Data Mining. Home work #9.

Andrii Rozumnyi

Task #1

- ML algorithms generalize the data from examples
- Learning = Representation + Evaluation + Optimization. Representation of input is in other words what features to use. Evaluation function (objective function or scoring function) is needed to distinguish good classifiers from bad ones. We need a proper optimization method for receiving the highest scoring.
- To make real generalization we need to devide data in training and testing set and check the results on test data.
- In ML there are often some assumptions used about the data. Every learner must embody some knowledge or assumptions beyond the data it is given in order to generalize beyond it. ML is not a magic; it cannot get something from nothing.
- There are many methods to combat the overfitting. Cross-validation can help to choose the best parameters for the model. Another method is to add regularization term to the evaluation function. Another optin is to use a statistical significance test.
- There is no universal technique which can always avoid overfiiting and underfitting at the same time.
- The next problem after overfitting is high dimensionality. As in most applications examples are not spread uniformly throughout the instance space, but are concentrated on or near a lower dimensional manifold, that is why it is possible effectively to reduce dimensionality.
- Feature enginnering is the key. Learning is easy if you have many independent features that each correlate well with the class. On the other hand, if the class is a very complex function of the features, you may not be able to learn it. Often, the raw data is not in a form that is appropriate for learning, but you can construct fetures from it.
- More data beasts a clever algorithm (while we are able to process it in reasonable time and simplier classifiers often can process data faster).
- Learn many models instead of just one. As the best learner usually varies from one application to another, that is why often it is more efficient to use several learners at the same time.
- Correlation doesn't imply causation. The main aim is to predict the effects, not just correlations between observable variables.

Task #2

```
## Warning: package 'flux' was built under R version 3.2.4
```

```
## Loading required package: caTools
```

```
## This is flux 0.3-0
```

Let's write a function for assigning true labels in our ordering list and call it. In such a way we will receive ordered column of true labels.

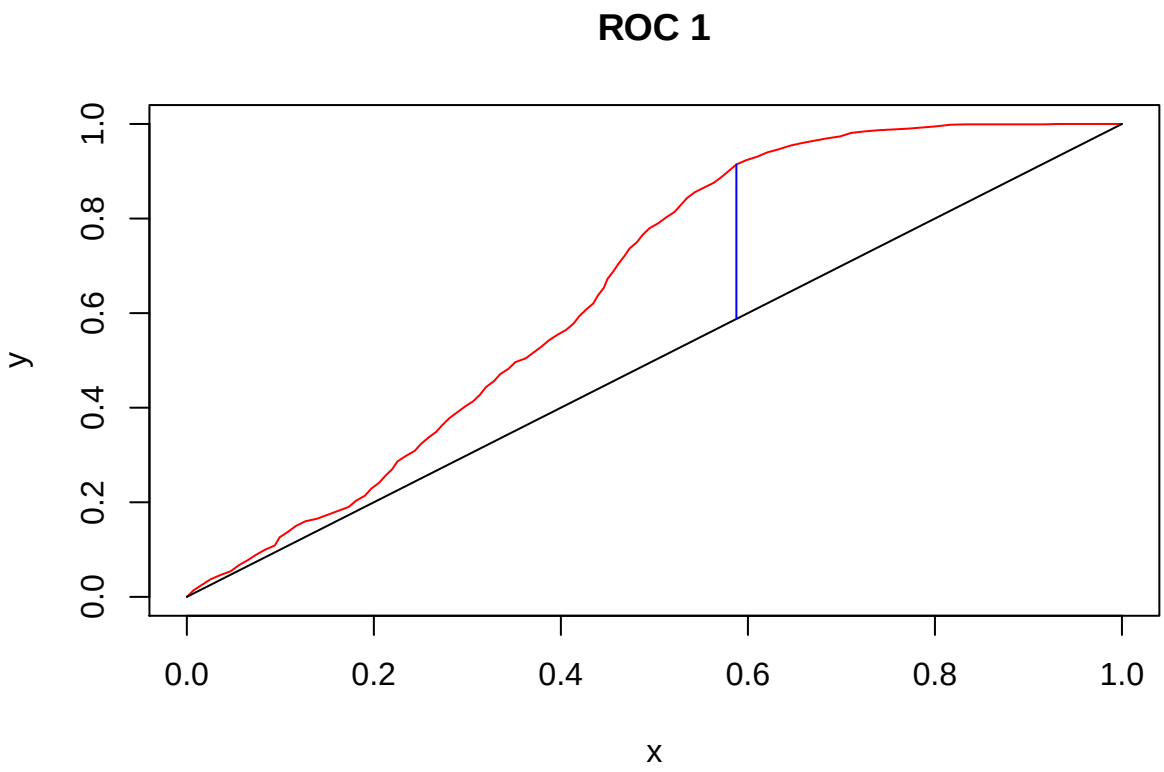
```
# assign true labels for ordering
add_true_class <- function(roc, true_labels){
    n <- nrow(roc)
    labeling <- c()
    for(i in 1:n){
        ind <- which(true_labels[, 1] == roc[i, ])
        labeling <- c(labeling, true_labels[ind, 2])
    }
    roc$class <- labeling
    return(roc)
}

roc1 <- add_true_class(roc1, data)
roc2 <- add_true_class(roc2, data)
roc3 <- add_true_class(roc3, data)
roc4 <- add_true_class(roc4, data)
```

Let's write a function which calculate TPR and FPR for each of possible k from 0 to 100 with step 1. Also, it calculates index J which will be used in the next exercise and output the best cutoff for the classifier. After that let's use built-in function for calculating AUC from $flux$ package:

```
ROC.analysis <- function(roc, msg="", step=1){
    k <- 0
    n <- nrow(roc)
    y <- c()
    x <- c()
    # TPR = TP/P = TP/(TP+FN)
    # FPR = FP/N = FP/(FP+TN)
    # SPC = TN/N
    P <- sum(roc[ ,2])
    N <- n - P
    SPC <- c()
    while(k <= 100){
        split <- round(n*k/100)
        TP <- sum(roc[0:split, 2])
        FP <- split - TP
        TN.value <- n - split - sum(roc[min(n, split+1):n, 2])
        TPR.value <- TP/P
        FPR.value <- FP/N
        y <- c(y, TPR.value)
        x <- c(x, FPR.value)
        SPC <- c(SPC, TN.value/N)
        k <- k + step
    }
    plot(x, y, col="red", pch=16, type="l", xlim=c(0, 1), ylim=c(0, 1),main=msg)
    points(seq(0, 1, 0.01), seq(0, 1, 0.01), type = "l")
    print(paste("AUC:", auc(x, y)))
    J <- y + SPC - 1
    best_J <- which.max(J)
    print(paste("Best cutoff: ", best_J))
    segments(x[best_J], x[best_J], x[best_J], y[best_J], col="blue")
}
```

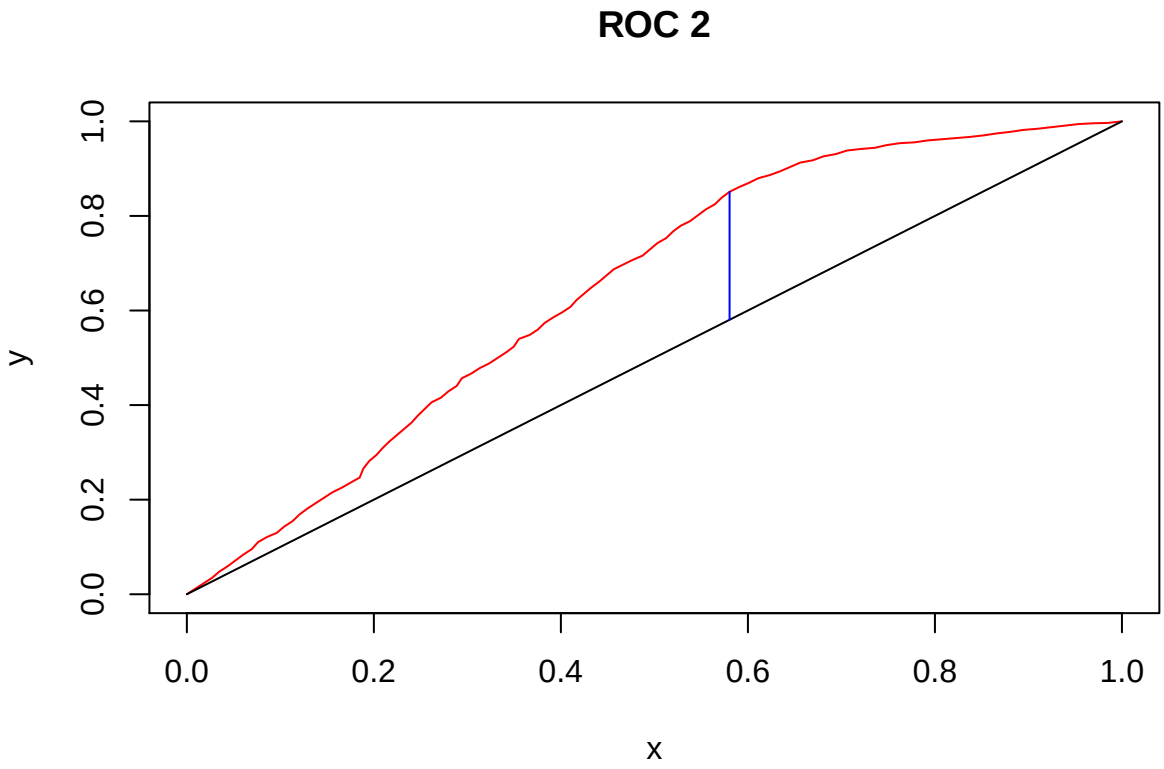
Let's draw ROC for each case and output AUC values:

```
ROC.analysis(roc1, "ROC 1")
```



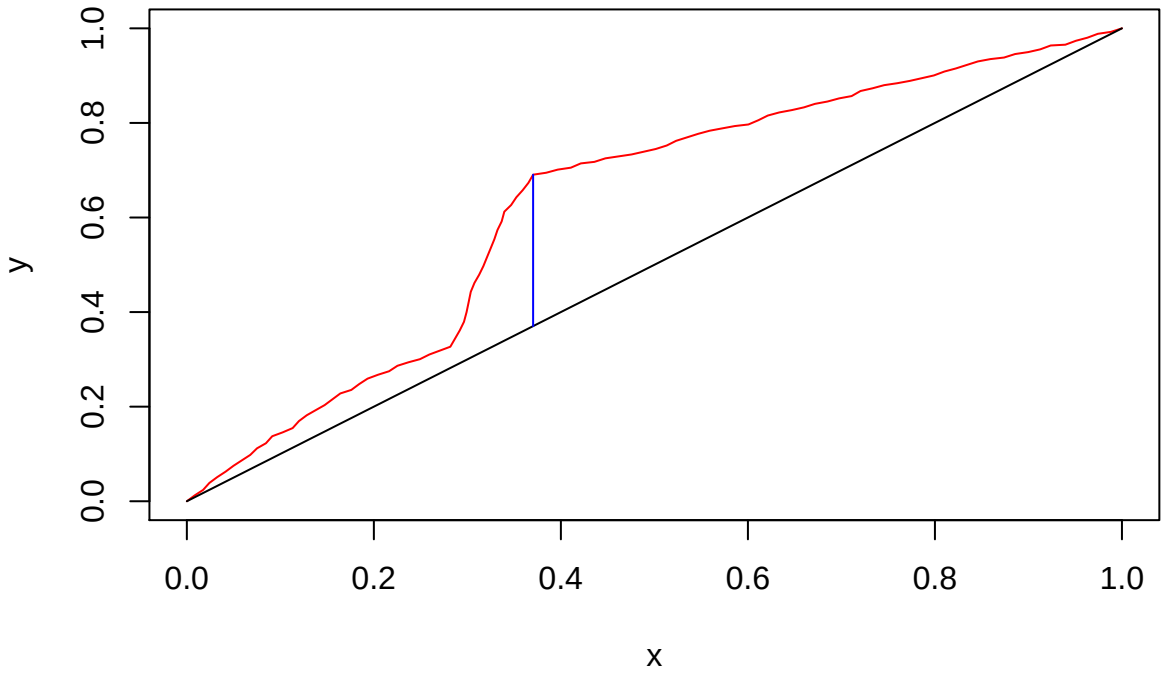

```
## [1] "AUC: 0.651516409032762"
## [1] "Best cutoff:  73"
```

```
ROC.analysis(roc2, "ROC 2")
```



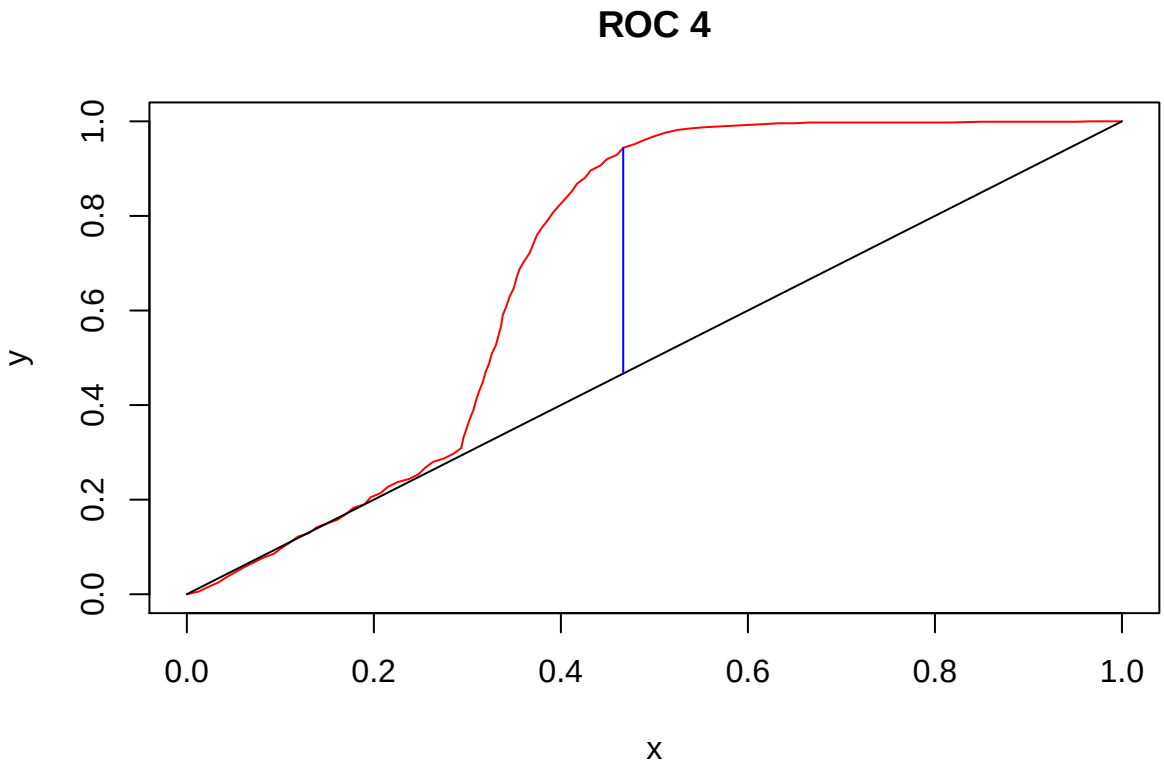

```
## [1] "AUC: 0.647698585607082"
## [1] "Best cutoff:  70"
```

```
ROC.analysis(roc3, "ROC 3")
```




```
## [1] "AUC: 0.63061520904658"
## [1] "Best cutoff:  51"
```

```
ROC.analysis(roc4, "ROC 4")
```




```
## [1] "AUC: 0.697440951689342"
## [1] "Best cutoff:  67"
```

Task #3

First two classifiers are quite similar to each other: they are quite smooth. While two last have some hill in the middle. AUC values are very similar but in the last case it is a little bit bigger.

Let's find the best cutoff using Youden's J statistic. The index is defined for all points of an ROC curve, and the maximum value of the index may be used as a criterion for selecting the optimum cut-off point. The index is represented graphically as the height above the chance line.

The formulas are:

$J = Sensitivity + Specificity - 1$

$Sensitivity = TPR$

$Specificity = \frac{TN}{N} = \frac{TN}{TN+FP}$

Please have a look at the results in previous subtask. There outputted the best cutoff for all classifiers.

Task #4

So, let's load the data and make linear regression on top of them. We use last column as a target and others as variables for regression. The object that return *lm* function has a field *residuals*. Let's use it for caclulating $RMSE$:

```
housing <- read.table("housing.data.txt")
model <- lm(V14 ~ ., housing)
sqrt(mean(model$residuals^2))
```

```
## [1] 4.679191
```

Task #5

Let's build a linear regression for each of the attribute and use others for prediction. In each case let's output summary about the model. In such a way we can find the most significant predictors in each case (those which have *** near the variable name). Also, let's output RMSE for each case as we have an access to residuals from the model.

Let's store RMSE and standart deviation for each case:

```
results <- matrix(NA, nrow=14, ncol=2)
colnames(results) <- c("RMSE", "sd")
rownames(results) <- seq(1, 14, 1)

model <- lm(V1 ~ ., housing)
results[1, 1] <- sqrt(mean(model$residuals^2))
results[1, 2] <- sd(housing$V1)
summary(model)
```

```
##
## Call:
## lm(formula = V1 ~ ., data = housing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## V2            0.044855   0.018734   2.394 0.017025 *
## V3           -0.063855   0.083407  -0.766 0.444294
## V4           -0.749134   1.180147  -0.635 0.525867
## V5          -10.313535   5.275536  -1.955 0.051152 .
## V6            0.430131   0.612830   0.702 0.483089
## V7            0.001452   0.017925   0.081 0.935488
## V8           -0.987176   0.281817  -3.503 0.000502 ***
## V9            0.588209   0.088049   6.680 6.46e-11 ***
## V10          -0.003780   0.005156  -0.733 0.463793
## V11          -0.271081   0.186450  -1.454 0.146611
## V12          -0.007538   0.003673  -2.052 0.040702 *
## V13           0.126211   0.075725   1.667 0.096208 .
## V14          -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V2 ~ ., housing)
results[2, 1] <- sqrt(mean(model$residuals^2))
results[2, 2] <- sd(housing$V2)
summary(model)
```

```
##
## Call:
## lm(formula = V2 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.390  -8.489  -1.270   5.711  62.001
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.339495  17.402356  -0.537 0.591731
## V1           0.256775   0.107244   2.394 0.017025 *
## V3          -0.475633   0.198524  -2.396 0.016955 *
## V4          -0.409075   2.824713  -0.145 0.884912
## V5          -1.038664  12.671082  -0.082 0.934703
## V6           2.771249   1.461661   1.896 0.058551 .
## V7          -0.113812   0.042580  -2.673 0.007769 **
## V8           6.439838   0.617814  10.424  < 2e-16 ***
## V9          -0.649564   0.218056  -2.979 0.003036 **
## V10          0.065790   0.011980   5.492 6.40e-08 ***
## V11         -2.432019   0.433404  -5.611 3.35e-08 ***
## V12         -0.002375   0.008826  -0.269 0.788008
## V13          0.448603   0.180561   2.484 0.013305 *
## V14          0.489312   0.144699   3.382 0.000778 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.41 on 492 degrees of freedom
## Multiple R-squared:  0.5749, Adjusted R-squared:  0.5636
## F-statistic: 51.18 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V3 ~ ., housing)
results[3, 1] <- sqrt(mean(model$residuals^2))
results[3, 2] <- sd(housing$V3)
summary(model)
```

```
##
## Call:
## lm(formula = V3 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8990 -2.0907 -0.2367  1.3749 15.9708
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.052135   3.923648  -1.288  0.19849
```

```
## V1          -0.018634   0.024340  -0.766  0.44429
## V2          -0.024246   0.010120  -2.396  0.01695 *
## V4           1.413125   0.634587   2.227  0.02641 *
## V5          16.588805   2.761413   6.007 3.67e-09 ***
## V6          -0.666711   0.329850  -2.021  0.04379 *
## V7           0.000229   0.009683   0.024  0.98114
## V8          -0.668223   0.151152  -4.421 1.21e-05 ***
## V9          -0.303652   0.047751  -6.359 4.64e-10 ***
## V10          0.026911   0.002509  10.728  < 2e-16 ***
## V11          0.286835   0.100105   2.865  0.00434 **
## V12         -0.001555   0.001992  -0.781  0.43522
## V13          0.068617   0.040905   1.677  0.09409 .
## V14          0.011047   0.033044   0.334  0.73829
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.478 on 492 degrees of freedom
## Multiple R-squared:  0.7495, Adjusted R-squared:  0.7429
## F-statistic: 113.3 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V4 ~ ., housing)
results[4, 1] <- sqrt(mean(model$residuals^2))
results[4, 2] <- sd(housing$V4)
summary(model)
```

```
##
## Call:
## lm(formula = V4 ~ ., data = housing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33259 -0.08695 -0.04420 -0.01023  0.95204
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.052e-01  2.778e-01  -0.379  0.70515
## V1          -1.092e-03  1.721e-03  -0.635  0.52587
## V2          -1.042e-04  7.195e-04  -0.145  0.88491
## V3           7.061e-03  3.171e-03   2.227  0.02641 *
## V5           2.740e-01  2.019e-01   1.358  0.17524
## V6          -1.266e-02  2.341e-02  -0.541  0.58896
## V7           7.584e-04  6.836e-04   1.109  0.26783
## V8           7.065e-03  1.089e-02   0.649  0.51680
## V9           5.917e-03  3.501e-03   1.690  0.09165 .
## V10         -4.298e-04  1.960e-04  -2.192  0.02882 *
## V11         -7.073e-03  7.128e-03  -0.992  0.32152
## V12          8.969e-05  1.408e-04   0.637  0.52447
## V13          3.550e-04  2.900e-03   0.122  0.90262
## V14          7.214e-03  2.313e-03   3.118  0.00193 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2459 on 492 degrees of freedom
## Multiple R-squared:  0.08694,    Adjusted R-squared:  0.06282
```

```
## F-statistic: 3.604 on 13 and 492 DF,  p-value: 1.904e-05
```

```
model <- lm(V5 ~ ., housing)
results[5, 1] <- sqrt(mean(model$residuals^2))
results[5, 2] <- sd(housing$V5)
summary(model)
```

```
##
## Call:
## lm(formula = V5 ~ ., data = housing)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.120257 -0.034766 -0.003939  0.031042  0.182760
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.851e-01  5.082e-02  15.448  < 2e-16 ***
## V1          -7.474e-04  3.823e-04  -1.955   0.0512 .
## V2          -1.315e-05  1.604e-04  -0.082   0.9347
## V3           4.120e-03  6.857e-04   6.007 3.67e-09 ***
## V4           1.362e-02  1.003e-02   1.358   0.1752
## V6          -7.633e-04  5.219e-03  -0.146   0.8838
## V7           8.800e-04  1.474e-04   5.972 4.50e-09 ***
## V8          -1.760e-02  2.296e-03  -7.667 9.48e-14 ***
## V9           3.091e-03  7.703e-04   4.012 6.95e-05 ***
## V10          4.110e-05  4.387e-05   0.937   0.3493
## V11         -1.307e-02  1.477e-03  -8.848  < 2e-16 ***
## V12         -2.835e-05  3.138e-05  -0.904   0.3667
## V13         -3.695e-04  6.462e-04  -0.572   0.5678
## V14         -2.371e-03  5.097e-04  -4.651 4.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05482 on 492 degrees of freedom
## Multiple R-squared:  0.782,  Adjusted R-squared:  0.7762
## F-statistic: 135.8 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V6 ~ ., housing)
results[6, 1] <- sqrt(mean(model$residuals^2))
results[6, 2] <- sd(housing$V6)
summary(model)
```

```
##
## Call:
## lm(formula = V6 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2000 -0.1995  0.0040  0.2289  2.2429
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  6.0289510  0.4607699  13.085  < 2e-16 ***
## V1           0.0023255  0.0033133   0.702 0.483089
## V2           0.0026173  0.0013805   1.896 0.058551 .
## V3          -0.0123523  0.0061112  -2.021 0.043795 *
## V4          -0.0469248  0.0867850  -0.541 0.588958
## V5          -0.0569500  0.3894013  -0.146 0.883784
## V7           0.0055650  0.0012939   4.301 2.05e-05 ***
## V8           0.0014353  0.0209785   0.068 0.945481
## V9           0.0098605  0.0067468   1.461 0.144517
## V10         -0.0001040  0.0003793  -0.274 0.784084
## V11         -0.0065718  0.0137358  -0.478 0.632548
## V12         -0.0009290  0.0002680  -3.467 0.000573 ***
## V13         -0.0356301  0.0053476  -6.663 7.22e-11 ***
## V14          0.0379285  0.0041606   9.116  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4735 on 492 degrees of freedom
## Multiple R-squared:  0.5576, Adjusted R-squared:  0.5459
## F-statistic:  47.7 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V7 ~ ., housing)
results[7, 1] <- sqrt(mean(model$residuals^2))
results[7, 2] <- sd(housing$V7)
summary(model)
```

```
##
## Call:
## lm(formula = V7 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.343  -9.817   0.978  10.307  46.911
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -32.886872  18.238427  -1.803  0.07197 .
## V1            0.009183   0.113388   0.081  0.93549
## V2           -0.125763   0.047051  -2.673  0.00777 **
## V3            0.004964   0.209901   0.024  0.98114
## V4            3.289931   2.965675   1.109  0.26783
## V5           76.810662  12.861831   5.972 4.50e-09 ***
## V6            6.510990   1.513895   4.301 2.05e-05 ***
## V8           -4.386519   0.689789  -6.359 4.64e-10 ***
## V9           -0.383179   0.230631  -1.661  0.09726 .
## V10           0.008625   0.012968   0.665  0.50629
## V11           0.782077   0.468620   1.669  0.09577 .
## V12           0.012883   0.009260   1.391  0.16478
## V13           1.313825   0.181574   7.236 1.79e-12 ***
## V14           0.008063   0.153864   0.052  0.95823
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.2 on 492 degrees of freedom
```

```
## Multiple R-squared:  0.6775, Adjusted R-squared:  0.669
## F-statistic: 79.51 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V8 ~ ., housing)
results[8, 1] <- sqrt(mean(model$residuals^2))
results[8, 2] <- sd(housing$V8)
summary(model)
```

```
##
## Call:
## lm(formula = V8 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5462 -0.6364 -0.0463  0.5347  4.3103
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.6670771  1.0442339  10.215  < 2e-16 ***
## V1          -0.0246488  0.0070367  -3.503 0.000502 ***
## V2           0.0280890  0.0026948  10.424  < 2e-16 ***
## V3          -0.0571758  0.0129331  -4.421 1.21e-05 ***
## V4           0.1209801  0.1864785   0.649 0.516795
## V5          -6.0641064  0.7909332  -7.667 9.48e-14 ***
## V6           0.0066286  0.0968849   0.068 0.945481
## V7          -0.0173148  0.0027228  -6.359 4.64e-10 ***
## V9           0.0160172  0.0145126   1.104 0.270271
## V10         -0.0005065  0.0008148  -0.622 0.534469
## V11         -0.0093794  0.0295223  -0.318 0.750843
## V12          0.0003656  0.0005827   0.627 0.530708
## V13         -0.0268033  0.0119385  -2.245 0.025203 *
## V14         -0.0678417  0.0091703  -7.398 6.01e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 492 degrees of freedom
## Multiple R-squared:  0.7725, Adjusted R-squared:  0.7665
## F-statistic: 128.5 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V9 ~ ., housing)
results[9, 1] <- sqrt(mean(model$residuals^2))
results[9, 2] <- sd(housing$V9)
summary(model)
```

```
##
## Call:
## lm(formula = V9 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.932  -1.909   0.561   2.209   6.518
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -25.923363   3.370100  -7.692 7.95e-14 ***
## V1            0.141386   0.021164   6.680 6.46e-11 ***
## V2           -0.027275   0.009156  -2.979  0.00304 **
## V3           -0.250115   0.039332  -6.359 4.64e-10 ***
## V4            0.975423   0.577157   1.690  0.09165 .
## V5           10.251666   2.555010   4.012 6.95e-05 ***
## V6            0.438382   0.299954   1.461  0.14452
## V7           -0.014560   0.008764  -1.661  0.09726 .
## V8            0.154192   0.139707   1.104  0.27027
## V10           0.044481   0.001541  28.867  < 2e-16 ***
## V11           0.486140   0.088947   5.465 7.35e-08 ***
## V12          -0.004070   0.001799  -2.262  0.02415 *
## V13           0.097205   0.036972   2.629  0.00883 **
## V14           0.135457   0.029365   4.613 5.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.157 on 492 degrees of freedom
## Multiple R-squared:  0.8719, Adjusted R-squared:  0.8685
## F-statistic: 257.7 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V10 ~ ., housing)
results[10, 1] <- sqrt(mean(model$residuals^2))
results[10, 2] <- sd(housing$V10)
summary(model)
```

```
##
## Call:
## lm(formula = V10 ~ ., data = housing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.224  -20.809   -5.443   14.671  259.844
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 208.50779   62.88822   3.316 0.000982 ***
## V1           -0.28873    0.39381  -0.733 0.463793
## V2            0.87786    0.15986   5.492  6.4e-08 ***
## V3            7.04413    0.65663  10.728  < 2e-16 ***
## V4          -22.51192   10.26845  -2.192 0.028823 *
## V5           43.32664   46.24471   0.937 0.349270
## V6           -1.46898    5.35830  -0.274 0.784084
## V7            0.10415    0.15659   0.665 0.506289
## V8           -1.54947    2.49258  -0.622 0.534469
## V9           14.13533    0.48968  28.867  < 2e-16 ***
## V11           0.91618    1.63252   0.561 0.574912
## V12          -0.00247    0.03224  -0.077 0.938957
## V13          -1.11261    0.66179  -1.681 0.093357 .
## V14          -1.73487    0.52892  -3.280 0.001112 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 56.28 on 492 degrees of freedom
## Multiple R-squared:  0.8914, Adjusted R-squared:  0.8885
## F-statistic: 310.5 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V11 ~ ., housing)
results[11, 1] <- sqrt(mean(model$residuals^2))
results[11, 2] <- sd(housing$V11)
summary(model)
```

```
##
## Call:
## lm(formula = V11 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1190 -1.0126 -0.0060  0.8961  4.8945
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.484e+01  1.352e+00  18.379  < 2e-16 ***
## V1          -1.578e-02  1.085e-02  -1.454  0.14661
## V2          -2.473e-02  4.408e-03  -5.611 3.35e-08 ***
## V3           5.722e-02  1.997e-02   2.865  0.00434 **
## V4          -2.824e-01  2.846e-01  -0.992  0.32152
## V5          -1.050e+01  1.187e+00  -8.848  < 2e-16 ***
## V6          -7.076e-02  1.479e-01  -0.478  0.63255
## V7           7.198e-03  4.313e-03   1.669  0.09577 .
## V8          -2.187e-02  6.883e-02  -0.318  0.75084
## V9           1.177e-01  2.154e-02   5.465 7.35e-08 ***
## V10          6.983e-04  1.244e-03   0.561  0.57491
## V12          1.573e-03  8.873e-04   1.773  0.07692 .
## V13         -3.770e-02  1.824e-02  -2.067  0.03929 *
## V14         -1.021e-01  1.402e-02  -7.283 1.31e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.554 on 492 degrees of freedom
## Multiple R-squared:  0.4982, Adjusted R-squared:  0.485
## F-statistic: 37.58 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V12 ~ ., housing)
results[12, 1] <- sqrt(mean(model$residuals^2))
results[12, 2] <- sd(housing$V12)
summary(model)
```

```
##
## Call:
## lm(formula = V12 ~ ., data = housing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -307.973   -7.861    7.139   23.176  232.954
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 447.32955   86.59737   5.166 3.49e-07 ***
## V1           -1.12576    0.54862  -2.052 0.040702 *
## V2           -0.06195    0.23026  -0.269 0.788008
## V3           -0.79600    1.01930  -0.781 0.435221
## V4            9.18610   14.42258   0.637 0.524470
## V5          -58.43343   64.66882  -0.904 0.366661
## V6          -25.66414    7.40330  -3.467 0.000573 ***
## V7            0.30418    0.21863   1.391 0.164780
## V8            2.18663    3.48539   0.627 0.530708
## V9           -2.52865    1.11800  -2.262 0.024148 *
## V10          -0.00483    0.06304  -0.077 0.938957
## V11           4.03476    2.27625   1.773 0.076923 .
## V13          -1.51395    0.92553  -1.636 0.102528
## V14           2.56083    0.73867   3.467 0.000573 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 78.69 on 492 degrees of freedom
## Multiple R-squared:  0.2761, Adjusted R-squared:  0.257
## F-statistic: 14.44 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V13 ~ ., housing)
results[13, 1] <- sqrt(mean(model$residuals^2))
results[13, 2] <- sd(housing$V13)
summary(model)
```

```
##
## Call:
## lm(formula = V13 ~ ., data = housing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.4344  -2.2498  -0.5345   1.9152  20.0091
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.155876   3.981340   9.333  < 2e-16 ***
## V1           0.044485   0.026690   1.667  0.09621 .
## V2           0.027621   0.011117   2.484  0.01330 *
## V3           0.082877   0.049406   1.677  0.09409 .
## V4           0.085799   0.700912   0.122  0.90262
## V5          -1.797038   3.143107  -0.572  0.56776
## V6          -2.322790   0.348622  -6.663 7.22e-11 ***
## V7           0.073206   0.010117   7.236 1.79e-12 ***
## V8          -0.378356   0.168523  -2.245  0.02520 *
## V9           0.142536   0.054213   2.629  0.00883 **
## V10         -0.005134   0.003054  -1.681  0.09336 .
## V11         -0.228266   0.110452  -2.067  0.03929 *
## V12         -0.003573   0.002184  -1.636  0.10253
## V14         -0.340572   0.032915 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.823 on 492 degrees of freedom
## Multiple R-squared:  0.7208, Adjusted R-squared:  0.7134
## F-statistic:  97.7 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
model <- lm(V14 ~ ., housing)
results[14, 1] <- sqrt(mean(model$residuals^2))
results[14, 2] <- sd(housing$V14)
summary(model)
```

```
##
## Call:
## lm(formula = V14 ~ ., data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## V1          -1.080e-01  3.286e-02  -3.287 0.001087 **
## V2           4.642e-02  1.373e-02   3.382 0.000778 ***
## V3           2.056e-02  6.150e-02   0.334 0.738288
## V4           2.687e+00  8.616e-01   3.118 0.001925 **
## V5          -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## V6           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## V7           6.922e-04  1.321e-02   0.052 0.958229
## V8          -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## V9           3.060e-01  6.635e-02   4.613 5.07e-06 ***
## V10         -1.233e-02  3.760e-03  -3.280 0.001112 **
## V11         -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## V12          9.312e-03  2.686e-03   3.467 0.000573 ***
## V13         -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

I think that variables with lower standart deviation (variance) "easier" to predict. Let's sort matrix by RMSE score and, indeed, we see that standart deviation also are almost sorted. Thus, it might be true.

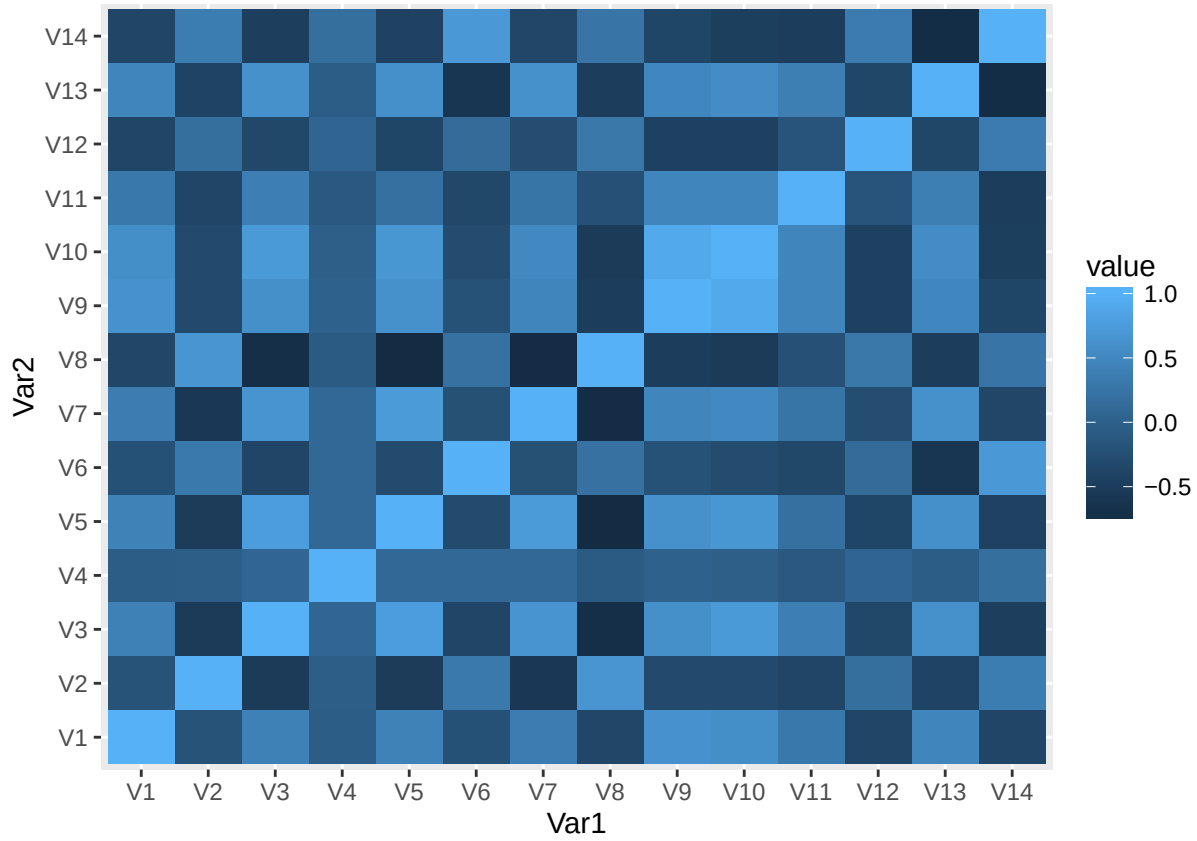
```
results[order(results[, 1]), ]
```

```
##           RMSE          sd
## 5   0.05405173   0.1158777
## 4   0.24246148   0.2539940
## 6   0.46687325   0.7026171
## 8   1.00332080   2.1057101
## 11  1.53201158   2.1649455
## 9   3.11298116   8.7072594
```

```
## 3   3.43000107   6.8603529
## 13  3.76960275   7.1410615
## 14  4.67919130   9.1971041
## 1   6.34949362   8.6015451
## 2  15.19178610  23.3224530
## 7  15.96950251  28.1488614
## 10 55.49339623 168.5371161
## 12 77.59739768  91.2948644
```

Let's calculate matrix of correlation which represents pairwise correlations between the variables and draw correlation heatmap in order easily to identify the most correlated (those which have lighter color):

```
library(ggplot2)
library(reshape2)
ggplot(data = melt(cor(housing)), aes(x=Var1, y=Var2, fill=value)) +
    geom_tile()
```




```
cor(housing)
```

```
##             V1          V2          V3           V4          V5
## V1   1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## V2  -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## V3   0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## V4  -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## V5   0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
```

```
## V6  -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## V7   0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## V8  -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## V9   0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## V10  0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## V11  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## V12 -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## V13  0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## V14 -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
##              V6          V7          V8           V9         V10
## V1  -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431
## V2   0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332
## V3  -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018
## V4   0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652
## V5  -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320
## V6   1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783
## V7  -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559
## V8   0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158
## V9  -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819
## V10 -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000
## V11 -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304
## V12  0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801
## V13 -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341
## V14  0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593
##             V11         V12        V13        V14
## V1   0.2899456 -0.38506394  0.4556215 -0.3883046
## V2  -0.3916785  0.17552032 -0.4129946  0.3604453
## V3   0.3832476 -0.35697654  0.6037997 -0.4837252
## V4  -0.1215152  0.04878848 -0.0539293  0.1752602
## V5   0.1889327 -0.38005064  0.5908789 -0.4273208
## V6  -0.3555015  0.12806864 -0.6138083  0.6953599
## V7   0.2615150 -0.27353398  0.6023385 -0.3769546
## V8  -0.2324705  0.29151167 -0.4969958  0.2499287
## V9   0.4647412 -0.44441282  0.4886763 -0.3816262
## V10  0.4608530 -0.44180801  0.5439934 -0.4685359
## V11  1.0000000 -0.17738330  0.3740443 -0.5077867
## V12 -0.1773833  1.00000000 -0.3660869  0.3334608
## V13  0.3740443 -0.36608690  1.0000000 -0.7376627
## V14 -0.5077867  0.33346082 -0.7376627  1.0000000
```

Task #6

Let's write a straighforward function which calculates costs for each k. In such a way we can find the min cost and corresponding cutoff which we output at the end of the function.

```
ROC.cost <- function(roc, FN_cost=20, FP_cost=15, msg="", step=1){
    k <- 0
    n <- nrow(roc)
    min_cost <- +Inf
    best_cutoff <- k
    while(k <= 100){
        split <- round(n*k/100)
        FP <- split - sum(roc[0:split, 2])
        FN <- sum(roc[min(n, split+1):n, 2])
```

```
        cost <- FP*FP_cost + FN*FN_cost
        if(cost < min_cost){
            min_cost <- cost
            best_cutoff <- k
            FP_cutoff <- FP
            FN_cutoff <- FN
        }
        k <- k + step
    }
    print(paste("Best cutoff:", best_cutoff, "with min cost:", min_cost))
    print(paste("FP:", FP_cutoff, "FN:", FN_cutoff))
}
```

Let's call the function for each of the ROC with cost 20 for FN and 15 for FP.

```
ROC.cost(roc1, msg="ROC 1")
```

```
## [1] "Best cutoff: 72 with min cost: 17815"
## [1] "FP: 1049 FN: 104"
```

```
ROC.cost(roc2, msg="ROC 1")
```

```
## [1] "Best cutoff: 69 with min cost: 19160"
## [1] "FP: 1036 FN: 181"
```

```
ROC.cost(roc3, msg="ROC 1")
```

```
## [1] "Best cutoff: 50 with min cost: 17435"
## [1] "FP: 661 FN: 376"
```

```
ROC.cost(roc4, msg="ROC 1")
```

```
## [1] "Best cutoff: 66 with min cost: 13855"
## [1] "FP: 833 FN: 68"
```

Let's provide some examples with biased costs:

```
ROC.cost(roc1, FN_cost=100, FP_cost=10)
```

```
## [1] "Best cutoff: 89 with min cost: 14770"
## [1] "FP: 1457 FN: 2"
```

```
ROC.cost(roc2, FN_cost=100, FP_cost=10)
```

```
## [1] "Best cutoff: 97 with min cost: 17720"
## [1] "FP: 1702 FN: 7"
```

```
ROC.cost(roc3, FN_cost=100, FP_cost=10)
```

```
## [1] "Best cutoff: 100 with min cost: 17850"
## [1] "FP: 1785 FN: 0"
```

```
ROC.cost(roc4, FN_cost=100, FP_cost=10)
```

```
## [1] "Best cutoff: 73 with min cost: 11400"
## [1] "FP: 990 FN: 15"
```

We see that because of significantly biased cost, we received biased results.