HW2.3

Mary Futey

5/5/2020

Load data check, dimensions, structure and for NAs

```r
data(BostonHousing)
boston <- BostonHousing
str(boston)
```

```
## '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   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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    : num  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 ...
##  $ b      : 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 ...
```

```r
print(paste0("cols: ", ncol(boston)))
```

```
## [1] "cols: 14"
```

```r
print(paste0("# rows: ", nrow(boston)))
```

```
## [1] "# rows: 506"
```

```r
#check for NAs: looks good
sum(is.na(boston))
```

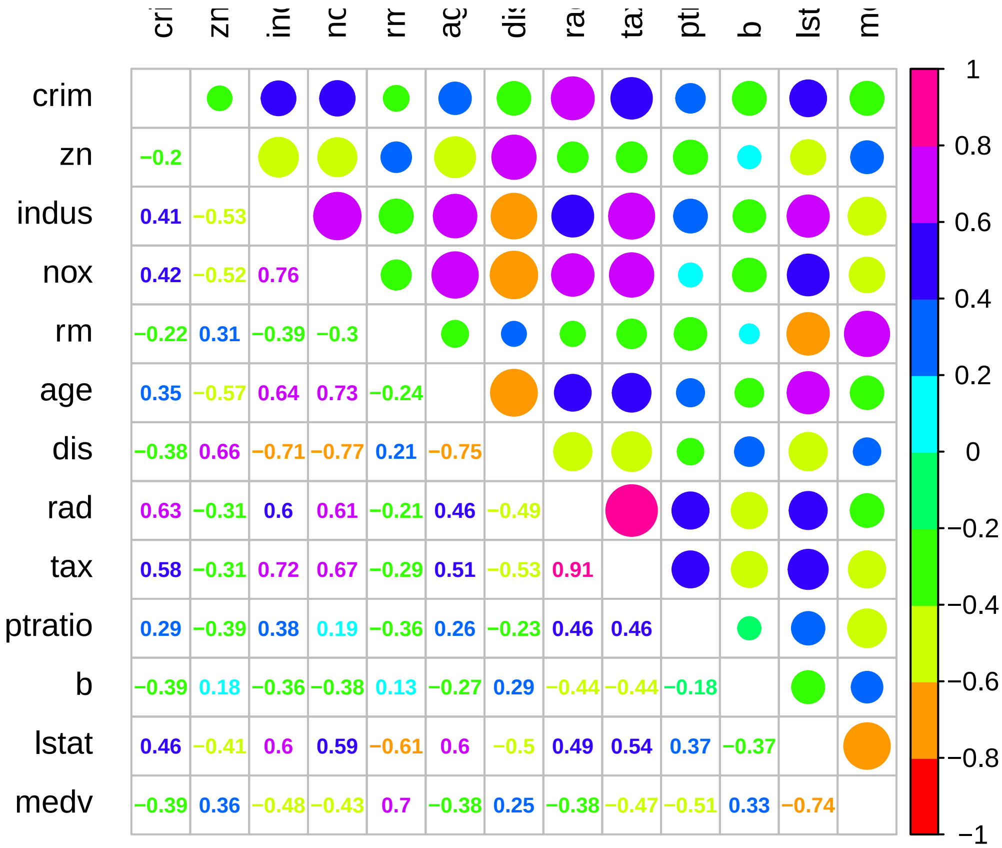
```
## [1] 0
```

```r
# 506 observations, 13 predictors

# remove Charles river variable
boston_sub <- boston[, c(1:3, 5:14)]
```

correlation plot

```r
cor_data <- cor(boston_sub, method ='pearson', use = 'pairwise.complete.obs')
p1 <- corrplot.mixed(cor_data, lower ='number',tl.offset = 1, tl.cex= 1,
```

```r
tl.col = "black", number.cex = 0.7, tl.pos='lt',
lower.col=rainbow(10), upper.col=rainbow(10))
```




```r
p1
```

```
##               crim         zn      indus        nox         rm        age
## crim     1.0000000 -0.2004692  0.4065834  0.4209717 -0.2192467  0.3527343
## zn      -0.2004692  1.0000000 -0.5338282 -0.5166037  0.3119906 -0.5695373
## indus    0.4065834 -0.5338282  1.0000000  0.7636514 -0.3916759  0.6447785
## nox      0.4209717 -0.5166037  0.7636514  1.0000000 -0.3021882  0.7314701
## rm      -0.2192467  0.3119906 -0.3916759 -0.3021882  1.0000000 -0.2402649
## age      0.3527343 -0.5695373  0.6447785  0.7314701 -0.2402649  1.0000000
## dis     -0.3796701  0.6644082 -0.7080270 -0.7692301  0.2052462 -0.7478805
## rad      0.6255051 -0.3119478  0.5951293  0.6114406 -0.2098467  0.4560225
## tax      0.5827643 -0.3145633  0.7207602  0.6680232 -0.2920478  0.5064556
## ptratio  0.2899456 -0.3916785  0.3832476  0.1889327 -0.3555015  0.2615150
## b       -0.3850639  0.1755203 -0.3569765 -0.3800506  0.1280686 -0.2735340
## lstat    0.4556215 -0.4129946  0.6037997  0.5908789 -0.6138083  0.6023385
## medv    -0.3883046  0.3604453 -0.4837252 -0.4273208  0.6953599 -0.3769546
##                dis        rad        tax    ptratio          b      lstat
## crim    -0.3796701  0.6255051  0.5827643  0.2899456 -0.3850639  0.4556215
## zn       0.6644082 -0.3119478 -0.3145633 -0.3916785  0.1755203 -0.4129946
## indus   -0.7080270  0.5951293  0.7207602  0.3832476 -0.3569765  0.6037997
## nox     -0.7692301  0.6114406  0.6680232  0.1889327 -0.3800506  0.5908789
## rm       0.2052462 -0.2098467 -0.2920478 -0.3555015  0.1280686 -0.6138083
## age     -0.7478805  0.4560225  0.5064556  0.2615150 -0.2735340  0.6023385
## dis      1.0000000 -0.4945879 -0.5344316 -0.2324705  0.2915117 -0.4969958
```

```
## rad     -0.4945879  1.0000000  0.9102282  0.4647412 -0.4444128  0.4886763
## tax     -0.5344316  0.9102282  1.0000000  0.4608530 -0.4418080  0.5439934
## ptratio -0.2324705  0.4647412  0.4608530  1.0000000 -0.1773833  0.3740443
## b        0.2915117 -0.4444128 -0.4418080 -0.1773833  1.0000000 -0.3660869
## lstat   -0.4969958  0.4886763  0.5439934  0.3740443 -0.3660869  1.0000000
## medv     0.2499287 -0.3816262 -0.4685359 -0.5077867  0.3334608 -0.7376627
##               medv
## crim    -0.3883046
## zn       0.3604453
## indus   -0.4837252
## nox     -0.4273208
## rm       0.6953599
## age     -0.3769546
## dis      0.2499287
## rad     -0.3816262
## tax     -0.4685359
## ptratio -0.5077867
## b        0.3334608
## lstat   -0.7376627
## medv     1.0000000
```

Create categorial variable to do classification

```r
# create categorical medv (Median house value)
quant_00  = min(boston_sub$medv)
quant_25  = quantile(boston_sub$medv, 0.25)
quant_50  = quantile(boston_sub$medv, 0.50)
quant_75  = quantile(boston_sub$medv, 0.75)
quant_100 = max(boston_sub$medv)

rb = rbind(quant_00, quant_25, quant_50, quant_75, quant_100)
dimnames(rb)[[2]] = "Value"

boston_sub$medv_F[boston$medv >= quant_00 &
                   boston_sub$medv < quant_25]  = "first"
boston_sub$medv_F[boston$medv >= quant_25 &
                   boston_sub$medv <  quant_50]  = "second"
boston_sub$medv_F[boston$medv >= quant_50 &
                   boston_sub$medv <= quant_75] = "third"
boston_sub$medv_F[boston$medv >= quant_75 &
                   boston_sub$medv <= quant_100] = "fourth"
boston_sub$medv_F = factor(boston_sub$medv_F,
                    levels=c("first", "second", "third", "fourth"))

# check numbers in each class
table(boston_sub$medv_F)
```

```
##
##  first second  third fourth
##    127    124    123    132
```

```r
# remove numerical medv
boston_df <- boston_sub[,-13]
```

```r
str(boston_df)
```

```
## 'data.frame':    506 obs. of  13 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 ...
##  $ 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    : num  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 ...
##  $ b      : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv_F : Factor w/ 4 levels "first","second",..: 3 3 4 4 4 4 3 4 1 2 ...
```

pca

```r
# pca to check clusters
boston_pca <- boston_df[, -13]
pca <- prcomp(boston_pca, center = TRUE,
              scale. = TRUE)

autoplot(pca, data = boston_df, colour = "medv_F")
```

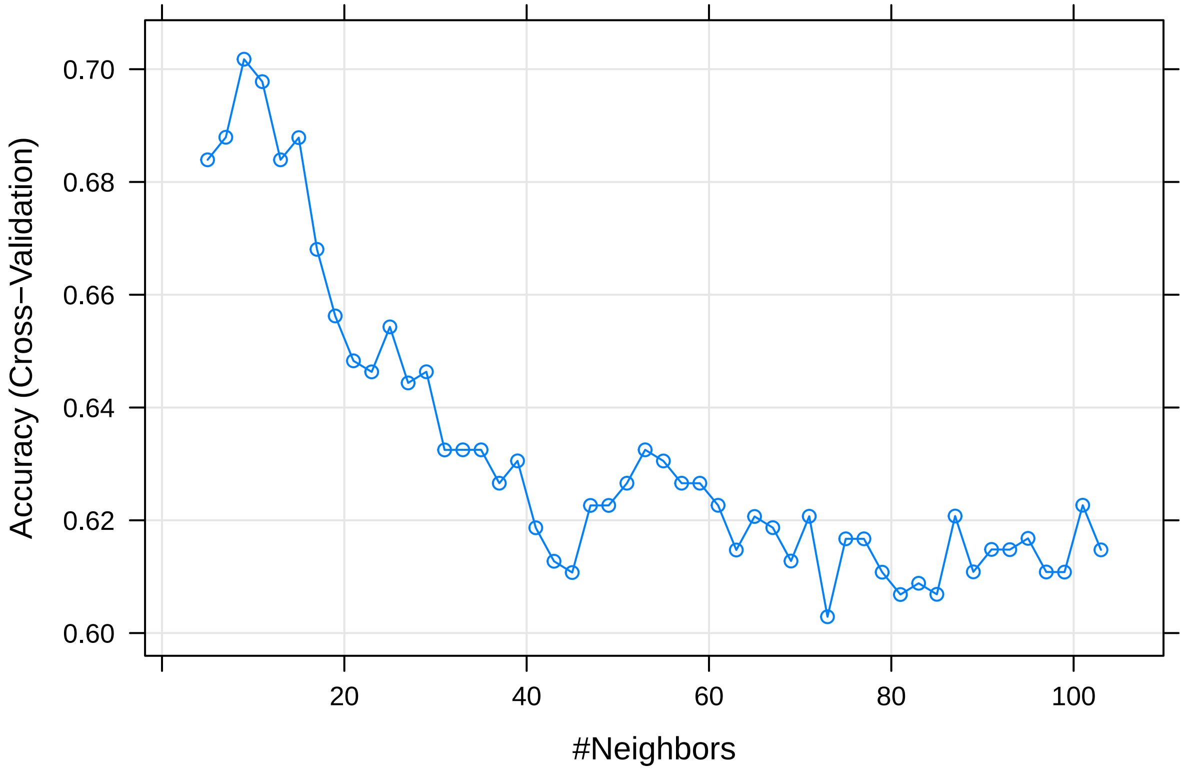
knn

```r
set.seed(33)

# create knn model
# should we exclude the response from the dataset?
knn <- train(medv_F ~ .,
               data = boston_df,
               method = "knn",
  # set 10-fold cross validation
  trControl = trainControl("cv", number = 5),
  # normalize data
  preProcess = c("center","scale"),
  # number of k values to check
  tuneLength = 50,
  metric = "Accuracy"
  )
knn
```

```
## k-Nearest Neighbors
##
## 506 samples
##  12 predictor
##   4 classes: 'first', 'second', 'third', 'fourth'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 405, 405, 405, 405, 404
## Resampling results across tuning parameters:
##
##   k    Accuracy   Kappa
##     5  0.6839255  0.5786099
##     7  0.6879247  0.5840216
##     9  0.7017666  0.6025119
##    11  0.6977868  0.5971826
##    13  0.6839255  0.5788848
##    15  0.6878664  0.5841981
##    17  0.6680450  0.5577526
##    19  0.6562415  0.5420761
##    21  0.6482819  0.5314933
##    23  0.6463211  0.5287987
##    25  0.6543001  0.5394592
##    27  0.6443603  0.5261907
##    29  0.6463405  0.5288834
##    31  0.6324791  0.5103871
##    33  0.6324985  0.5103545
##    35  0.6324985  0.5103854
##    37  0.6265774  0.5024509
##    39  0.6305378  0.5077700
##    41  0.6186760  0.4919695
##    43  0.6127354  0.4840071
##    45  0.6107358  0.4813931
##    47  0.6226364  0.4972543
##    49  0.6226364  0.4972128
```

```
##    51  0.6265774  0.5023565
##    53  0.6324985  0.5101949
##    55  0.6305183  0.5075913
##    57  0.6265774  0.5022678
##    59  0.6265774  0.5022678
##    61  0.6226558  0.4970314
##    63  0.6147350  0.4865330
##    65  0.6206756  0.4943671
##    67  0.6186954  0.4917340
##    69  0.6127742  0.4837849
##    71  0.6207144  0.4944119
##    73  0.6028926  0.4706313
##    75  0.6167152  0.4891844
##    77  0.6167152  0.4890748
##    79  0.6107940  0.4812121
##    81  0.6068336  0.4759575
##    83  0.6088332  0.4786485
##    85  0.6068725  0.4760426
##    87  0.6207533  0.4945205
##    89  0.6108523  0.4813724
##    91  0.6148321  0.4867382
##    93  0.6147932  0.4866638
##    95  0.6167929  0.4893118
##    97  0.6108328  0.4814609
##    99  0.6108134  0.4813764
##   101  0.6226752  0.4972482
##   103  0.6147544  0.4867342
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
# plot knn model
plot(knn)
```




```r
# best tuning parameter for k that minimizes MSE
print(paste0("best k to minimize MSE: ", knn$bestTune))
```

```
## [1] "best k to minimize MSE: 9"
```

Test the model

```r
set.seed(33)
# create test set
test <- sample(nrow(boston_df), 0.8*nrow(boston_df))

# test model with k = 9
knn_test <- knn(train = boston_df[-test, -13, drop = F],
                test = boston_df[test, -13, drop = F],
                cl = boston_df[-test, "medv_F"],
                k = 9)

# check the accuracy
table(knn_test, Real = boston_df[test, "medv_F"])
```

```
##          Real
## knn_test first second third fourth
##   first      72     22    10      7
##   second     28     45    26     22
##   third       2     18    50     32
##   fourth      2     11    11     46
```

```r
(72+45+50+46)/405
```

```
## [1] 0.5259259
```

logistic regression

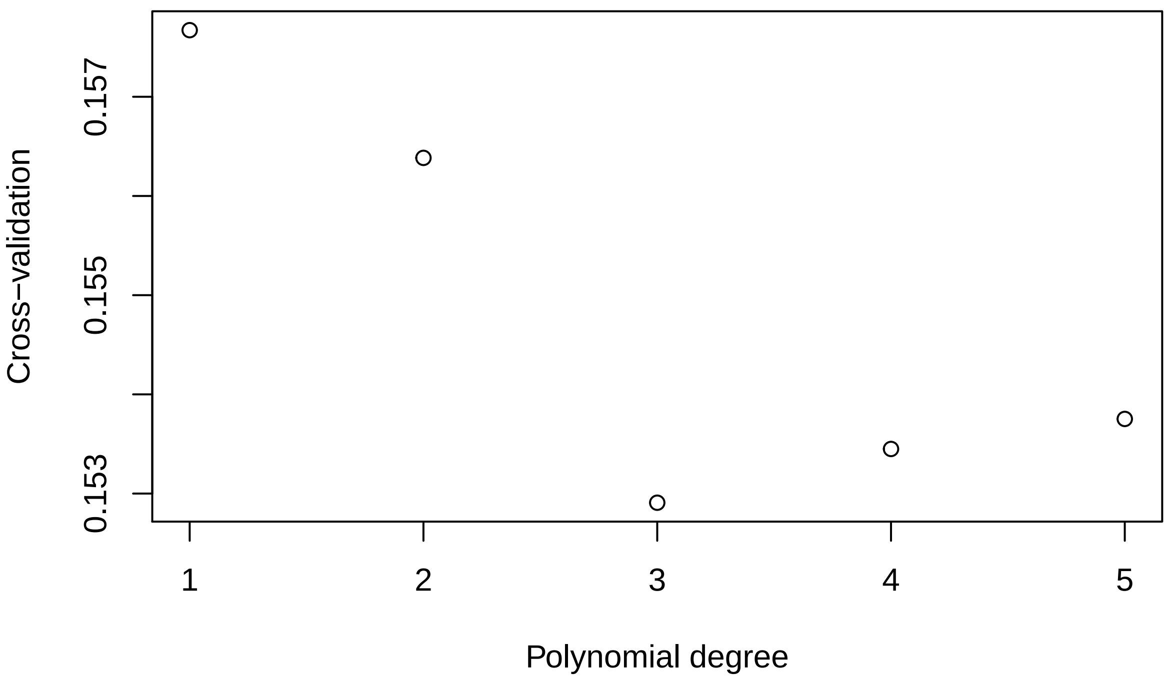
```
# need to make a binomial variable
mean(boston$medv)
```

```
## [1] 22.53281
```

```
boston_bi <- boston_sub %>% mutate(medv_bi = ifelse(medv >= 22.5, "high", "low")) %>%
  mutate(medv_bi = factor(medv_bi, levels = c("high", "low")))

boston_bi <- boston_bi[,-c(13:14)]

set.seed(3)

cv.err <- 1:5
# use i index to determine power of polynomial
# on each iteration, create the model for the given power and est MSE
for (i in 1:5){
  gl <- glm(medv_bi ~ poly(rm, i), family = "binomial",
            data = boston_bi)
  cv.err[i] <- cv.glm(boston_bi, gl)$delta[1]
}
cv.err
```

```
## [1] 0.1576716 0.1563841 0.1529078 0.1534497 0.1537524
```

```
plot(x = 1:5, y = cv.err,
     xlab = 'Polynomial degree', ylab='Cross-validation')
```




```
# use 3rd degree

glm <- glm(medv_bi ~ poly(rm, 3), data = boston_bi, family = "binomial")
summary(glm)
```

```
##
```

```
## Call:
## glm(formula = medv_bi ~ poly(rm, 3), family = "binomial", data = boston_bi)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5703  -0.6721   0.3852   0.7137   2.4747
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.3929     0.1198   3.279  0.00104 **
## poly(rm, 3)1 -34.8454     3.5521  -9.810  < 2e-16 ***
## poly(rm, 3)2  -5.0733     3.1524  -1.609  0.10754
## poly(rm, 3)3  18.5221     3.0243   6.124  9.1e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 688.12  on 505  degrees of freedom
## Residual deviance: 469.84  on 502  degrees of freedom
## AIC: 477.84
##
## Number of Fisher Scoring iterations: 5
```

```r
pred_glm <- predict(glm, type = "response") > 0.5
table(pred_glm, Real = boston_bi[, 13])
```

```
##         Real
## pred_glm high low
##    FALSE  146  41
##    TRUE    66 253
```

```r
# test
glm2  <- glm(medv_bi ~ rm, data = boston_bi[-test, ],
                  family = "binomial")
summary(glm2)
```

```
##
## Call:
## glm(formula = medv_bi ~ rm, family = "binomial", data = boston_bi[-test,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9520  -0.6719   0.4169   0.6798   1.6882
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  19.0723     3.8776   4.919 8.72e-07 ***
## rm           -2.9634     0.6137  -4.829 1.37e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 137.455  on 101  degrees of freedom
## Residual deviance:  93.185  on 100  degrees of freedom
## AIC: 97.185
##
## Number of Fisher Scoring iterations: 5
```

```r
pred_glm2 <- predict(glm2, type = "response", newdata = boston_bi[test, ]) > 0.5
table(pred_glm2, Real = boston_bi[test, 13])
```

```
##          Real
## pred_glm2 high low
##     FALSE  109  27
##     TRUE    62 206
```