R Notebook

Mert Göksel

```
suppressPackageStartupMessages(library(openxlsx))
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(lmtest))
suppressPackageStartupMessages(library(car))
suppressPackageStartupMessages(library(corrplot))
suppressPackageStartupMessages(library(leaps))
suppressPackageStartupMessages(library(stats))
suppressPackageStartupMessages(library(olsrr))
suppressPackageStartupMessages(library(ResourceSelection))
suppressPackageStartupMessages(library(pROC))
options(scipen=7)
```

Question 1

```
titanic <- openxlsx::read.xlsx("./titanic.xlsx")
head(titanic)
```

```
##   Pclass    Sex Age SibSp Parch    Fare Embarked Survived
## 1      3   male  22     1     0  7.2500        S        0
## 2      1 female  38     1     0 71.2833        C        1
## 3      3 female  26     0     0  7.9250        S        1
## 4      1 female  35     1     0 53.1000        S        1
## 5      3   male  35     0     0  8.0500        S        0
## 6      3   male  NA     0     0  8.4583        Q        0
```

```
titanic <- drop_na(titanic)

titanic_train_indexes <- sample(1:nrow(titanic),
                        floor(nrow(titanic)*8/10)) #80% is train
titanic_train <- titanic[titanic_train_indexes,]
titanic_test <- titanic[-titanic_train_indexes,]
```

Part A

```
model.titanic <- glm(Survived~., data=titanic_train, family="binomial")

summary(model.titanic)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = "binomial", data = titanic_train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5730  -0.7275  -0.4155   0.6938   2.3898
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.3423898  0.6962624   7.673 1.68e-14 ***
## Pclass      -1.1721300  0.1777818  -6.593 4.31e-11 ***
## Sexmale     -2.4179588  0.2381316 -10.154  < 2e-16 ***
## Age         -0.0390111  0.0089623  -4.353 1.34e-05 ***
## SibSp       -0.3063472  0.1454894  -2.106   0.0352 *
## Parch       -0.0417291  0.1342052  -0.311   0.7558
## Fare         0.0001341  0.0028658   0.047   0.9627
## EmbarkedQ   -0.8748999  0.6786025  -1.289   0.1973
## EmbarkedS   -0.4498436  0.2954327  -1.523   0.1278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.79  on 568  degrees of freedom
## Residual deviance: 534.92  on 560  degrees of freedom
## AIC: 552.92
##
## Number of Fisher Scoring iterations: 4
```

From this model, our response function is;

```
cat("logodds =\n")
```

```
## logodds =
```

```
cat(paste(row.names(summary(model.titanic)$coefficients),
          summary(model.titanic)$coefficients[,1], sep="*"),sep="+\n")
```

```
## (Intercept)*5.34238976129524+
## Pclass*-1.17213002946299+
## Sexmale*-2.4179588065223+
## Age*-0.0390111261327486+
## SibSp*-0.306347168592785+
## Parch*-0.0417291401773174+
## Fare*0.000134054071432965+
## EmbarkedQ*-0.874899870948799+
## EmbarkedS*-0.449843581731476
```

Part B

We have our parameters above, these are for the logodds. Meaning if above function is BX, then;

$$
\begin{aligned}
& logodds = BX \\
& \Rightarrow \; ln(\frac{\pi(x)}{1-\pi(x)}) = BX \\
& \Rightarrow \; \pi(x) = \frac{e^{BX}}{1+e^{BX}} \\
& \Rightarrow \; \pi(x) = \frac{1}{1+e^{-BX}}
\end{aligned}
$$

From this formula we can interpret these coefficient as following: - When BX = logodds, unit of increase in any variable will increase/decrease the logodds by the respective coefficient. Meaning, the odds will be affected as much as e^{B_k} - On the other side, if we are looking at probability formula the change in the probability will be;

$$
\begin{aligned}
When \; x_i &\Rightarrow x_i + 1 \\
B_i * x_i &\Rightarrow B_i * x_i + B_i \\
\pi(x) &\Rightarrow \frac{1}{1+e^{-(B_i x_i + B_i)}}
\end{aligned}
$$

These apply for each parameter.

Part C

```
given <- data.frame(Pclass=3, Sex="female", Age=35, SibSp=0, Parch=0, Fare=75, Embarked="C")
ifelse((predict(model.titanic, given, type="response") >= 0.5)[[1]], 1, 0)
```

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

Part D

We will apply wald test to all parameters with size $\alpha = 0.01$

$$
Reject \; H_0 \Rightarrow |z^*| > z_{\frac{1-\alpha}{2}}
$$

```
chi.crit <- qnorm(0.01/2, lower.tail = 0)

data.frame(z=summary(model.titanic)$coefficients[,3]) %>%
  mutate(z_star = abs(z), reject = ifelse(z_star > chi.crit, 1, 0))
```

```
##                        z      z_star reject
## (Intercept)   7.67295463  7.67295463      1
## Pclass       -6.59308072  6.59308072      1
## Sexmale     -10.15387497 10.15387497      1
## Age          -4.35280924  4.35280924      1
## SibSp        -2.10563190  2.10563190      0
## Parch        -0.31093535  0.31093535      0
## Fare          0.04677718  0.04677718      0
## EmbarkedQ    -1.28926715  1.28926715      0
## EmbarkedS    -1.52265998  1.52265998      0
```

Wald test tells us that **Parch, Fare and Embarked** are insignificant variables and thus needs to be dropped

Part E

To test the overall significance of the model we have to use **Likelihood** test

```
summary(model.titanic)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = "binomial", data = titanic_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5730  -0.7275  -0.4155   0.6938   2.3898
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.3423898  0.6962624   7.673 1.68e-14 ***
## Pclass      -1.1721300  0.1777818  -6.593 4.31e-11 ***
## Sexmale     -2.4179588  0.2381316 -10.154  < 2e-16 ***
## Age         -0.0390111  0.0089623  -4.353 1.34e-05 ***
## SibSp       -0.3063472  0.1454894  -2.106   0.0352 *
## Parch       -0.0417291  0.1342052  -0.311   0.7558
## Fare         0.0001341  0.0028658   0.047   0.9627
## EmbarkedQ   -0.8748999  0.6786025  -1.289   0.1973
## EmbarkedS   -0.4498436  0.2954327  -1.523   0.1278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.79  on 568  degrees of freedom
## Residual deviance: 534.92  on 560  degrees of freedom
## AIC: 552.92
##
## Number of Fisher Scoring iterations: 4
```

We know that likelihood of the model will be used as following to test the significance;

$$LR \sim \chi^2_{0.95,8}$$

$$LR = 2ln(\frac{L(FM)}{L(RM)})$$

```
model.titanic.reduced <- glm(Survived~1, data=titanic_train, family = 'binomial')

summary(model.titanic.reduced)
```

```
##
## Call:
## glm(formula = Survived ~ 1, family = "binomial", data = titanic_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.018  -1.018  -1.018   1.346   1.346
##
## Coefficients:
```

```
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -0.38792    0.08543  -4.541 0.0000056 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.79  on 568  degrees of freedom
## Residual deviance: 767.79  on 568  degrees of freedom
## AIC: 769.79
##
## Number of Fisher Scoring iterations: 4
```

```
lr <- 2*(logLik(model.titanic)[1] - logLik(model.titanic.reduced)[1])

lr
```

```
## [1] 232.8664
```

```
pchisq(454.837105802761, df = 8, lower.tail = 0)
```

```
## [1] 3.399737e-93
```

well, since this test score is basically 0 and thus < 0.05 we will reject H_0 and say that this model is significant.

Part F

In order to get confidence intervals for the parameters we will use this formula

$$\hat{\beta}_k \pm z_{\frac{1-\alpha}{2}}.se(\hat{\beta}_k)$$

```
conf <- as.data.frame(summary(model.titanic)$coefficients) %>%
  mutate(std.error = `Std. Error`, z.val = `z value`, `z value` = NULL,
         `Std. Error` = NULL) %>%
  select(Estimate, std.error, z.val) %>%
  mutate(conf.int_lower = Estimate-qnorm(0.05/2, lower.tail = 0)*std.error,
         conf.int_higher = Estimate+qnorm(0.05/2, lower.tail = 0)*std.error)

conf
```

```
##                  Estimate   std.error        z.val conf.int_lower
## (Intercept)  5.3423897613 0.696262394   7.67295463    3.977740545
## Pclass      -1.1721300295 0.177781841  -6.59308072   -1.520576035
## Sexmale     -2.4179588065 0.238131631 -10.15387497   -2.884688227
## Age         -0.0390111261 0.008962287  -4.35280924   -0.056576886
## SibSp       -0.3063471686 0.145489422  -2.10563190   -0.591501196
## Parch       -0.0417291402 0.134205198  -0.31093535   -0.304766495
## Fare         0.0001340541 0.002865801   0.04677718   -0.005482813
## EmbarkedQ   -0.8748998709 0.678602466  -1.28926715   -2.204936265
## EmbarkedS   -0.4498435817 0.295432721  -1.52265998   -1.028881076
##             conf.int_higher
## (Intercept)     6.707038978
## Pclass         -0.823684024
## Sexmale        -1.951229386
## Age            -0.021445366
## SibSp          -0.021193141
```

```
## Parch            0.221308215
## Fare             0.005750921
## EmbarkedQ        0.455136523
## EmbarkedS        0.129193912
```

The confidence intervals with size α gives us the possible values of the said variable. Generally, if this confidence interval includes 0 as a possible value then that variable is insignificant.

Part G

$$confidence\ interval \Rightarrow lower < \ \beta_i < higher$$

$$\Rightarrow e^{lower} < e^{\beta_i} < e^{higher}$$

$$\Rightarrow \frac{e^{lower}}{e^{\beta_i}+1} < \frac{e^{\beta_i}}{e^{\beta_i}+1} < \frac{e^{higher}}{e^{\beta_i}+1}$$

```
conf %>% mutate(conf.int_lower_exponential = exp(conf.int_lower)/(exp(Estimate)+1),
                conf.int_higher_exponential = exp(conf.int_higher)/(exp(Estimate)+1))
```

```
##                   Estimate   std.error        z.val conf.int_lower
## (Intercept)   5.3423897613 0.696262394   7.67295463    3.977740545
## Pclass       -1.1721300295 0.177781841  -6.59308072   -1.520576035
## Sexmale      -2.4179588065 0.238131631 -10.15387497   -2.884688227
## Age          -0.0390111261 0.008962287  -4.35280924   -0.056576886
## SibSp        -0.3063471686 0.145489422  -2.10563190   -0.591501196
## Parch        -0.0417291402 0.134205198  -0.31093535   -0.304766495
## Fare          0.0001340541 0.002865801   0.04677718   -0.005482813
## EmbarkedQ    -0.8748998709 0.678602466  -1.28926715   -2.204936265
## EmbarkedS    -0.4498435817 0.295432721  -1.52265998   -1.028881076
##              conf.int_higher conf.int_lower_exponential
## (Intercept)      6.707038978                 0.25425382
## Pclass          -0.823684024                 0.16689688
## Sexmale         -1.951229386                 0.05130111
## Age             -0.021445366                 0.48171206
## SibSp           -0.021193141                 0.31880990
## Parch            0.221308215                 0.37633832
## Fare             0.005750921                 0.49723277
## EmbarkedQ        0.455136523                 0.07781584
## EmbarkedS        0.129193912                 0.21823323
##              conf.int_higher_exponential
## (Intercept)                    3.8957110
## Pclass                         0.3350461
## Sexmale                        0.1304736
## Age                            0.4989361
## SibSp                          0.5639147
## Parch                          0.6368685
## Fare                           0.5028500
## EmbarkedQ                      1.1125587
## EmbarkedS                      0.6948106
```

After this transformation now this confidence interval became the probabilities confidence interval.

Part H

```
hoslem.test(titanic_train$Survived, model.titanic$fitted.values)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  titanic_train$Survived, model.titanic$fitted.values
## X-squared = 30.743, df = 8, p-value = 0.0001561
```

Since our value is more extreme we will reject H_0 and say that model is not a good fit.

Part I

```
predictions <- ifelse(predict(model.titanic, titanic_test %>%
                                select(-Survived), type='response') > 0.5, 1, 0)

model.df <- data.frame(real=titanic_test$Survived, prediction=predictions)

t(table(model.df))
```

```
##           real
## prediction  0  1
##          0 79 12
##          1  6 46
```

```
sensitivity <- 44/(44+14)
specificity <- 66/(66+19)
prevalence <- (66+44)/(66+14+19+44)
noinfrate <- (66+19)/(66+14+19+44)

ppv <- sensitivity*prevalence/((sensitivity+prevalence)+
                                 ((1-specificity)*(1-prevalence)))
npv <- sensitivity*(1-prevalence)/(((1-sensitivity)+prevalence)+
                                     ((specificity)*(1-prevalence)))

detection.rate <- 44/(66+14+19+44)
detection.prevalence <- (44+14)/(66+14+19+44)
balanced.acc <- (sensitivity+specificity)/2
precision = 44/(14+44)
recall = 44/(19+44)

sensitivity
```

```
## [1] 0.7586207
```

```
specificity
```

```
## [1] 0.7764706
```

```
noinfrate
```

```
## [1] 0.5944056
```

```
prevalence
```

```
## [1] 0.7692308
```

```r
ppv
```

```
## [1] 0.3694703
```

```r
npv
```

```
## [1] 0.1471398
```

```r
detection.rate
```

```
## [1] 0.3076923
```

```r
detection.prevalence
```

```
## [1] 0.4055944
```

```r
balanced.acc
```

```
## [1] 0.7675456
```

```r
precision
```

```
## [1] 0.7586207
```

```r
recall
```

```
## [1] 0.6984127
```

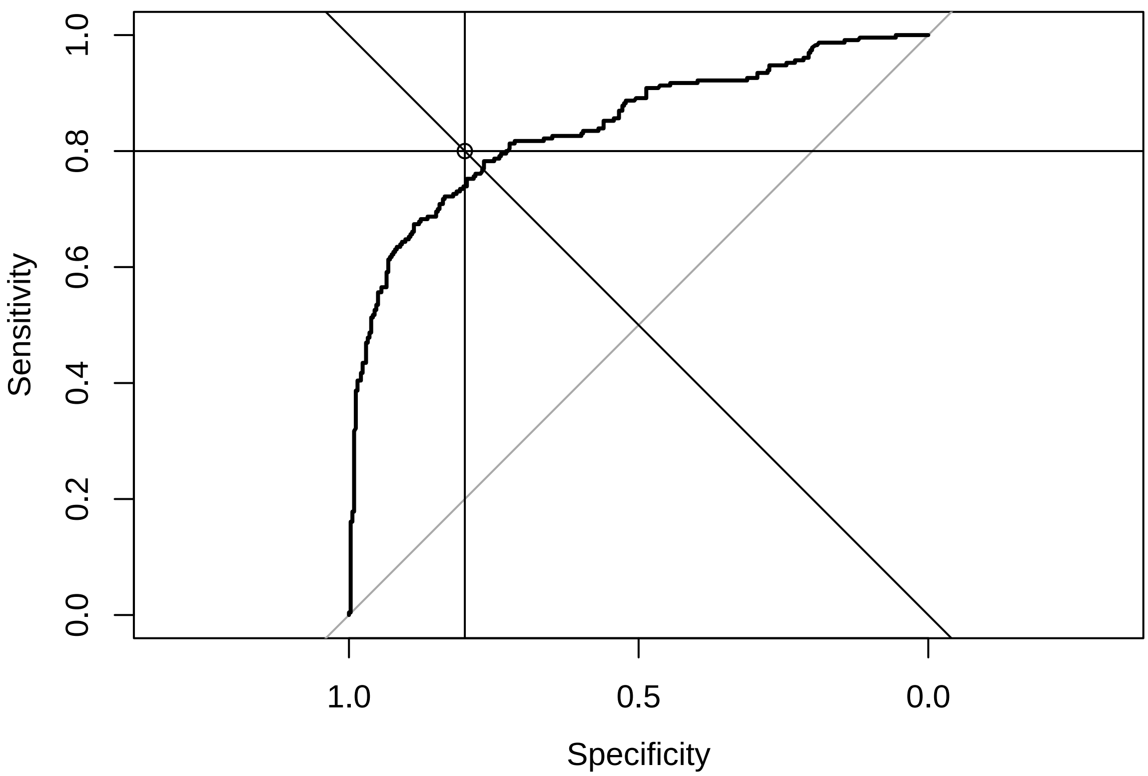
```r
confusionMatrix(factor(model.df$prediction), factor(model.df$real), positive = "0")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 79 12
##          1  6 46
##
##                Accuracy : 0.8741
##                  95% CI : (0.8084, 0.9237)
##     No Information Rate : 0.5944
##     P-Value [Acc > NIR] : 2.047e-13
##
##                   Kappa : 0.7346
##
##  Mcnemar's Test P-Value : 0.2386
##
##             Sensitivity : 0.9294
##             Specificity : 0.7931
##          Pos Pred Value : 0.8681
##          Neg Pred Value : 0.8846
##              Prevalence : 0.5944
##          Detection Rate : 0.5524
##    Detection Prevalence : 0.6364
##       Balanced Accuracy : 0.8613
##
##        'Positive' Class : 0
##
```

same.

Part J

```r
g <- roc(Survived ~ model.titanic$fitted.values, data=titanic_train, quiet = TRUE)
plot(g)
points(.8,.8)
abline(0.8,0)
abline(v=0.8)
abline(0,1)
```

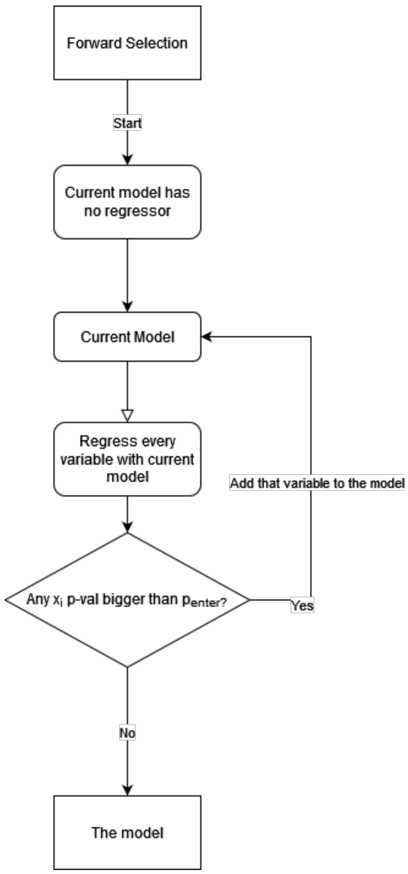


0.8 for both sensitivity and specificity seems like the most efficient point. This plot visualizes the trade off between sensitivity and specificity, best place is the most left upper corner.

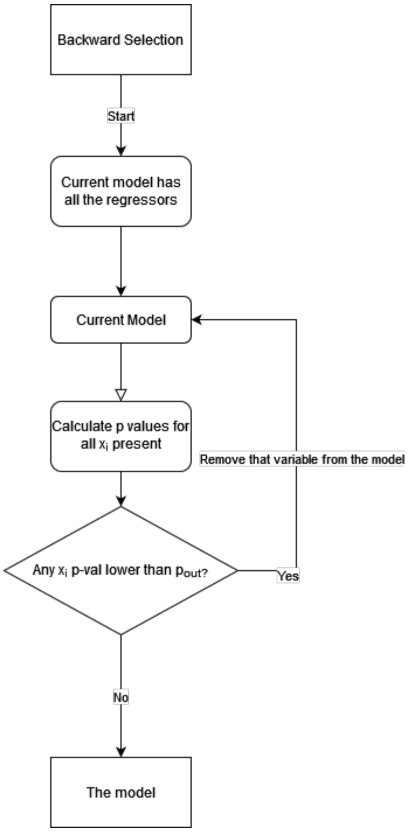
Q2

Part A

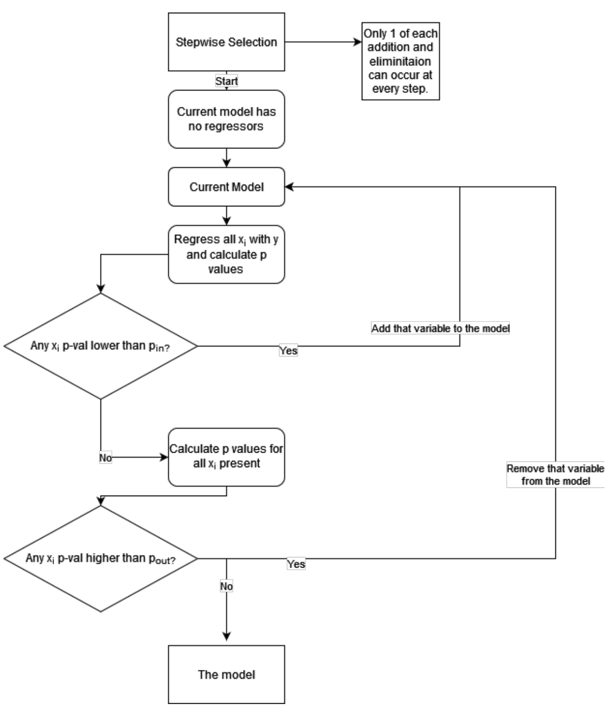
Forward Selection



Backwards elimination



Stepwise regression



Part B

Model validation may not be mentioned alot in papers but its a must have component of model building as it is the process of making sure that model works and can be used in real life.

Part C

- Gender
- Age
- n.projects
- Graduation.CGPA
- haslover
- personality
- enthusiasm
- living.place
- living.conditions.score
- monthly.earnings
- has.necessary.equipment
- disability.severity

Part D

Thats because we want to be adding variables less often than we remove them. If the p value for the entry is higher than p value for removal then that means we will needlessly increase the size of the model with not so good variables.

Q3

```r
train <- read.xlsx("./job_model_building_data.xlsx")
test <- read.xlsx("./job_validation_data.xlsx")

head(train)
```

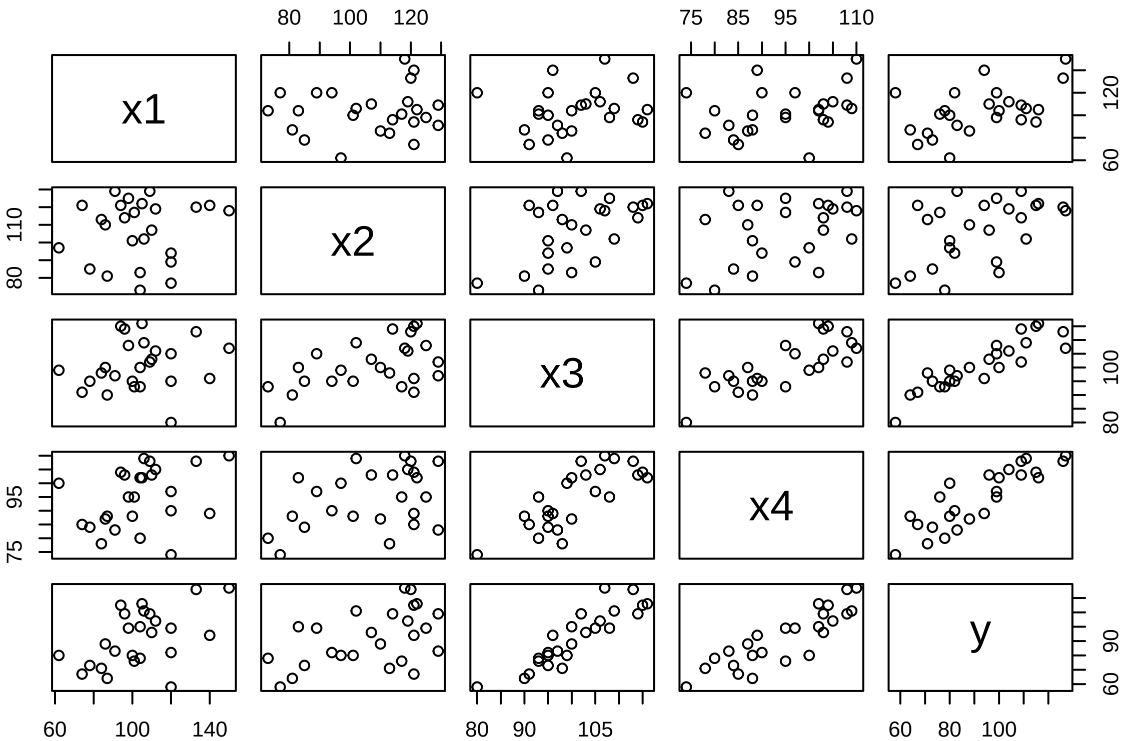
```
##    x1  x2  x3  x4  y
## 1  86 110 100  87 88
## 2  62  97  99 100 80
## 3 110 107 103 103 96
## 4 101 117  93  95 76
## 5 100 101  95  88 80
## 6  78  85  95  84 73
```

```r
head(test)
```

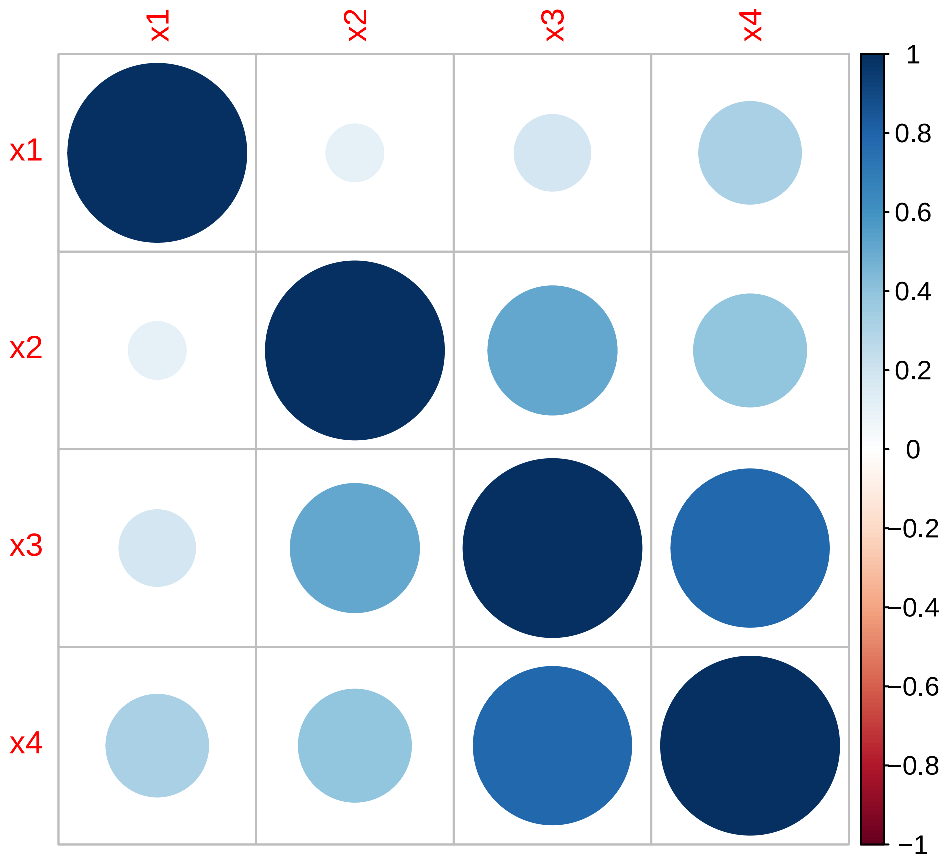
```
##    x1  x2  x3 x4  y
## 1  65 109  88 84 58
## 2  85  90 104 98 92
## 3  93  73  91 82 71
## 4  95  57  95 85 77
## 5 102 139 101 92 92
## 6  63 101  93 84 66
```

Part A

```r
pairs(train)
```




```
corrplot(cor(train %>% select(-y)))
```



x3 & x4 seems to be highly correlated, there may be multicolinearity problem. Also in the plots, y~x3 and y~x4 looks near identical.

```
vif(lm(y~., data=train))
```

```
##       x1       x2       x3       x4
## 1.138043 1.369512 3.016549 2.834776
```

These vif values are not that high. But if we wanted to be more conservative we could drop either x3 or x4.

Part B

```
model.new <- lm(y~., data=train)

summary(model.new)
```

```
##
## Call:
## lm(formula = y ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9779 -3.4506  0.0941  2.4749  5.9959
##
## Coefficients:
##               Estimate Std. Error t value        Pr(>|t|)
## (Intercept) -124.38182    9.94106 -12.512 0.0000000000648 ***
```

```
## x1             0.29573    0.04397   6.725 0.0000015237103 ***
## x2             0.04829    0.05662   0.853         0.40383
## x3             1.30601    0.16409   7.959 0.0000001261726 ***
## x4             0.51982    0.13194   3.940         0.00081 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.099 on 20 degrees of freedom
## Multiple R-squared:  0.9629, Adjusted R-squared:  0.9555
## F-statistic: 129.7 on 4 and 20 DF,  p-value: 5.262e-14
```

To me it seems like x2 should be dropped. That is because 0 is within the confidence interval.

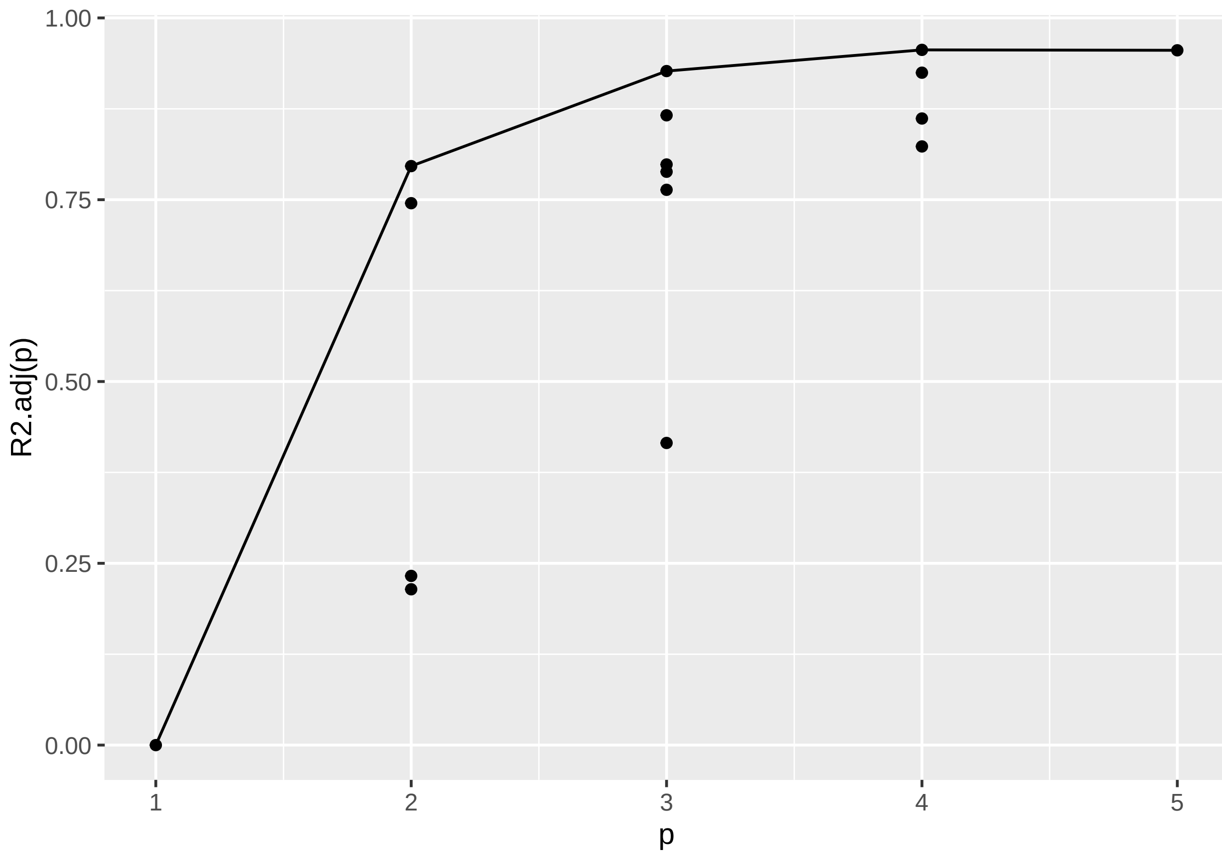
Part C

```r
number.of.predictors <- 4
predictor.names <- colnames(train)[1:number.of.predictors]
ind <- expand.grid(c(1,0), c(1,0), c(1,0), c(1,0))
c <- c()
for(i in 1:nrow(ind)){
c[i] <- paste(predictor.names[as.logical(ind[i,])], collapse="+")
}
c[16] <- 1
c <- c[order(nchar(c))]
c <- paste("y ~", c)
all <- data.frame(formula=rep(NA, 16), rsq.p= rep(NA, 16))
for(i in 1:length(c)){
    all$formula[i] <- c[i]
    all$rsq.p[i] <- summary(lm(as.formula(c[i]), data=train))$adj.r.squared
}
```

```r
all$p <- c(1,2,2,2,2,3,3,3,3,3,3,4,4,4,4,5)
all
```

```
##              formula     rsq.p p
## 1              y ~ 1 0.0000000 1
## 2             y ~ x4 0.7452170 2
## 3             y ~ x3 0.7962344 2
## 4             y ~ x2 0.2142762 2
## 5             y ~ x1 0.2326452 2
## 6          y ~ x3+x4 0.8660988 3
## 7          y ~ x2+x4 0.7635916 3
## 8          y ~ x1+x4 0.7984716 3
## 9          y ~ x2+x3 0.7884436 3
## 10         y ~ x1+x3 0.9269043 3
## 11         y ~ x1+x2 0.4154853 3
## 12      y ~ x2+x3+x4 0.8616797 4
## 13      y ~ x1+x3+x4 0.9560482 4
## 14      y ~ x1+x2+x4 0.8232664 4
## 15      y ~ x1+x2+x3 0.9246779 4
## 16 y ~ x1+x2+x3+x4 0.9554702 5
```

```r
max.rsq <- group_by(all, p) %>% summarize(max.rsq=max(rsq.p))
ggplot(all, aes(p, rsq.p)) + geom_point() + ylab("R2.adj(p)") +
  geom_line(data= max.rsq, aes(p,max.rsq))
```



When the formula is $y = x1 + x3 + x4$ the R^2_{adj} has the highest value.

Part D

```
model <- lm(y~., data=train)
ols_step_both_p(model, pent = 0.05, prem = 0.1)$model
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Coefficients:
## (Intercept)           x3           x1           x4
##   -124.2000       1.3570       0.2963       0.5174
```

Part E

They are the same..

Part F

```
model.new <- lm(y~.-x2, train)
PRESS <- function(model){
  i <- residuals(model)/(1 - lm.influence(model)$hat)
  sum(i^2)
```

```
}
PRESS(model.new)
```

```
## [1] 471.452
```

$$SSE: \sum(\hat{y} - \bar{y})^2$$

$$MSE: \frac{SSE}{n-p}$$

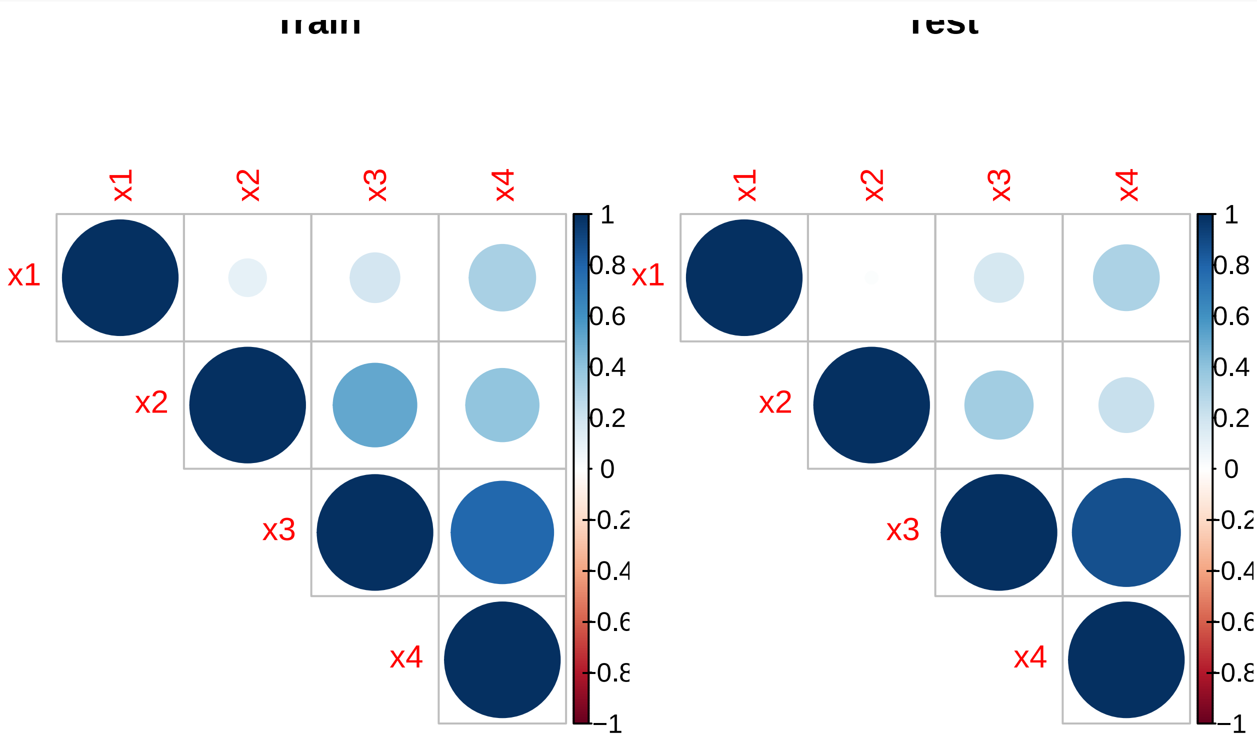
```
sse <- sum((model.new$fitted.values - train$y)**2)
sse
```

```
## [1] 348.197
```

```
mse <- sse/(nrow(train)-1)
```

Part G

```
par(mfrow=c(1,2))
corrplot(cor(train %>% select(-y)), type="upper", title = "Train")
corrplot(cor(test %>% select(-y)), type="upper", title = "Test")
```



corplots are near identical with the sole difference being correlation of x2~x4

Part H

```
summary(lm(y~.-x2, test))
```

```
##
## Call:
## lm(formula = y ~ . - x2, data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.4619 -2.3836  0.6834  2.1123  7.2394
##
## Coefficients:
##               Estimate Std. Error t value      Pr(>|t|)
## (Intercept) -122.76705   11.84783 -10.362 0.00000000104 ***
## x1             0.31238    0.04729   6.605 0.00000153528 ***
## x3             1.40676    0.23262   6.048 0.00000530801 ***
## x4             0.42838    0.19749   2.169        0.0417 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.284 on 21 degrees of freedom
## Multiple R-squared:  0.9489, Adjusted R-squared:  0.9416
## F-statistic:   130 on 3 and 21 DF,  p-value: 1.017e-13
```

```
summary(model.new)
```

```
##
## Call:
## lm(formula = y ~ . - x2, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4579 -3.1563 -0.2057  1.8070  6.6083
##
## Coefficients:
##               Estimate Std. Error t value        Pr(>|t|)
## (Intercept) -124.20002    9.87406 -12.578 0.0000000000304 ***
## x1             0.29633    0.04368   6.784 0.0000010397036 ***
## x3             1.35697    0.15183   8.937 0.0000000133381 ***
## x4             0.51742    0.13105   3.948        0.000735 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.072 on 21 degrees of freedom
## Multiple R-squared:  0.9615, Adjusted R-squared:  0.956
## F-statistic:   175 on 3 and 21 DF,  p-value: 5.16e-15
```

These two models look very alike but the importance of x4 seems to be much higher in the test model. This might mean that model is not correctly specified.

data.frame(train=c(4.284, 0.94), test=c(4.072, 0.96), row.names = c("MSE", "Rsquared"))

As we can see the values are near identical. This means the model has high chance to work with new data. It is useful.

Part I

```r
mspe <- sum((predict(model.new, test %>%
                       select(-y), type="response")-test$y)**2)/(nrow(test)-1)

mspe
```

```
## [1] 16.3643
```

```r
mse
```

```
## [1] 14.50821
```

mean squared errors are close also.

Part J

```r
df <- rbind(train, test)
model.full <- lm(y~.-x2, df)

summary(model.new)$coefficients[,'Std. Error']
```

```
## (Intercept)          x1          x3          x4
##  9.87405909  0.04367948  0.15183247  0.13105392
```

```r
data.frame(train = summary(model.new)$coefficients[,'Std. Error'],
           full = summary(model.full)$coefficients[,'Std. Error'])
```

```
##                  train       full
## (Intercept) 9.87405909 7.16508011
## x1          0.04367948 0.03071562
## x3          0.15183247 0.12280465
## x4          0.13105392 0.10475295
```

All standard deviations are lower, although not that much.