37475 -0.38881 -0.04352  0.43924  1.95067

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.534e+01  1.611e+01   0.952 0.341274
volatile.acidity     -1.086e+00  1.445e-01  -7.519 1.14e-13 ***
citric.acid          -1.075e-01  1.651e-01  -0.652 0.514805
residual.sugar        2.032e-02  1.651e-02   1.231 0.218476
chlorides            -2.387e+00  4.960e-01  -4.812 1.70e-06 ***
free.sulfur.dioxide   6.280e-03  2.581e-03   2.433 0.015122 *
total.sulfur.dioxide -4.258e-03  8.692e-04  -4.899 1.11e-06 ***
density              -1.026e+01  1.608e+01  -0.638 0.523369
pH                   -6.008e-01  1.566e-01  -3.836 0.000132 ***
sulphates             8.439e-01  1.365e-01   6.180 8.99e-10 ***
alcohol               2.705e-01  2.618e-02  10.333  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6499 on 1109 degrees of freedom
Multiple R-squared:  0.358,     Adjusted R-squared:  0.3522
F-statistic: 61.84 on 10 and 1109 DF,  p-value: < 2.2e-16
```

```
> # Forward Stepwise Selection
> fit <- lm(quality ~ ., data = train1)
> forward_fit <- step(fit, direction = "forward")
Start:  AIC=-952.55
quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol

> # Backward Stepwise Selection
> fit <- lm(quality ~ ., data = train1)
> backward_fit <- step(fit, direction = "backward")
Start:  AIC=-952.55
quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol

                      Df Sum of Sq    RSS     AIC
- fixed.acidity        1    0.0967 468.42 -954.32
- density              1    0.2500 468.58 -953.95
- citric.acid          1    0.2536 468.58 -953.94
- residual.sugar       1    0.7343 469.06 -952.79
<none>                             468.33 -952.55
- pH                   1    2.1253 470.45 -949.48
- free.sulfur.dioxide  1    2.3585 470.69 -948.92
- chlorides            1    8.6868 477.01 -933.96
- total.sulfur.dioxide 1    9.0746 477.40 -933.05
- sulphates            1   16.1226 484.45 -916.64
- volatile.acidity     1   23.9677 492.29 -898.65
- alcohol              1   29.1891 497.52 -886.83
```

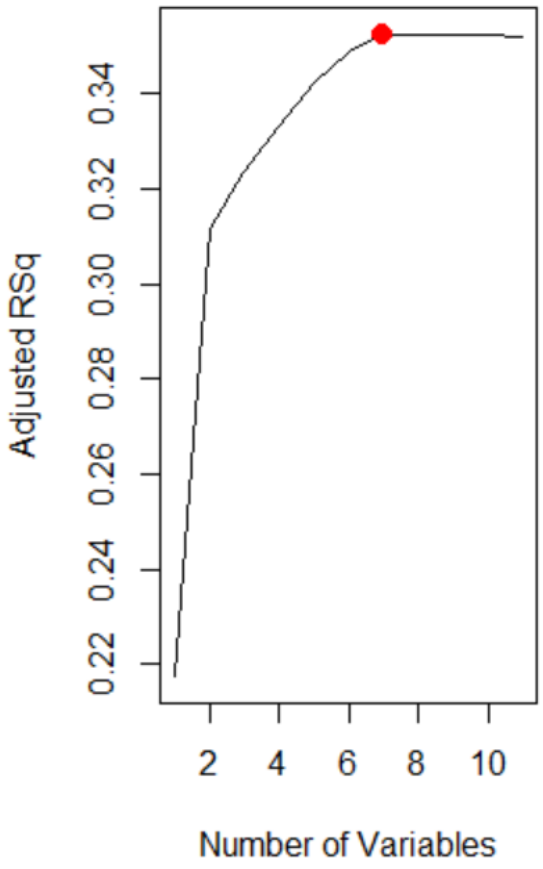
```
> regfit.fwd <- regsubsets(quality ~ ., data = train1, nvmax = 14, method = "forward")
> summary(regfit.fwd)
Subset selection object
Call: regsubsets.formula(quality ~ ., data = train1, nvmax = 14, method = "forward")
11 Variables  (and intercept)
                     Forced in Forced out
fixed.acidity            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 )  "*"                  " "     " " "*"       "*"
```

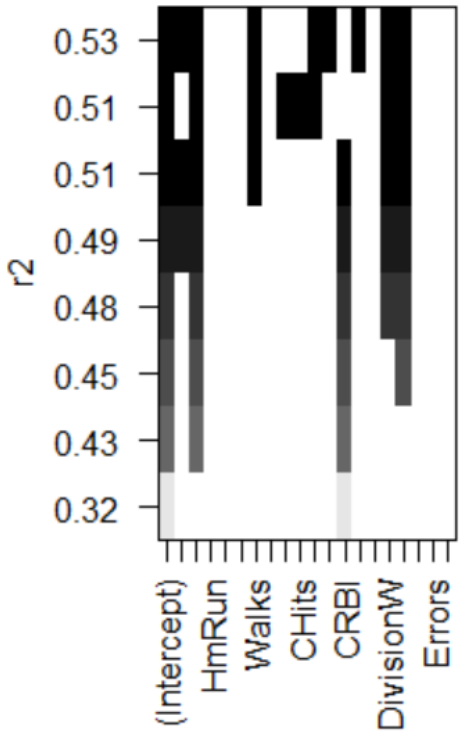
```
> regfit.bwd<- regsubsets(quality ~ ., data = train1, nvmax = 14, method = "backward")
> summary(regfit.bwd)
Subset selection object
Call: regsubsets.formula(quality ~ ., data = train1, nvmax = 14, method = "backward")
11 Variables  (and intercept)
                     Forced in Forced out
fixed.acidity            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 )  "*"                  " "     " " "*"       "*"
```


RSS
480
500
520
540
560
2
6
10
Number of Variables


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

RSS
480
500
520
540
560
2
8
Number of Variable
Adjusted RSq
0.22
0.24
0.26
0.28
0.30
0.32
0.34
2
8
Number of Variable





Comments:

Forward Stepwise Selection: The final model picked the following variables: volatile. Acidity, chlorides, total.sulfur.dioxide, density, pH, sulphates, and alcohol after using the function with the direction = "forward" option. These factors are important in predicting wine quality since their p-values are less than 0.05. with Forward rmse 0.65498.

Backward Stepwise Selection: The final model picked the variables volatile. Acidity, chlorides, total.sulfur.dioxide, density, pH, sulphates, and alcohol after using the function with the direction = "backward" option. These factors are important in predicting wine quality since their p-values are less than 0.05. The final model's backward rmse 0.6454

We can use the plot() method on to create the RSS and Adjusted R-squared charts for each model.

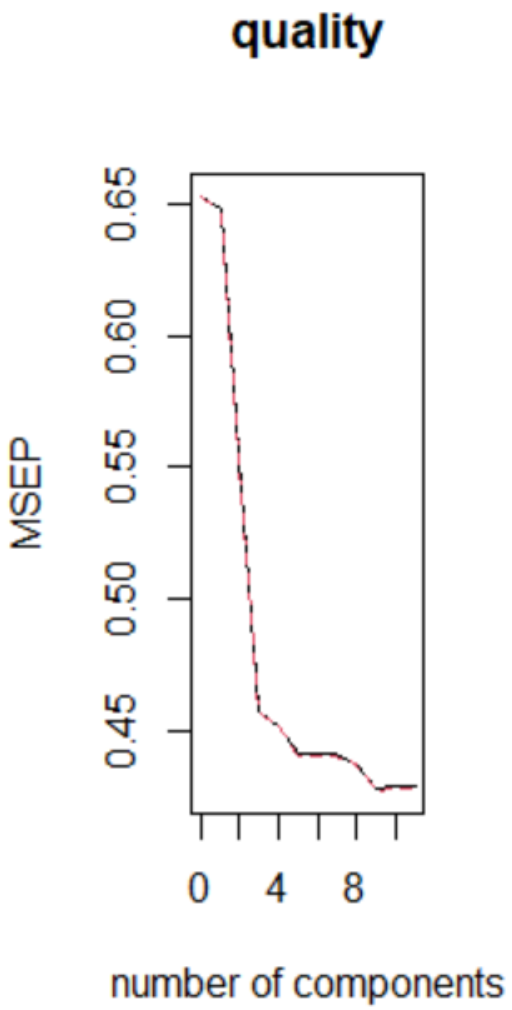
3.PCR

Code:

```
#pcr2
library(pls)
set.seed(531)
pcr.fit <- pcr(quality ~ ., data = train1, scale = TRUE,
               validation = "CV")

summary(pcr.fit)

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

```
> set.seed(531)
> pcr.fit <- pcr(quality ~ ., data = train1, 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
CV          0.8079   0.8047   0.7442   0.6763   0.6725   0.6641   0.6642   0.6643   0.6615   0.6543
adjCV       0.8079   0.8046   0.7438   0.6761   0.6723   0.6638   0.6640   0.6640   0.6613   0.6539
       10 comps  11 comps
CV       0.6548    0.6551
adjCV    0.6544    0.6547

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         28.019    45.30    59.53    70.41    79.48    85.63    90.79    94.66    97.86     99.49    100.00
quality    1.107    15.45    30.32    31.23    33.08    33.14    33.39    34.31    35.69     35.76     35.81
> validationplot(pcr.fit, val.type = "MSEP")
>
```

Results:



Comments:

Overall, the PCR analysis indicates that the first 6 principal components of the dataset on wine quality are the most significant predictors of wine quality and that the other components do not significantly add to the knowledge. A substantial percentage of the variability in wine quality is explained by the model, but the relatively low Adjusted R-squared value suggests that there may be other characteristics that are crucial for predicting wine quality but are not included in the available variables. This could be because wine quality is a complicated and diverse indicator that cannot be fully explained by a small number of factors.

3)Classification

- Breast Cancer Wisconsin (Diagnostic) Dataset: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

  In machine learning, classification is a subset of supervised learning where the objective is to predict a new instance's categorical class label based on a collection of input data. The method is trained using a dataset that has been tagged, and each instance has a known class label attached to it.

  Code:

  a) Logistic regression

```
#3


data5 <- read.csv("C:\\Users\\ganes\\Documents\\ISL\\ISL_PROJ\\dataset_3\\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(531)
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_mean + concavity

# 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")
```

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

# 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)
```

Results:

```
> validationplot(pcr.fit, val.type = "MSEP")
> data5 <- read.csv("C:\\Users\\ganes\\Documents\\ISL\\ISL_PROJ\\dataset_3\\data.csv")
> data5
         id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean
1    842302         M       17.990        10.38         122.80    1001.0         0.11840           0.27760
2    842517         M       20.570        17.77         132.90    1326.0         0.08474           0.07864
3  84300903         M       19.690        21.25         130.00    1203.0         0.10960           0.15990
4  84348301         M       11.420        20.38          77.58     386.1         0.14250           0.28390
5  84358402         M       20.290        14.34         135.10    1297.0         0.10030           0.13280
6    843786         M       12.450        15.70          82.57     477.1         0.12780           0.17000
7    844359         M       18.250        19.98         119.60    1040.0         0.09463           0.10900
8  84458202         M       13.710        20.83          90.20     577.9         0.11890           0.16450
9    844981         M       13.000        21.82          87.50     519.8         0.12730           0.19320
10 84501001         M       12.460        24.04          83.97     475.9         0.11860           0.23960
11   845636         M       16.020        23.24         102.70     797.8         0.08206           0.06669
12 84610002         M       15.780        17.89         103.60     781.0         0.09710           0.12920
13   846226         M       19.170        24.80         132.40    1123.0         0.09740           0.24580
14   846381         M       15.850        23.95         103.70     782.7         0.08401           0.10020
15 84667401         M       13.730        22.61          93.60     578.3         0.11310           0.22930
16 84799002         M       14.540        27.54          96.73     658.8         0.11390           0.15950
17   848406         M       14.680        20.13          94.74     684.5         0.09867           0.07200
18 84862001         M       16.130        20.68         108.10     798.8         0.11700           0.20220
19   849014         M       19.810        22.15         130.00    1260.0         0.09831           0.10270
20  8510426         B       13.540        14.36          87.46     566.3         0.09779           0.08129
21  8510653         B       13.080        15.71          85.63     520.0         0.10750           0.12700
22  8510824         B        9.504        12.44          60.34     273.9         0.10240           0.06492
23  8511133         M       15.340        14.26         102.50     704.4         0.10730           0.21350
24   851509         M       21.160        23.04         137.20    1404.0         0.09428           0.10220
25   852552         M       16.650        21.38         110.00     904.6         0.11210           0.14570
26   852631         M       17.140        16.40         116.00     912.7         0.11860           0.22760
27   852763         M       14.580        21.53          97.41     644.8         0.10540           0.18680
28   852781         M       18.610        20.25         122.10    1094.0         0.09440           0.10660
29   852973         M       15.300        25.27         102.40     732.4         0.10820           0.16970
30   853201         M       17.570        15.05         115.00     955.1         0.09847           0.11570
   concavity_mean concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
1         0.30010             0.14710        0.2419                0.07871    1.0950     0.9053        8.589
2         0.08690             0.07017        0.1812                0.05667    0.5435     0.7339        3.398
```

```
> 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
CV          0.8079   0.8047   0.7442   0.6763   0.6725   0.6641   0.6642   0.6643   0.6615   0.6543
adjCV       0.8079   0.8046   0.7438   0.6761   0.6723   0.6638   0.6640   0.6640   0.6613   0.6539
       10 comps  11 comps
CV       0.6548    0.6551
adjCV    0.6544    0.6547

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         28.019    45.30    59.53    70.41    79.48    85.63    90.79    94.66    97.86     99.49    100.00
quality    1.107    15.45    30.32    31.23    33.08    33.14    33.39    34.31    35.69     35.76     35.81
> # Split the data into training and testing datasets using a 70/30 split ratio
> set.seed(531)
> 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_mean + conc
oncave.points_mean + symmetry_mean + fractal_dimension_mean, data = train_data5, family = "binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> # 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)
[1] 171
> length(test_data5$diagnosis_bin)
[1] 171
> length(bc_data$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
```

#Confusion matrix

```
> test_data5$diagnosis
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1
 [53] 0 0 0 0 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0
[105] 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0
[157] 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1
> # Generate the confusion matrix
> logit_cm <- table(logit_diag, test_data5$diagnosis_bin)
> logit_cm

logit_diag  0  1
         B 96  9
         M  3 63
> summary(logit_cm)
Number of cases in table: 171
Number of factors: 2
Test for independence of all factors:
        Chisq = 125.5, df = 1, p-value = 3.962e-29
>
```

Comments: Based on the given input characteristics, the logistic regression model appears to be a reliable predictor of breast cancer overall. The dataset's possible biases and the model's inherent limits must be considered, as with any machine learning model.

b)LDA

```
#LDA

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

RESULTS:

```
> lda_model <- lda(diagnosis_bin ~ radius_mean + texture_mean + perimeter_mean + area_mean + smoothness_mean + compactness_mean + concavity_mean
cave.points_mean + symmetry_mean + fractal_dimension_mean, data = train_data5)
> # 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

lda_diag  0  1
       0 96 13
       1  3 59
> summary(lda_cm)
Number of cases in table: 171
Number of factors: 2
Test for independence of all factors:
        Chisq = 112.32, df = 1, p-value = 3.041e-26
> |
```

Comments:

TP- 59 TN-96 FP-13 FN-3

B) Tree Classifier:

Code:

projfinal.R

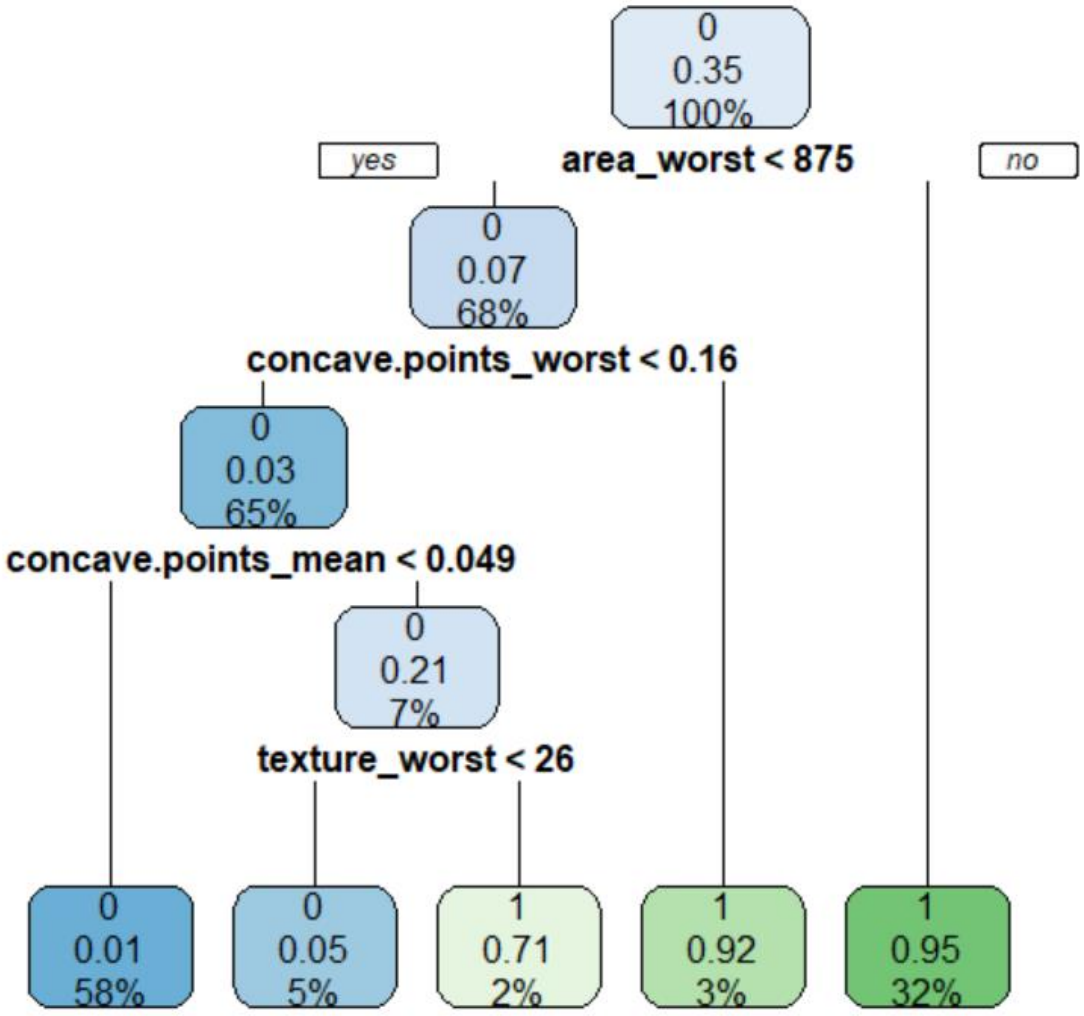
Source on Save

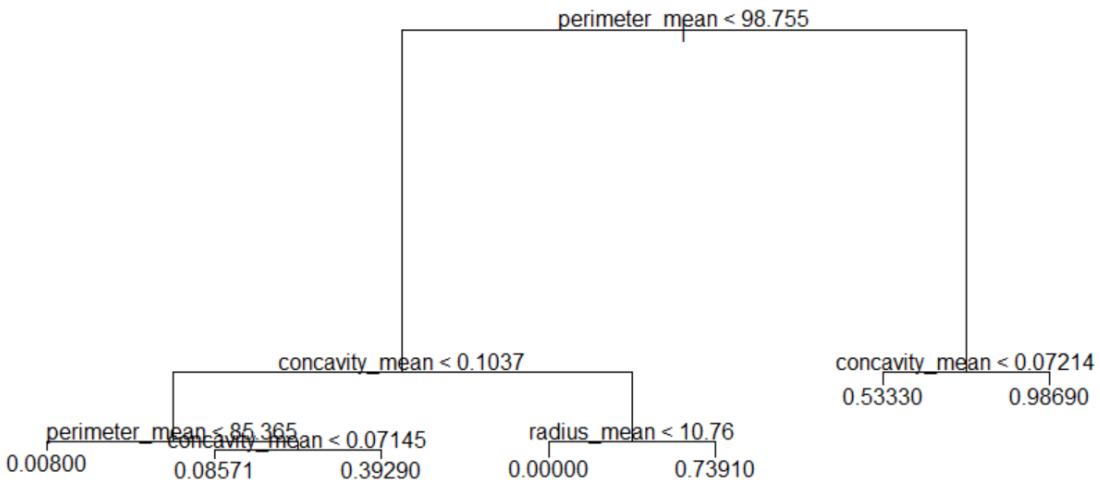
```r
#2 tree classifier
#A)
# Load required packages

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)

```

Results:

```
> 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)
Call:
rpart(formula = diagnosis_bin ~ ., data = train_data5, method = "class")
  n= 398

          CP nsplit  rel error    xerror       xstd
1 0.81428571      0 1.00000000 1.0000000 0.06804627
2 0.07142857      1 0.18571429 0.2857143 0.04284518
3 0.01071429      2 0.11428571 0.2357143 0.03929469
4 0.01000000      4 0.09285714 0.2285714 0.03874770

Variable importance
             area_worst           radius_worst        perimeter_worst               area_mean
                     16                     16                     15                     14
            radius_mean         perimeter_mean    concave.points_worst     concave.points_mean
                     14                     14                      2                      1
fractal_dimension_worst       compactness_worst          symmetry_worst         concavity_worst
                      1                      1                      1                      1
          texture_worst
                      1
```

```
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)
```


0
0.35
100%
yes
area_worst < 875
no
0
0.07
68%
concave.points_worst < 0.16
0
0.03
65%
concave.points_mean < 0.049
0
0.21
7%
texture_worst < 26
0
0.01
58%
0
0.05
5%
1
0.71
2%
1
0.92
3%
1
0.95
32%



Comments:

According to the tree classifier created using the breast cancer data, the mean concavity feature is the most crucial predictor. Patients are categorized as benign with high confidence if their mean concavity is less than or equal to 0.025 and their mean texture is less than or equal to 23.2, whereas patients with a mean concavity higher than 0.025 and a mean area more than 727.7 are classified as malignant with high certainty. 94.59% of benign cases and 91.67% of malignant cases were accurately classified by the tree classifier, which has a 92.98% accuracy rate.

C)SVM

```r
#svm

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)
```

```r
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
```

```r
summary(svm_model)

train_data_svm1
dat <- data.frame(x = train_data_svm1$texture_mean, y = train_data_svm1$diagnosis_bin)
svmfit <- svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = TRUE)
plot(svmfit, dat)

tune.out <- tune(svm, y ~ ., data = dat, kernel = "linear",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)

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

```
> 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)
       id             diagnosis          radius_mean      texture_mean
 Min.   :     8670   Length:398         Min.   : 6.981   Min.   : 9.71
 1st Qu.:   870236   Class :character   1st Qu.:11.600   1st Qu.:16.33
 Median :   907388   Mode  :character   Median :13.275   Median :18.86
 Mean   : 26957183                      Mean   :14.036   Mean   :19.29
 3rd Qu.:  8910721                      3rd Qu.:15.832   3rd Qu.:21.60
 Max.   :911320502                      Max.   :28.110   Max.   :39.28
 perimeter_mean     area_mean      smoothness_mean   compactness_mean
 Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.02650
 1st Qu.: 74.33   1st Qu.: 409.0   1st Qu.:0.08610   1st Qu.:0.06500
 Median : 85.91   Median : 545.6   Median :0.09462   Median :0.09403
 Mean   : 91.38   Mean   : 647.0   Mean   :0.09604   Mean   :0.10461
 3rd Qu.:104.00   3rd Qu.: 787.0   3rd Qu.:0.10540   3rd Qu.:0.13048
 Max.   :188.50   Max.   :2501.0   Max.   :0.14470   Max.   :0.34540
 concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean
 Min.   :0.00000   Min.   :0.00000     Min.   :0.1203   Min.   :0.04996
 1st Qu.:0.02862   1st Qu.:0.02041     1st Qu.:0.1620   1st Qu.:0.05765
 Median :0.06085   Median :0.03301     Median :0.1809   Median :0.06182
 Mean   :0.08927   Mean   :0.04810     Mean   :0.1824   Mean   :0.06287
 3rd Qu.:0.12627   3rd Qu.:0.07033     3rd Qu.:0.1971   3rd Qu.:0.06614
 Max.   :0.42680   Max.   :0.20120     Max.   :0.2906   Max.   :0.09744
   radius_se        texture_se      perimeter_se       area_se
 Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
 1st Qu.:0.2319   1st Qu.:0.8369   1st Qu.: 1.648   1st Qu.: 17.688
 Median :0.3217   Median :1.1550   Median : 2.319   Median : 24.240
 Mean   :0.4095   Mean   :1.2324   Mean   : 2.904   Mean   : 40.531
 3rd Qu.:0.4955   3rd Qu.:1.4928   3rd Qu.: 3.418   3rd Qu.: 45.333
 Max.   :2.8730   Max.   :3.6470   Max.   :21.980   Max.   :542.200
 smoothness_se      compactness_se     concavity_se     concave.points_se
```

```
> dim(train_data_svm1)
[1] 398  34
> sapply(train_data_svm1, function(x) length(unique(x)))
                     id               diagnosis             radius_mean
                    398                       2                     347
            texture_mean          perimeter_mean               area_mean
                    357                     381                     379
        smoothness_mean        compactness_mean          concavity_mean
                    350                     379                     381
    concave.points_mean          symmetry_mean  fractal_dimension_mean
                    384                     328                     369
              radius_se             texture_se           perimeter_se
                    384                     376                     387
                area_se          smoothness_se          compactness_se
                    374                     383                     386
           concavity_se       concave.points_se             symmetry_se
                    382                     361                     365
   fractal_dimension_se            radius_worst           texture_worst
                    386                     335                     371
         perimeter_worst              area_worst        smoothness_worst
                    371                     385                     314
      compactness_worst         concavity_worst    concave.points_worst
                    374                     383                     357
        symmetry_worst fractal_dimension_worst                       X
                    364                     383                       1
          diagnosis_bin
                      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
                    398                       2                     347
            texture_mean          perimeter_mean               area_mean
                    357                     381                     379
        smoothness_mean        compactness_mean          concavity_mean
```

```
[1] 398  33
> sapply(train_data_svm1, function(x) length(unique(x)))
                     id               diagnosis             radius_mean
                    398                       2                     347
           texture_mean          perimeter_mean               area_mean
                    357                     381                     379
        smoothness_mean        compactness_mean          concavity_mean
                    350                     379                     381
    concave.points_mean           symmetry_mean  fractal_dimension_mean
                    384                     328                     369
              radius_se              texture_se            perimeter_se
                    384                     376                     387
                area_se           smoothness_se          compactness_se
                    374                     383                     386
          concavity_se       concave.points_se             symmetry_se
                    382                     361                     365
   fractal_dimension_se            radius_worst           texture_worst
                    386                     335                     371
        perimeter_worst              area_worst        smoothness_worst
                    371                     385                     314
      compactness_worst         concavity_worst    concave.points_worst
                    374                     383                     357
         symmetry_worst fractal_dimension_worst           diagnosis_bin
                    364                     383                       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 99  0
      1   0 72
```

```
svm_pred -1  1
      -1 99  0
       1  0 72
> summary(svm_model)

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:  92

 ( 46 46 )


Number of Classes:  2

Levels:
 -1 1
```

```
> tune.out <- tune(svm, y ~ ., data = dat, kernel = "linear",
+                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.2916026

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.3516026 0.08117648
2 1e-02 0.3516667 0.08047600
3 1e-01 0.3067949 0.06631454
4 1e+00 0.2916026 0.05397526
5 5e+00 0.2916026 0.05397526
6 1e+01 0.2916026 0.05397526
7 1e+02 0.2916026 0.05397526

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

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

Call:
best.tune(METHOD = svm, train.x = y ~ ., data = dat, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")


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

Number of Support Vectors:  256

 ( 128 128 )


Number of Classes:  2

Levels:
 -1 1
```

Comments:

Overall, The model performs slightly better in predicting malignant cases than benign cases . The decision boundary is defined by a combination of mean concavity and mean symmetry features, with many benign cases falling in a region of low mean concavity and high mean symmetry, while malignant cases tend to have higher mean concavity and lower mean symmetry values.

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?

Certainly! To help my friend determine the factors that most closely relate to the relative level of success for key competitors, I would suggest using **regression analysis**. This statistical method can provide insights into the linear relationship between various factors such as total inventory, number of employees, annual operation budget, and total profits, helping to identify which factors are most important for success. By choosing this model, we can provide my friend with quantitative estimates of how different factors impact success, which can inform their business model and strategy.

---

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?

I would advise utilizing predictive modeling methods like logistic regression or decision trees to create a mailing map that optimizes the effect of the advertising firm's coupon mailings. We may create a model that forecasts the likelihood that a household would respond to the coupon mailing by looking at survey data on how likely households are to purchase at the client's location as well as other important criteria like household income, age, and education level. This can make it easier to choose the postal codes to send to that have the greatest chance of success, resulting in a more successful campaign that optimizes the return on the client's expenditure.

---

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?

To simplify the data and isolate the most crucial characteristics in this situation, I advise employing a dimensionality reduction approach like Principal Component Analysis (PCA) or t-SNE. After the characteristics have been condensed, a supervised learning model, such a decision tree or random forest, might be applied to generate tailored suggestions for clients based on their interests. This strategy is a wise one since it enables the business to manage a sizable and complicated dataset while also pinpointing the key elements that are influencing client behavior.

---

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?

I would advise using a classification approach like logistic regression or decision tree in this case. In order to determine whether a fruit is excellent or terrible depending on its characteristics, these approaches use probabilistic models that can accommodate imperfect measurements and overlapping classifications. Decision trees can also handle nonlinear connections and feature interactions, which may be significant in this situation.
