

```
title: "Boosting"
author: "Ефременко Влада"
date: "2023-12-06"
```

```
```{r library, eval=TRUE}

library(adabag)
library(mlbench)
library(rpart)
library(dplyr)
library(pryr)
library(igraph)
library(kknn)
```
```

Задание 1

Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма `adaboost.M1` на наборе данных `Vehicle` из пакета `mlbench` (обучающая выборка должна состоять из 7/10 всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, ..., 301, объясните полученные результаты.

```
```{r get_data_task1, eval=TRUE}
data(Vehicle)
head(Vehicle)
```
```

Description: df [6 × 19]

| | Comp
<dbl> | Circ
<dbl> | D.Circ
<dbl> | Rad.Ra
<dbl> | Pr.Axis.Ra
<dbl> | Max.L.Ra
<dbl> | Scat.Ra
<dbl> | Elong
<dbl> | Pr.Axis.Rect
<dbl> |
|---|---------------|---------------|-----------------|-----------------|---------------------|-------------------|------------------|----------------|-----------------------|
| 1 | 95 | 48 | 83 | 178 | 72 | 10 | 162 | 42 | 20 |
| 2 | 91 | 41 | 84 | 141 | 57 | 9 | 149 | 45 | 19 |
| 3 | 104 | 50 | 106 | 209 | 66 | 10 | 207 | 32 | 23 |
| 4 | 93 | 41 | 82 | 159 | 63 | 9 | 144 | 46 | 19 |
| 5 | 85 | 44 | 70 | 205 | 103 | 52 | 149 | 45 | 19 |
| 6 | 107 | 57 | 106 | 172 | 50 | 6 | 255 | 26 | 28 |

6 rows | 1–10 of 19 columns

```
```{r split_data_task1, eval=TRUE}
n <- dim(Vehicle)[1]

data_rand1 <- Vehicle[order(runif(n)),]
df_train1 <- data_rand1[1:as.integer(n*0.7),]
df_test1 <- data_rand1[(as.integer(n*0.7)+1):n,]
```
```

```

```{r get_errors_task1, eval=TRUE}
mfinal <- seq(1, 301, by = 10)
maxdepth <- 5

error <- c()
for (i in (1:length(mfinal))) {
 Vehicle.adaboost <- boosting(Class~., data=df_train1, mfinal=mfinal[i],
 maxdepth=maxdepth)
 Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost, df_test1)

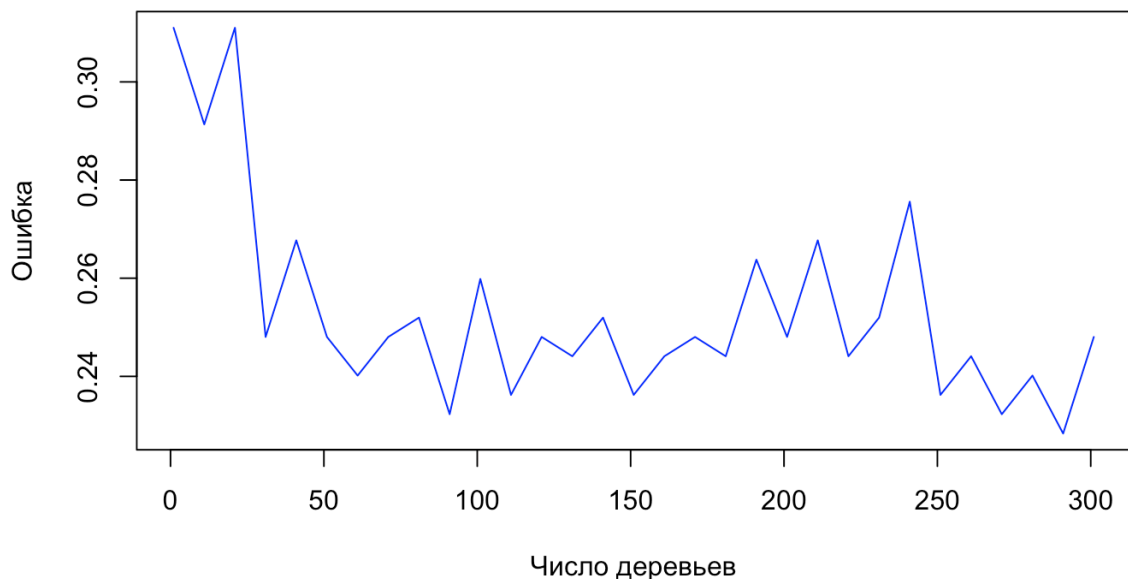
 error <- append(error, Vehicle.adaboost.pred$error)
}
```

```

```

```{r show_graph_task1, eval=TRUE}
plot(mfinal, error, type='l', xlab = "Число деревьев",
 ylab = "Ошибка", col = "blue")
```

```



Выше представлен график зависимости тестовой ошибки от числа деревьев. Из него видим, что наименьшее значение ошибки, равное 0.2283465, наблюдается при 291 дереве, в то время как при 1 и 21 дереве видим самую большую ошибку = 0.3110236.

Задание 2

Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма bagging на наборе данных Glass из пакета mlbench (обучающая выборка должна состоять из 7/10 всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, ..., 201, объясните полученные результаты.

```
```{r get_data_task2, eval=TRUE}
data(Glass)
head(Glass)
```
```

Description: df [6 × 10]

| | RI
<dbl> | Na
<dbl> | Mg
<dbl> | Al
<dbl> | Si
<dbl> | K
<dbl> | Ca
<dbl> | Ba
<dbl> | Fe
<dbl> |
|---|-------------|-------------|-------------|-------------|-------------|------------|-------------|-------------|-------------|
| 1 | 1.52101 | 13.64 | 4.49 | 1.10 | 71.78 | 0.06 | 8.75 | 0 | 0.00 |
| 2 | 1.51761 | 13.89 | 3.60 | 1.36 | 72.73 | 0.48 | 7.83 | 0 | 0.00 |
| 3 | 1.51618 | 13.53 | 3.55 | 1.54 | 72.99 | 0.39 | 7.78 | 0 | 0.00 |
| 4 | 1.51766 | 13.21 | 3.69 | 1.29 | 72.61 | 0.57 | 8.22 | 0 | 0.00 |
| 5 | 1.51742 | 13.27 | 3.62 | 1.24 | 73.08 | 0.55 | 8.07 | 0 | 0.00 |
| 6 | 1.51596 | 12.79 | 3.61 | 1.62 | 72.97 | 0.64 | 8.07 | 0 | 0.26 |

6 rows | 1-10 of 10 columns

```
```{r split_data_task2, eval=TRUE}
n <- dim(Glass)[1]

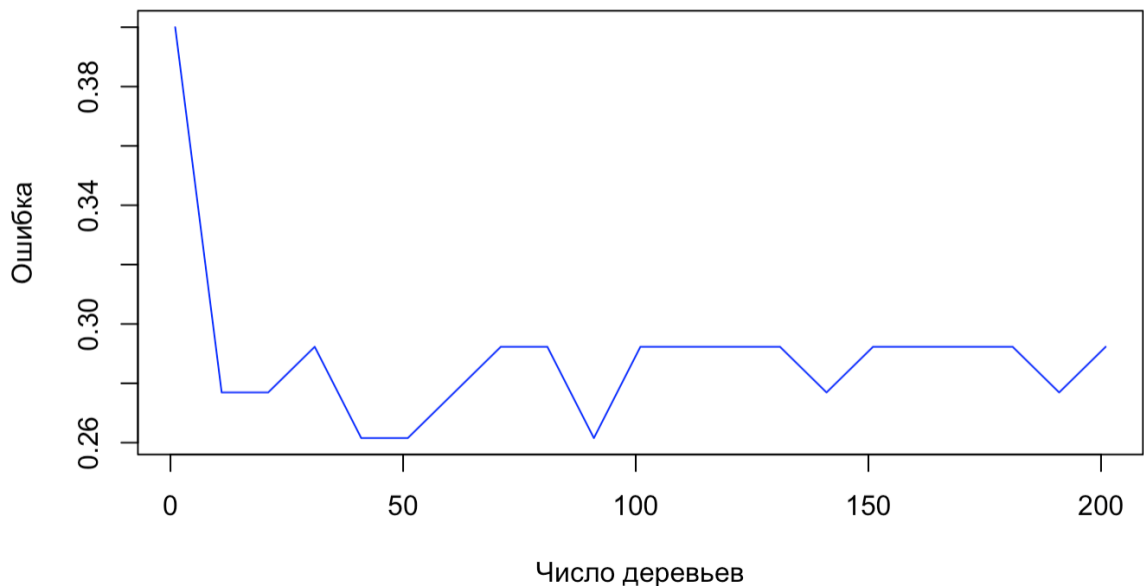
data_rand2 <- Glass[order(runif(n)),]
df_train2 <- data_rand2[1:as.integer(n*0.7),]
df_test2 <- data_rand2[(as.integer(n*0.7)+1):n,]
```
```

```
```{r get_errors_task2, eval=TRUE}
mfinal <- seq(1, 201, by = 10)
maxdepth <- 5

error <- c()
for (i in (1:length(mfinal))) {
 Glass.bagging <- bagging(Type~., data=df_train2, mfinal=mfinal[i],
 maxdepth=maxdepth)
 Glass.bagging.pred <- predict.bagging(Glass.bagging, df_test2)

 error <- append(error, Glass.bagging.pred$error)
}
```
```

```
```{r show_graph_task2, eval=TRUE}
plot(mfinal, error, type='l', xlab = "Число деревьев",
 ylab = "Ошибка", col = "blue")
```
```



Выше представлен график зависимости тестовой ошибки от числа деревьев. Из него видим, что наименьшее значение ошибки, равное 0.2615385, наблюдается при 41, 51 и 91 дереве, в то время как при 1 дереве видим самую большую ошибку = 0.4.

Задание 3

Реализуйте бустинг алгоритм с классификатором K ближайших соседей. Сравните тестовую ошибку, полученную с использованием данного классификатора на наборах данных Vehicle и Glass, с тестовой ошибкой, полученной с использованием единичного дерева классификации.

```
```{r get_errors_with_one_tree, eval=TRUE}
maxdepth <- 5

Vehicle.rpart <- rpart(Class~., data=df_train1, maxdepth=maxdepth)
Glass.rpart <- rpart(Type~., data=df_train2, maxdepth=maxdepth)

Vehicle.rpart.pred <- predict(Vehicle.rpart,newdata=df_test1,type="class")
Glass.rpart.pred <- predict(Glass.rpart,newdata=df_test2,type="class")

tb1 <- table(Vehicle.rpart.pred,df_test1$Class)
error1.rpart <- 1-(sum(diag(tb1))/sum(tb1))

tb2 <- table(Glass.rpart.pred,df_test2$Type)
error2.rpart <- 1-(sum(diag(tb2))/sum(tb2))

cat("Тестовая ошибка (с использованием единичного дерева классификации), Vehicle: ", error1.rpart, "\n")
cat("Тестовая ошибка (с использованием единичного дерева классификации), Glass: ", error2.rpart)
```
```

Тестовая ошибка (с использованием единичного дерева классификации), Vehicle: 0.2874016

Тестовая ошибка (с использованием единичного дерева классификации), Glass: 0.3846154

```

```{r knn_method, eval=TRUE}
calculate_weighted_frequencies <- function(train, trainlabels, n_number, w) {
 myfreq <- data.frame(names = levels(trainlabels), freq = rep(0, length(levels(trainlabels))))

 for (t in n_number) {
 myfreq[myfreq$names == trainlabels[t],][2] <- myfreq[myfreq$names == trainlabels[t],][2]
+ w[t]
 }

 return(myfreq)
}

knn_predict <- function(clfier, testdata) {
 n = nrow(testdata)
 pred = rep(NA_character_, n)
 trainlabels = clfier$train[, clfier$target]
 train <- clfier$train[, !(names(clfier$train) %in% clfier$target)]
 test <- testdata[, !(names(testdata) %in% clfier$target)]

 for (i in 1:n) {
 n_number <- order(apply(train, 1, function(x) sum((test[i,] - x)^2)))[1:clfier$k]
 myfreq <- calculate_weighted_frequencies(train, trainlabels, n_number, clfier$w)
 most_frequent <- clfier$levels[myfreq$freq == max(myfreq$freq)]
 pred[i] <- sample(most_frequent, 1)
 }

 factor(pred, levels = levels(trainlabels))
}
```

```

```

```{r boosting_algorithm, eval=TRUE}
knn_boosting <- function(target, data, k = 7, mfinal = 1) {

 n <- nrow(data)
 w <- rep(1/n, each = n)
 classifiers <- list()
 alphas <- vector()

 for (t in 1:mfinal) {
 clfier <- list(target = target, train = data, levels = levels(data[, target]), k = k, w = w)
 knn_predicted <- knn_predict(clfier, data)
 error <- w[data[, target] != knn_predicted]

 if (sum(error) >= 0.5) {
 break()
 }

 classifiers[[t]] <- clfier
 alphas[[t]] <- log((1 - sum(error)) / sum(error)) / 2

 for (i in 1:n) {
 if (knn_predicted[i] != data[, target][i]) {
 w[i] <- w[i]*exp(alphas[[t]])
 } else{
 w[i] <- w[i]*exp(-alphas[[t]])
 }
 }
 }

 result <- list()
 result$classifiers <- classifiers
 result$alphas <- alphas
 result$levels <- levels(data[, target])

 return(result)
}
```

```

```
boosting_predict <- function(clfier, testdata) {

  n <- nrow(testdata)
  pred <- rep(NA_character_, n)

  for (i in 1:n) {

    myfreq <- data.frame(names = clfier$levels, freq = rep(0, length(clfier$levels)))

    for (j in 1:length(clfier$classifiers)) {
      prediction <- knn_predict(clfier$classifiers[[j]], testdata[i, ])
      myfreq[myfreq$names == prediction, ][2] <- myfreq[myfreq$names == prediction, ][2] + clfier$alphas[j]
    }

    most_frequent <- clfier$levels[myfreq$freq == max(myfreq$freq)]
    pred[i] <- sample(most_frequent, 1)

  }

  factor(pred, levels = clfier$levels)
}
...
```

```
```{r get_Vehicle_pred, eval=TRUE}
knn_boost_model <- knn_boosting('Class', df_train1)
pred1 <- boosting_predict(boosting, df_test1)
```
```

```
```{r get_Vehicle_error, eval=TRUE}
tb_knn1 <- table(df_test1$Class, pred1)
print(tb_knn1)
error1.knn <- 1 - sum(diag(tb_knn1)) / sum(tb_knn1)
cat("Тестовая ошибка (с использованием бустинг алгоритма с классификатором К
ближайших соседей), Vehicle: ", error1.knn, "\n")
```
```

| | pred1 | | | |
|------|-------|------|------|-----|
| | bus | opel | saab | van |
| bus | 55 | 3 | 14 | 2 |
| opel | 6 | 14 | 36 | 11 |
| saab | 5 | 15 | 29 | 8 |
| van | 2 | 0 | 0 | 54 |

Тестовая ошибка (с использованием бустинг алгоритма с классификатором К ближайших соседей), Vehicle: **0.4015748**

```
```{r get_Glass_pred, eval=TRUE}
knn_boost_model <- knn_boosting('Type', df_train2)
pred2 <- boosting_predict(knn_boost_model, df_test2)
```
```

```
```{r get_Glass_error, eval=TRUE}
tb_knn2 <- table(df_test2$Type, pred2)
print(tb_knn2)
error2.knn <- 1 - sum(diag(tb_knn2)) / sum(tb_knn2)
cat("Тестовая ошибка (с использованием бустинг алгоритма с классификатором К
ближайших соседей), Glass: ", error2.knn, "\n")
```
```

| | pred2 | | | | | | |
|---|-------|----|---|---|---|----|--|
| | 1 | 2 | 3 | 5 | 6 | 7 | |
| 1 | 15 | 5 | 0 | 0 | 0 | 0 | |
| 2 | 8 | 12 | 0 | 1 | 0 | 0 | |
| 3 | 4 | 0 | 1 | 0 | 0 | 0 | |
| 5 | 0 | 1 | 0 | 2 | 0 | 1 | |
| 6 | 0 | 0 | 0 | 1 | 1 | 1 | |
| 7 | 2 | 0 | 0 | 0 | 0 | 10 | |

Тестовая ошибка (с использованием бустинг алгоритма с классификатором К ближайших соседей), Glass: **0.3692308**

Видим, что на разных данных получились разные результаты.

Для набора Vehicle тестовая ошибка меньше при использовании единичного дерева классификации. А для набора Glass, напротив, меньшая ошибка при использовании бустинг алгоритма с KNN классификатором.