Predictive Analytics for Health Data

Regression Models for Quantitaive Health Data

A multiple linear regression model for quantitative health response (Y) models the linear dependence of Y on a set of explanatory variables $X_1, X_2, \ldots, X_p$. Similar to other regression models for categorical, count, and time-to-event responses, linear regression is what is called a supervised learning method.

A multiple linear regression model for the response Y on a set of p predictors (explanatory variables) is written as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_1 X_2 + \ldots + \beta_1 X_p + \epsilon$$

Here, β_0 is the intercept and $\beta_j, j = 1, \ldots, p$ is the average effect of a one unit increase in X_j on Y holding all other predictors fixed.

```
library(NHANES)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

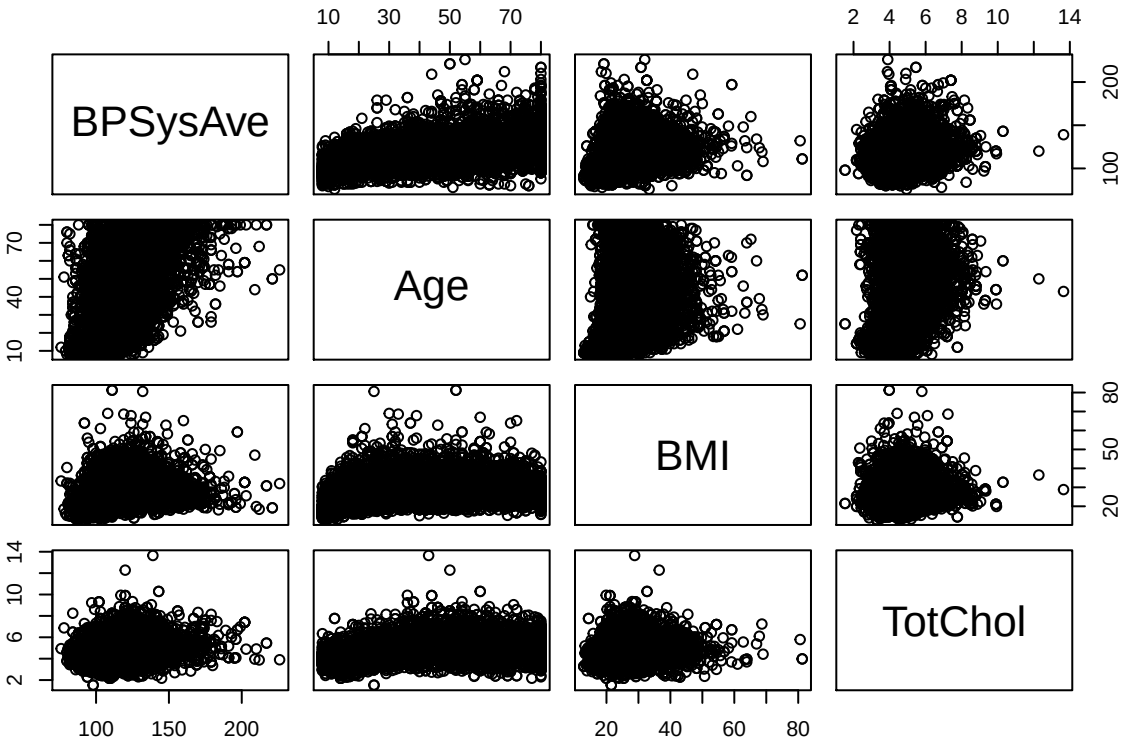
```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- select(NHANES, ID, SurveyYr, Gender, Age, Race1, Poverty, HomeOwn, Weight, Height, BMI, BPSysAve,

df <- filter(df, BPSysAve !="NA")
```

Example 1: Modeling linear relationship between average systolic BP and Age, BMI and Total Cholesterol of NHANES participants

```
# scatterplots
pairs(BPSysAve~ Age + BMI + TotChol, data=df)
```



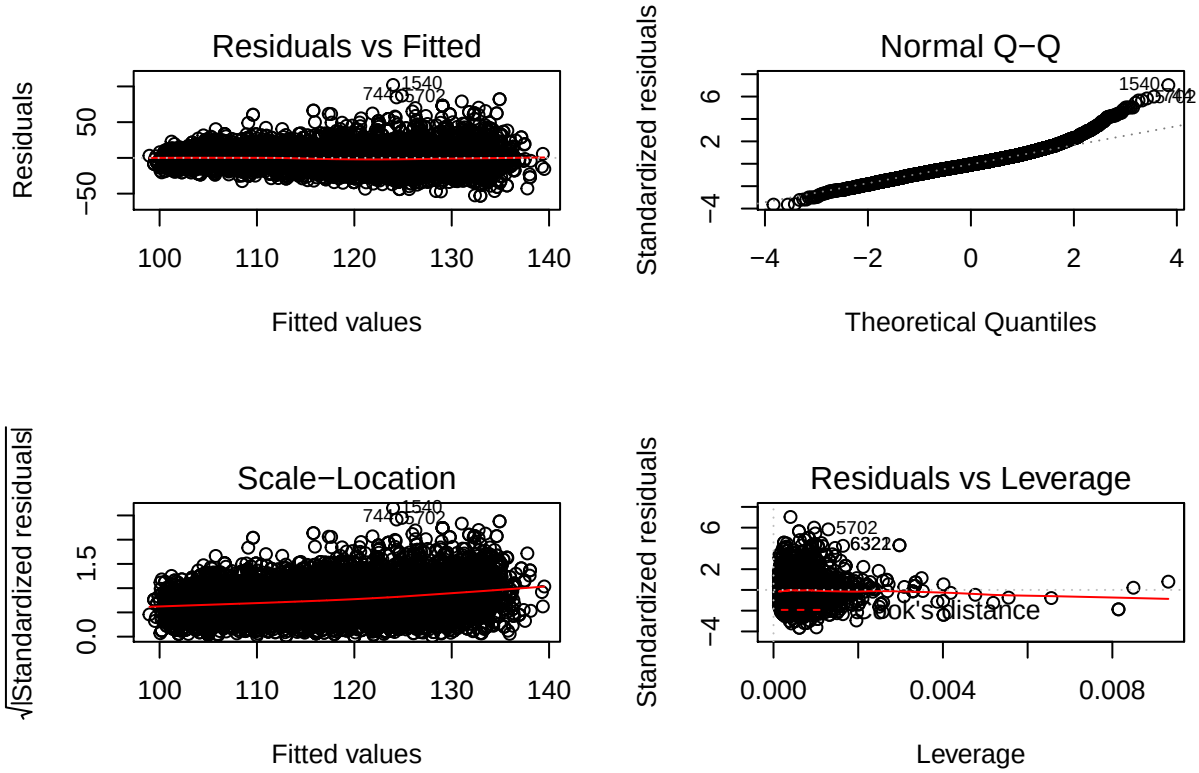

```
# fitting a multiple regression model

fit=lm(BPSysAve~ Age + BMI + TotChol, data=df)
summary(fit)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Age + BMI + TotChol, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.947  -8.877  -0.897   7.759 102.022
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 88.817237   0.948453  93.644  < 2e-16 ***
## Age          0.393064   0.008855  44.389  < 2e-16 ***
## BMI          0.307908   0.024275  12.684  < 2e-16 ***
## TotChol      0.947411   0.159786   5.929 3.17e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.52 on 7932 degrees of freedom
##   (615 observations deleted due to missingness)
## Multiple R-squared:  0.2758, Adjusted R-squared:  0.2755
## F-statistic:  1007 on 3 and 7932 DF,  p-value: < 2.2e-16
```

Note all three predictors are important in explaining variabilities in average systolic blood pressure. There is an increase of about 0.95 in the average systolic BP for one unit increase in total cholesterol keeping age and bmi at fixed values.

Model assumption checking/model diagnostics

```
par(mfrow=c(2,2))
plot(fit)
```



The **Residual versus Fitted plot** shows if residuals have non-linear patterns. There could be a non-linear relationship between predictor variables and an outcome variable and the pattern could be visible in this plot if the model doesn't capture the non-linear relationship. Equally spread residuals around a horizontal line without distinct patterns is a good indication that there is no non-linear relationships. In this example the residual versus fitted plot shows no such pattern.

The **Normal Q-Q plot** shows if residuals are normally distributed. If the residuals follow a straight line well and do not deviate severely, their distribution is approximately normal. In this example there is some deviation on the top right corner.

The **Scale - Location plot** shows if residuals are spread equally along the ranges of predictors. You can check the assumption of equal variance (homoscedasticity) using this plot. If there is a horizontal line with equally (randomly) spread points which is the case in this example, homoscedasticity assumption is satisfied.

The **Residuals vs Leverage plot** helps us to find influential observations, if any. Not all outliers are influential in linear regression analysis. Even though data have extreme values, they might not be influential to determine a regression line. That is, the results wouldn't be much different if we either include or exclude them from analysis. On the other hand, some observations could be very influential even if they look to be within a reasonable range of the values. They could be extreme observations against a regression line and can alter the results if we exclude them from analysis.

When observations are outside of the Cook's distance (have high Cook's distance scores), the observations are influential to the regression results. The regression results will be altered if we exclude those cases.

Predict sales for new values of the feature set

```
newdata=data.frame(Age= 40, BMI=27, TotChol = 4.5)
predict(fit, newdata, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 117.1166 88.64747 145.5858
```

Model Evaluation with Test Error

Predictive performance of a learning method should be evaluated based on unforseen or test data. Most of the time there will not be test data for us to evaluate our model. One apporach is to leave some data out as validation data set. Fit (train) your model on rest of the data (training data) and test (evaluate on the left out validation/ test data).

```
set.seed(1)
df1 <- complete.cases(select(df, Age, BMI, BPSysAve, TotChol))
df2 = df[df1,]
train <- sample(7936,5000) # Randomly sample 5000 items from 7936 items
train.dat <- df2[train,]

test.dat <- df2[-train,]

lm.tr = lm(BPSysAve ~ Age + BMI + TotChol, data=train.dat) # fit regression model on training data

mean((test.dat$BPSysAve - predict(lm.tr,test.dat))^2)    # Test error
```

```
## [1] 206.4407
```

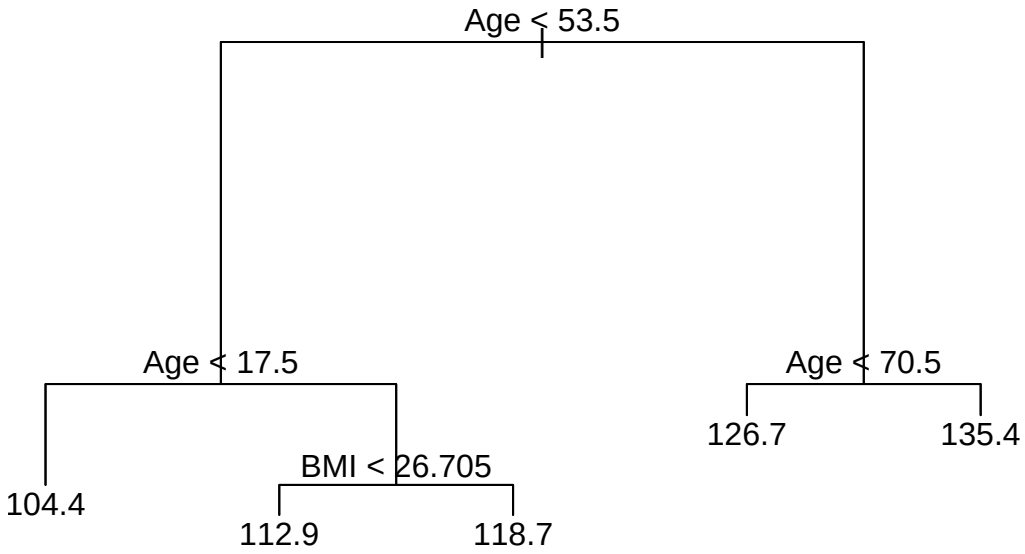
Regression tree

Decision trees are supervised learning methods that can be applied to both regression and classification problems. Decision tree-based methods partition the feature space in a hierarchical way. Breiman et al. (1984) proposed the classification and regression tree methods.

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
tree.sbp =tree(BPSysAve ~Age + BMI + TotChol , data=train.dat)

plot(tree.sbp )
text(tree.sbp, pretty =0)
```




```
yhat=predict(tree.sbp, newdata = test.dat)
mean((yhat-test.dat$BPSysAve)^2)
```

```
## [1] 211.1475
```

Random Forest for Regression problem

Introduced by Leo Breiman (2001), Random Forests are a combination of tree based predictive methods, where each tree is constructed based on a random sample of features to split each node.

```
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf.sbp = randomForest(BPSysAve ~Age + BMI + TotChol , data=train.dat)

rf.sbp
```

```
##
## Call:
##  randomForest(formula = BPSysAve ~ Age + BMI + TotChol, data = train.dat)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
```

```
##
##           Mean of squared residuals: 147.1996
##                     % Var explained: 49.84
```

```
yhat.rf = predict(rf.sbp, newdata=test.dat)
mean((yhat.rf -test.dat$BPSysAve)^2)
```

```
## [1] 148.1663
```

Note that the Random Forest does a pretty good job in terms reduction of test prediction error.