Homework 2.3

Valeriia

20 05 2020

Load all needed libraries.

```
library(MASS)
library(data.table)
library(ggplot2)
library(caret)
library(boot)
library(corrplot)
data(Boston)
```

Check our data. It's ok.

```
bos <- Boston
str(bos)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
dim(bos)
```

```
## [1] 506  14
```

```
sum(is.na(bos))
```

```
## [1] 0
```

```r
summary(bos)
```

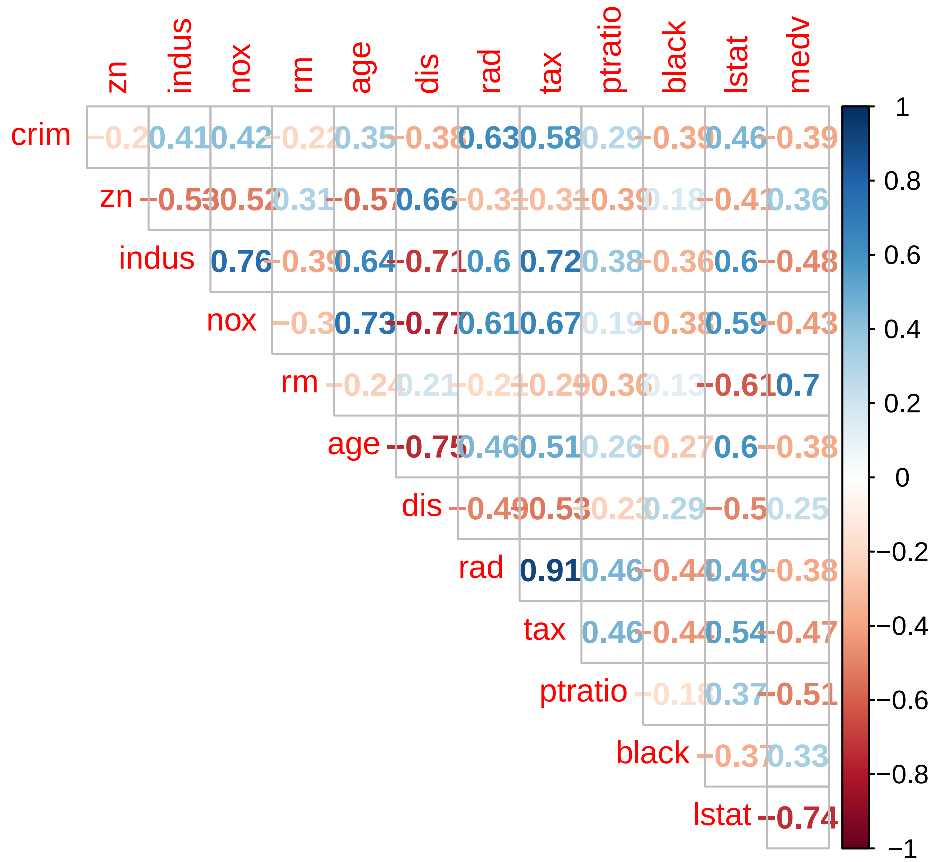
```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
sum(duplicated(bos))
```

```
## [1] 0
```

Charles River dummy variable (= 1 if tract bounds river; 0 otherwise). Drop it. Check correlation.

```r
bos <- bos[,-4]
corrplot(cor(bos), method = "number", type = "upper", diag = FALSE)
```



Cathecorize data.

```
bos <- data.table(bos)
bos[,medv_F:= cut(medv,c(0,quantile(bos$medv, 0.33),quantile(bos$medv, 0.66),
                        max(bos$medv)),labels=c("1","2","3"))]
bos[,table(medv_F)]
```

```
## medv_F
##   1   2   3
## 167 167 172
```

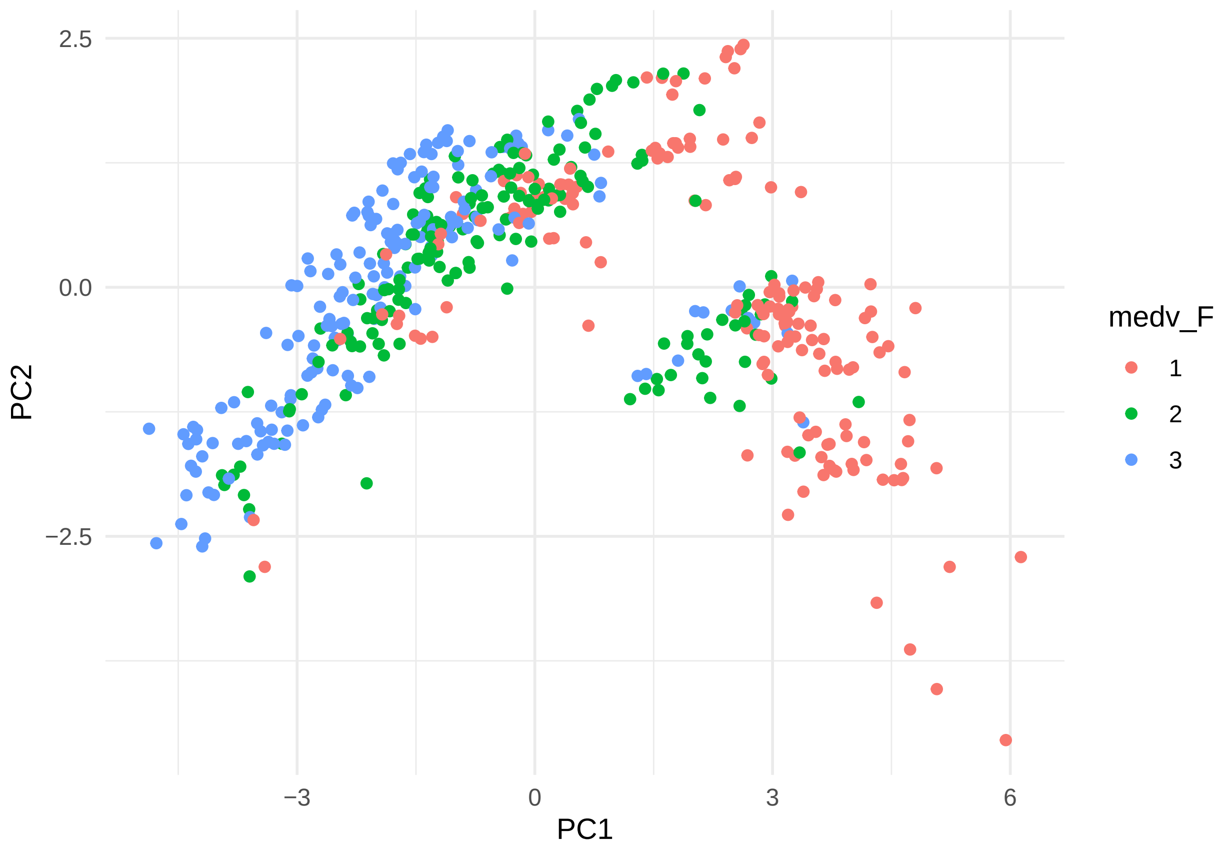
PCA.

```
pca <- prcomp(bos[,-c(13,14)], center = TRUE, scale. = TRUE)
pca
```

```
## Standard deviations (1, .., p=12):
##  [1] 2.4752210 1.1586541 1.0861790 0.9138194 0.8152738 0.7330805 0.6296169
##  [8] 0.5263720 0.4693245 0.4314643 0.4114793 0.2542551
##
## Rotation (n x k) = (12 x 12):
##                PC1         PC2         PC3         PC4         PC5         PC6
## crim     0.2510194 -0.40124935  0.06905890 -0.07537120  0.20811692 -0.77883466
## zn      -0.2562763 -0.43910123  0.09079765 -0.30453280  0.35173539  0.27063296
## indus    0.3466252  0.10826541  0.03147503  0.01032982  0.09094655  0.34042847
```

```
## nox      0.3427732  0.16538852  0.23634943 -0.14816181  0.14361054  0.19010878
## rm      -0.1893443 -0.07676264  0.67862805  0.39317754 -0.10468673 -0.07757913
## age      0.3135926  0.31549763  0.16446704 -0.03415323  0.04226173 -0.12864962
## dis     -0.3214520 -0.32731960 -0.25420475 -0.07645279  0.01083277  0.11493067
## rad      0.3198193 -0.38437642  0.11313425  0.21734697  0.16394203  0.14004949
## tax      0.3385180 -0.32057097  0.07803058  0.14109504  0.21000125  0.31042396
## ptratio  0.2050739 -0.17273359 -0.48516673  0.60814980 -0.24573826  0.01417231
## black   -0.2030293  0.33625146 -0.18803264  0.36330273  0.81001861 -0.09211058
## lstat    0.3098245  0.03364522 -0.29715025 -0.38592449  0.06096419 -0.08775570
##                  PC7         PC8         PC9        PC10        PC11
## crim     0.158230415 -0.26179898  0.01980470 -0.11039262 -0.08663749
## zn      -0.403359872 -0.35858760  0.26689361  0.26335598  0.07080081
## indus    0.173213403 -0.64380852 -0.36378621 -0.30263190  0.11400759
## nox      0.076735083  0.01984964  0.23045723  0.11122392 -0.80409242
## rm      -0.329939431 -0.04637159 -0.43204790  0.05329734 -0.15277946
## age     -0.602218414  0.06657278  0.36352844 -0.45777362  0.21328031
## dis     -0.118900334  0.15077227 -0.16897171 -0.69863534 -0.38969478
## rad      0.080571835  0.46980760  0.02306152  0.03388316  0.10601899
## tax      0.079371592  0.17846255 -0.03592292 -0.10173769  0.21630840
## ptratio -0.313349725 -0.25684385  0.15388089  0.17261849 -0.21044696
## black   -0.008231654  0.04631318 -0.09701329  0.02002712 -0.04151870
## lstat   -0.423864339  0.19492701 -0.60053449  0.26891408 -0.05644192
##                 PC12
## crim    -0.044517237
## zn       0.081746955
## indus    0.247896886
## nox     -0.047399129
## rm      -0.047915403
## age      0.035654465
## dis      0.019094950
## rad      0.635066577
## tax     -0.720833262
## ptratio -0.019118760
## black    0.002297948
## lstat   -0.020576836
```

```
dtp <- data.frame('medv_F' = bos$medv_F, pca$x[,1:2])
ggplot(data = dtp) +
       geom_point(aes(x = PC1, y = PC2, col = medv_F)) +
       theme_minimal()
```



KNN.

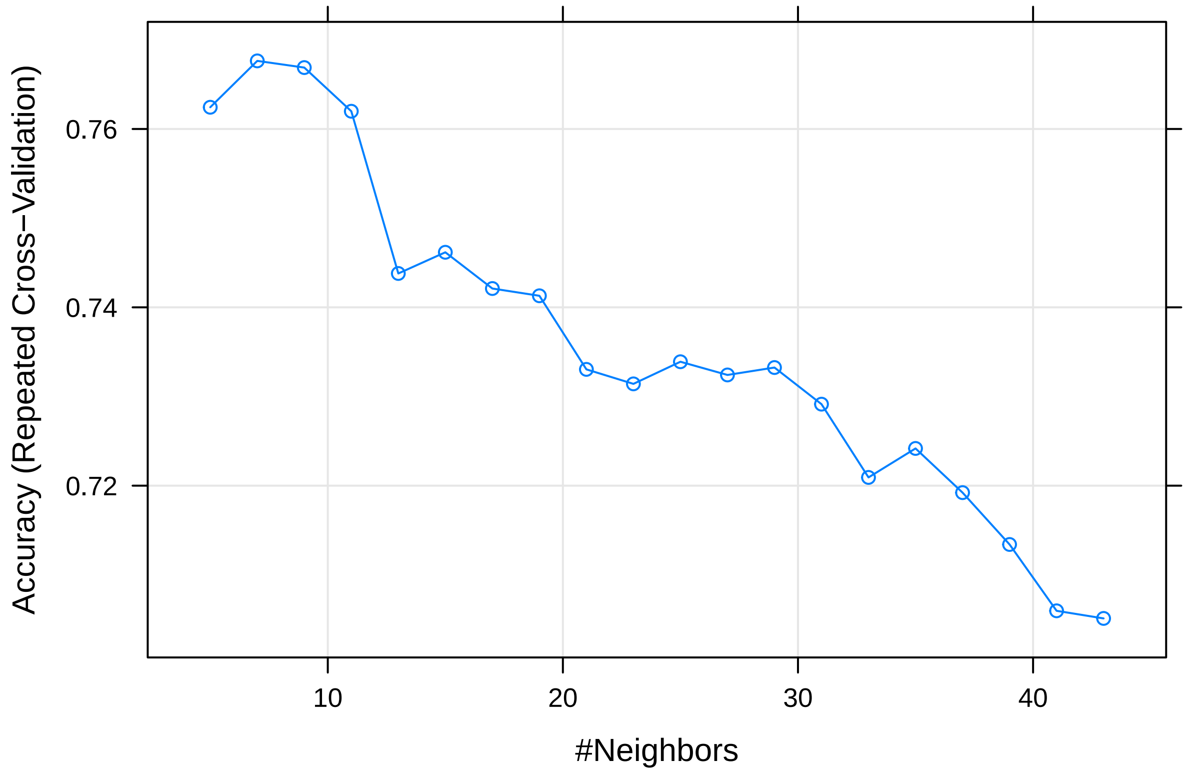
```
set.seed(3)
intrain <- createDataPartition(y = bos$medv_F, p= 0.8, list = FALSE)
training <- bos[intrain,-13]
testing <- bos[-intrain,-13]
# Repeat several times, 10 blocks, 3 times
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

knn_fit <- train(medv_F ~ ., data = training, method = "knn",
 trControl=trctrl,
 preProcess = c("center", "scale"),
 tuneLength = 20)
knn_fit
```

```
## k-Nearest Neighbors
##
## 406 samples
##  12 predictor
##   3 classes: '1', '2', '3'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 366, 365, 365, 367, 365, 365, ...
## Resampling results across tuning parameters:
##
```

```
##    k   Accuracy   Kappa
##     5  0.7624363  0.6437411
##     7  0.7676367  0.6516739
##     9  0.7668867  0.6506242
##    11  0.7619864  0.6432025
##    13  0.7437962  0.6158727
##    15  0.7461732  0.6193821
##    17  0.7421072  0.6134217
##    19  0.7412952  0.6121960
##    21  0.7330411  0.5998296
##    23  0.7314171  0.5974494
##    25  0.7338958  0.6011902
##    27  0.7324142  0.5989529
##    29  0.7332474  0.6002403
##    31  0.7291398  0.5941370
##    33  0.7209254  0.5818946
##    35  0.7241755  0.5867250
##    37  0.7192171  0.5792872
##    39  0.7134030  0.5706491
##    41  0.7059630  0.5595450
##    43  0.7051114  0.5583733
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```r
# The hieghest accuracy is in k=7
plot(knn_fit)
```




```
test_pred <- predict(knn_fit, newdata =testing)
head(data.frame(test_pred, testing$medv_F))
```

```
##   test_pred testing.medv_F
## 1         2              2
## 2         1              2
## 3         2              2
## 4         2              2
## 5         3              2
## 6         2              1
```

```
table(test_pred, Real = testing$medv_F)
```

```
##            Real
## test_pred  1  2  3
##          1 29  6  1
##          2  4 20  7
##          3  0  7 26
```

```
knn_fit <- knn3Train(train = training[,-13], test = testing[,-13], k=7, cl = training$medv_F)
xtab <- table(knn_fit, Real = testing$medv_F)
xtab
```

```
##          Real
```

```
## knn_fit  1  2  3
##       1 29 10  4
##       2  2 15 12
##       3  2  8 18
```

```r
accuracy = sum(knn_fit == testing$medv_F)/length(testing$medv_F)
precision = xtab[1,1]/sum(xtab[,1])
recall = xtab[1,1]/sum(xtab[1,])
f = 2 * (precision * recall) / (precision + recall)

#Accuracy - Accuracy is the most intuitive performance measure and it
#is simply a ratio of correctly predicted observation to the total observations. (TP+TN)/(FP+FN+TP+TN)

paste0("Accuracy:", accuracy)
```

```
## [1] "Accuracy:0.62"
```

```r
#Precision - Precision is the ratio of correctly predicted
#positive observations to the total predicted positive observations. TP/(TP+FP)

paste0("Precision:", precision)
```

```
## [1] "Precision:0.878787878787879"
```

```r
#Recall (Sensitivity) - Recall is the ratio of correctly predicted positive
#observations to the all observations in actual class - yes. TP/(TP+FN)

paste0("Recall:", recall)
```

```
## [1] "Recall:0.674418604651163"
```

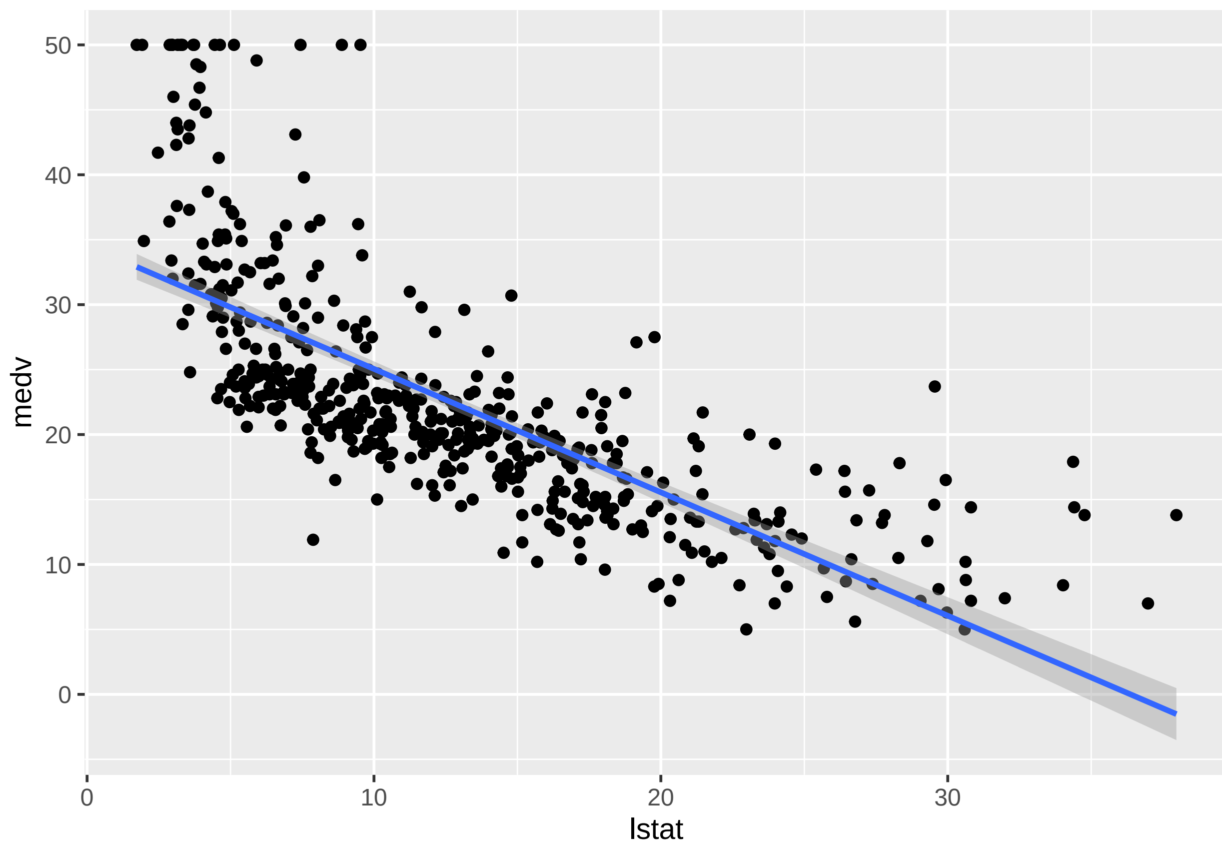
```r
#F1 Score = 2*(Recall * Precision) / (Recall + Precision)

paste0("F:", f)
```

```
## [1] "F:0.763157894736842"
```

Polynomial regression

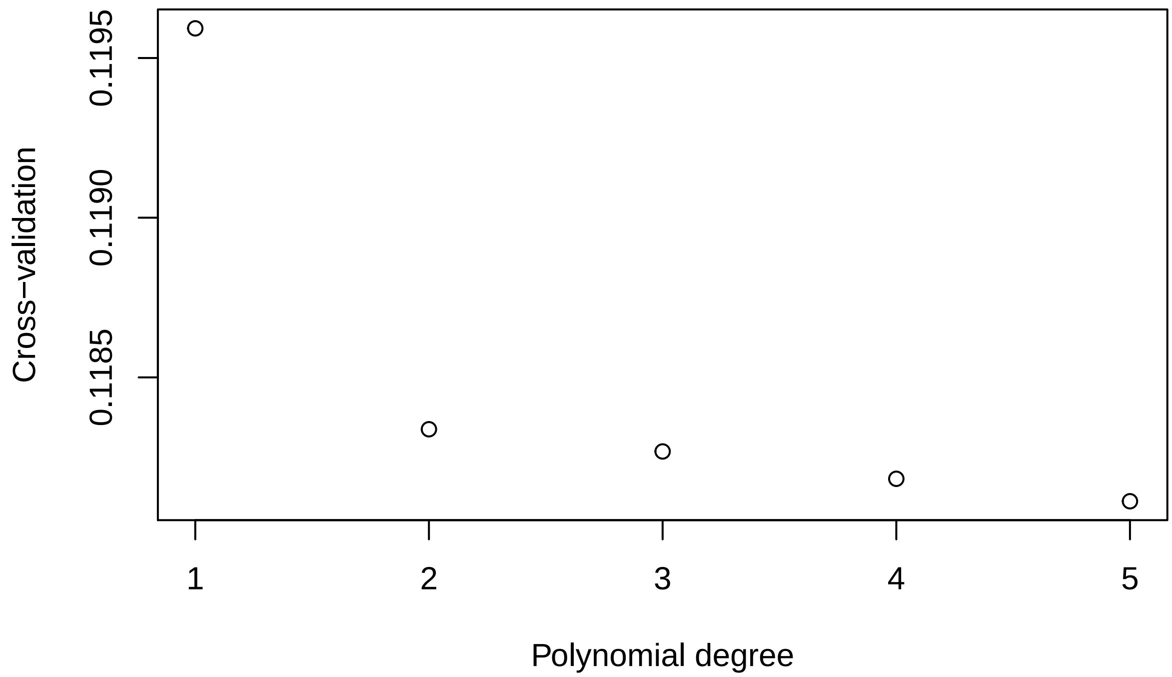
```r
qplot(lstat, medv, data = Boston, geom = c("point", "smooth"), method = "lm")
```




```r
bos[,medv_B:= cut(medv,c(0,median(bos$medv),max(bos$medv)),labels=c("low","high"))]
bos[,table(medv_B)]
```

```
## medv_B
##  low high
##  256  250
```

```r
set.seed(3)
intrain <- createDataPartition(y = bos$medv_B, p= 0.8, list = FALSE)
bos1 <- bos[,-c(13,14)]
training <- bos[intrain,-c(13,14)]
testing <- bos[-intrain,-c(13,14)]
#k-fold CV
errors <- c()
for (i in 1:5){
  g <- glm(medv_B ~ poly(lstat,i), family = "binomial", data = bos1)
  errors[i] <- cv.glm(bos1, g)$delta[1]
}
plot(x = 1:5, y = errors, xlab = 'Polynomial degree', ylab = 'Cross-validation')
```




```r
mod <- glm(medv_B ~ poly(lstat, 5), data = training, family = "binomial")
summary(mod)
```

```
##
## Call:
## glm(formula = medv_B ~ poly(lstat, 5), family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0865  -0.5115   0.0000   0.5056   2.2062
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -16.29      13.27  -1.227    0.220
## poly(lstat, 5)1 -1131.94     832.37  -1.360    0.174
## poly(lstat, 5)2 -1394.55    1094.94  -1.274    0.203
## poly(lstat, 5)3 -1241.24     925.87  -1.341    0.180
## poly(lstat, 5)4  -613.34     461.16  -1.330    0.184
## poly(lstat, 5)5  -221.79     154.13  -1.439    0.150
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 561.39  on 404  degrees of freedom
## Residual deviance: 284.08  on 399  degrees of freedom
## AIC: 296.08
```

```
##
## Number of Fisher Scoring iterations: 15
```

```
pred <- predict(mod, type = 'response') > 0.5
table(pred, Real = training$medv_B)
```

```
##        Real
## pred    low high
##   FALSE 170   33
##   TRUE   35  167
```

```
mod2 <- glm(medv_B ~ lstat, data = testing, family = "binomial")
summary(mod2)
```

```
##
## Call:
## glm(formula = medv_B ~ lstat, family = "binomial", data = testing)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6348  -0.5948  -0.0301   0.5975   3.2198
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.66041    0.73695   4.967 6.80e-07 ***
## lstat       -0.29910    0.06008  -4.978 6.41e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 140.006  on 100  degrees of freedom
## Residual deviance:  86.562  on  99  degrees of freedom
## AIC: 90.562
##
## Number of Fisher Scoring iterations: 5
```

```
pred2 <- predict(mod2, type = 'response', newdata = testing) > 0.5
table(pred2, Real = testing$medv_B)
```

```
##        Real
## pred2   low high
##   FALSE  39    8
##   TRUE   12   42
```