Санкт-Петербургский Политехнический Университет Петра Великого

Институт прикладной математики и механики

Кафедра Телематики при ЦНИИ РТК

Пояснительная записка

по дисциплине машинное обучение

«Разработка классификаторов для базы данных

Indian Liver Patient Dataset»

Преподаватель Уткин Л.В.

Студент гр.43607/1 Лисенкова А.А

Санкт-Петербург

2018 г.

Оглавление

[Введение 3](#_Toc513063749)

[Постановка задач 5](#_Toc513063750)

[1 Разработка классификаторов 6](#_Toc513063751)

[1.1 Метод опорных векторов 6](#_Toc513063752)

[1.2 Бэггинг 7](#_Toc513063753)

[1.3 Бустинг 8](#_Toc513063754)

[1.4 Выводы 8](#_Toc513063755)

[2 Кластеризация данных 10](#_Toc513063756)

[2.1 Clara 10](#_Toc513063757)

[2.2 Hierarchical Clustering 10](#_Toc513063758)

[2.3 k-means 11](#_Toc513063759)

[2.4 Выводы 11](#_Toc513063760)

[3 Определение наиболее значимых признаков 12](#_Toc513063761)

[4 Применение автокодера 13](#_Toc513063762)

[4.1 Автокодер для сокращения размерности 13](#_Toc513063763)

[4.2 Автокодер для реализации разреженного скрытого слоя 14](#_Toc513063764)

[4.3 Зашумленный автокодер 14](#_Toc513063765)

[4.4 Выводы 15](#_Toc513063766)

[Заключение 16](#_Toc513063767)

[Приложение. Листинг программы 17](#_Toc513063768)

# Введение

В работе используются следующие классификаторы:

1. Метод опорных векторов

* Максимизирует отступ между положительными и отрицательными объектами.
* Штрафует ошибки в случае неразделимой выборки.
* Только опорные векторы определяют решение.
* Отображение объектов при помощи ядер в новое нелинейное пространство/

1. Бустинг

* Ключевая идея: использование весовой версии одних и тех же обучающих данных вместо случайного выбора их подмножества.
* Слабые классификаторы образуются последовательно, различаясь только весами обучающих данных, которые зависят от точности предыдущих классификаторов.
* Большие веса назначаются “плохим” примерам, что позволяет на каждой итерации сосредоточиться на примерах, неправильно классифицированных.
* Базовые классификаторы должны быть слабыми, из сильных хорошую композицию не построить (“бритва Оккама”).

1. Бэггинг

* Благодаря различности базовых алгоритмов, их ошибки взаимно компенсируются при голосовании.
* Объекты-выбросы могут не попадать в некоторые обучающие подвыборки.
* Если каждый классификатор имеет высокую дисперсию (нестабильность), то комбинированный классификатор имеет меньшую дисперсию по сравнению с отдельными классификаторами.

Для кластеризации применяются следующие методы:

1. Метод k средних

Разделяет наблюдения на k кластеров, при этом каждое наблюдение относится к тому кластеру, к центру которого оно ближе всего.

1. Метод k медоидов

В отличии от первого метода, выбирает точки из набора данных в качестве центров и работает с произвольной метрикой расстояний между точками.

В работе используется база данных «Indian Liver Patient Dataset», содержащая 583 записи, среди которых 416 относятся к людям с болью в печени, а 167 – с отсутствием боли в печени. Выбранная база содержит в себе следующие признаки:

1. Age of the patient
2. Gender of the patient
3. Total Bilirubin
4. Direct Bilirubin
5. Alkaline Phosphotase
6. Alamine Aminotransferase
7. Aspartate Aminotransferase
8. Total Protiens
9. Albumin
10. Albumin and Globulin Ratio
11. Selector field used to split the data into two sets

Выполнение курсового проекта позволит предсказать вероятность наличия болей в печени у наблюдаемого, основываясь на построенных классификаторах.

В качестве программного обеспечения используется R-Studio.

# Постановка задач

Целью работы является исследование классификаторов с целью выявления наиболее оптимального для выбранной базы данных. Для каждого из классификаторов необходимо подобрать параметры для минимизации ошибки классификации. Кроме этого, необходимо осуществить кластеризацию данных с последующим определением ошибочно кластеризованных данных. Основываясь на методе Лассо, определить наиболее значимые признаки. Применить автокодер для сокращения размерности, реализации скрытого слоя нейронной сети, а также выполнить классификацию с использованием зашумленного автокодера. Сделать соответствующие выводы по проведённой работе, а именно выбрать наилучший классификатор.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать 3 классификатора и осуществить настройку их параметров для минимизации ошибки классификации на тестовых данных. Выполнить визуализацию данных при помощи метода t-SNE.
2. Сравнить классификаторы (по критерию вероятность ошибки классификации для тестовых данных) и обосновать выбор наилучшего из них.
3. Удалить их базы метки классов и осуществить кластеризацию данных. Построить дендограмму. Сравнить полученные результаты с реальными метками данных. Определить долю ошибочно кластеризованных данных.
4. Используя логистическую регрессию в рамках метода Лассо, определить наиболее значимые признаки, влияющие на отнесение объектов к определенному классу.
5. Использовать автокодер для сокращения размерности или для реализации разреженного скрытого слоя нейронной сети. Преобразовать обучающую выборку при помощи автокодера и осуществить классификацию новых данных с оценкой ошибки классификации. Выполнить визуализацию новых обучающих данных при помощи метода t-SNE. Определить, когда качество классификации лучше, если использовать сокращение размерности или разреженность скрытого слоя. Выполнить классификацию с использованием зашумленного автокодера (denoising autoencoder). Сравнить полученные результаты с пп.1 и 2.

# 1 Разработка классификаторов

Разработка классификаторов начинается с предварительной обработки данных: загрузка исходных данных из файла, разделение их на обучающую и тестирующую выборки, а также визуализация необработанных данных с помощью алгоритма t-SNE (см. Рис. 1).

Данные разделяются согласно следующему соотношению: 70% всех данных отводится на обучающую выборку, 30% - на тестирующую. Такой выбор объясняется тем, что для обучения предпочтительно отводить более половины всех данных, кроме этого, необходимо учитывать, что объём тестирующей выборки не должен быть слишком мал.

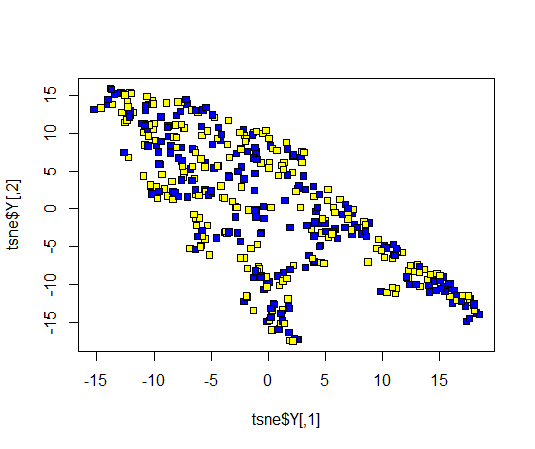


Рис. 1. Визуализация исходных данных с использованием t-SNE

Разработка классификаторов осуществляется с использованием следующих методов:

1. Метод опорных векторов.
2. Бэггинг.
3. Бустинг.

Оценка качества классификации производится согласно расчёту эмпирического функционала риска:

## Метод опорных векторов

Метод опорных векторов использует в качестве одного из параметров тип ядра. Основываясь на полученном разбиении исходной выборки, проведено исследование зависимости величины ошибки классификации от типа ядра. Полученные результаты представлены в Табл. 1.

|  |  |
| --- | --- |
| Тип ядра | Ошибка классификации |
| linear | 0.28 |
| radial | 0.29 |
| sigmoid | 0.30 |
| polynomial | 0.29 |

Табл. 1. Зависимость ошибки классификации от типа ядра для метода опорных векторов

После анализа Табл.1, принято решение взять линейное ядро, т.к. для него ошибка классификации оказалась наименьшей.

После построения классификатора найдена величина эмпирического функционала риска, которая составила 0.280.

## Бэггинг

В качестве одного из аргументов алгоритм бэггинга принимает значение mfinal, которое отвечает за число деревьев решений, которое необходимо построить. Для минимизации ошибки классификации проведено исследование значения ошибки от mfinal. Результаты представлены на Рис.2.

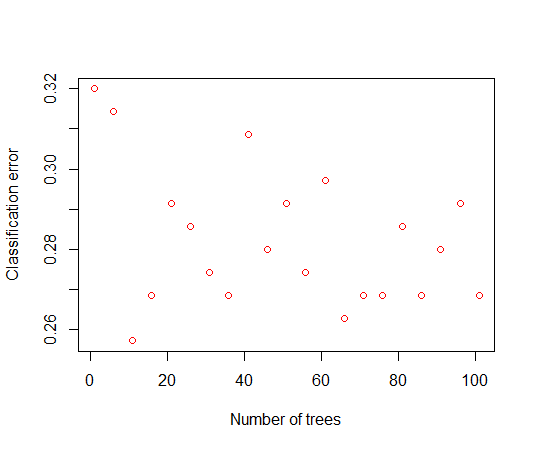


Рис. 2. Зависимость ошибки классификации от mfinal для алгоритма бэггинга

Из графика видно, что наименьшая ошибка достигается при значении mfinal = 11, которое соответственно будет использоваться в дальнейшем при построении классификатора.

После построения классификатора найдена величина эмпирического функционала риска, которая составила 0.246.

## Бустинг

В качестве одного из аргументов алгоритм бустинга принимает значение mfinal, которое отвечает за число деревьев решений, которое необходимо построить. Для минимизации ошибки классификации проведено исследование значения ошибки от mfinal. Результаты представлены на Рис.3.

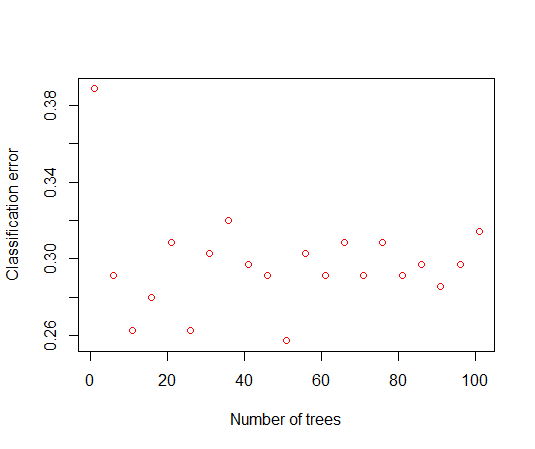


Рис. 3. Зависимость ошибки классификации от mfinal для алгоритма бустинга

Из графика видно, что наименьшая ошибка достигается при значении mfinal = 51, которое соответственно будет использоваться в дальнейшем при построении классификатора.

После построения классификатора найдена величина эмпирического функционала риска, которая составила 0.354.

## Выводы

В разделе рассмотрены три метода:

1. Метод опорных векторов.
2. Бэггинг.
3. Бустинг.

Для каждого метода подобраны оптимальные значения аргументов для достижения минимальной ошибки классификации. Построены графики и таблицы соответствующих зависимостей. Получены следующие результаты значений эмпирического функционала риска:

|  |  |
| --- | --- |
| Метод | Ошибка классификации |
| Метод опорных векторов | 0.290 |
| Бэггинг | 0.246 |
| Бустинг | 0.354 |

Табл. 2. Сравнительная таблица классификаторов

Основываясь на полученных значениях, можно сделать вывод о том, что для данного разбиения исходных данных наилучшим является классификатор, разработанный с помощью бэггинга.

# Кластеризация данных

Перед тем как осуществить кластеризацию, из заданной базы данных были удалены метки классов. Кластеризация проводится с помощью трёх методов:

* Clara (Clustering Large Applications)
* Hierarchical Clustering
* k-means

## Clara

В результате применения метода Clara, получаем график, представленный на Рис. 4, а также долю ошибочно кластеризованных данных, которая составила 0.556.

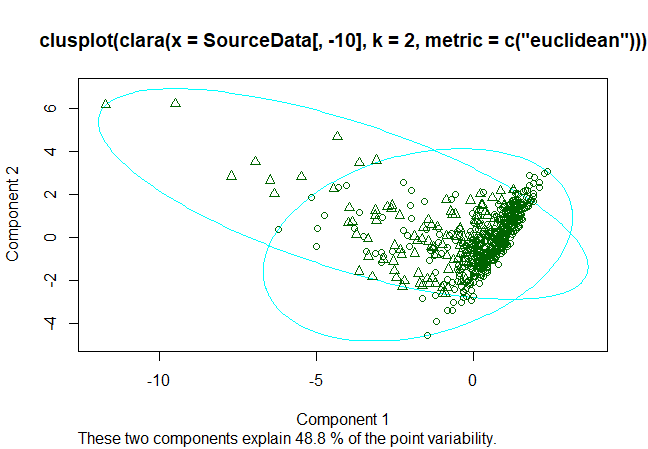


Рис. 4 Кластеризация методом Clara

## Hierarchical Clustering

В результате применения метода Hierarchical Clustering, получаем график, представленный на Рис. 5, а также долю ошибочно кластеризованных данных, которая составила 0.711.

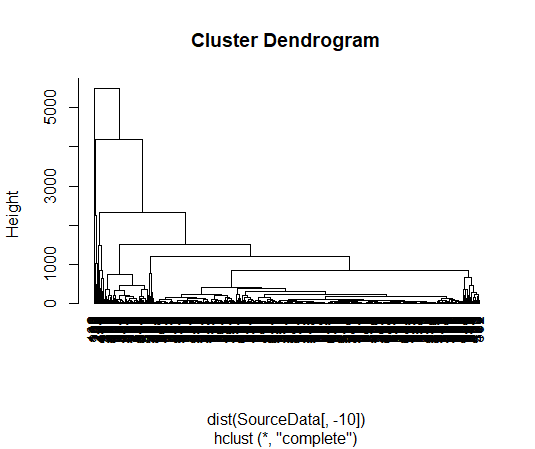


Рис. 5. Кластеризация методом Hierarchical Clustering

## k-means

Доля ошибочно кластеризованных данных составила 0.556.

## Выводы

В данном разделе рассмотрено три метода кластеризации данных. Для метода Clara и Hierarchical Clustering построены соответствующие графики. Анализируя полученные результаты долей ошибочно кластеризованных данных, можно сделать вывод о том, что методы Clara и k-means показали одинаковые результаты. В то время как метод Hierarchical Clustering оказался наименее эффективным для кластеризации.

# Определение наиболее значимых признаков

Одним из этапов машинного обучения является понижение размерности, т.е. отбор наиболее значимых признаков, которые влияют на отнесение объектов к определённому классу. С помощью логистической регрессии в рамках метода Лассо можно выявить наиболее значимые признаки.

Для реализации выше указанного метода используется функция glmnet, которая принимает на вход матрицу исходных данных, вектор меток класса, параметр alpha = 1, который указывает на то, что используется метод Лассо, а также параметр family = ‘binomial’, который был выбран согласно количеству меток.

Выделение наиболее значимых признаков заключается в исследовании значений коэффициентов : признаки, оказывающие наименьшее влияние на отнесение объектов к определённому классу, содержат большее количество нулей.

Список признаков:

1. Age of the patient
2. Gender of the patient
3. Total Bilirubin
4. Direct Bilirubin
5. Alkaline Phosphotase
6. Alamine Aminotransferase
7. Aspartate Aminotransferase
8. Total Protiens
9. Albumin
10. Albumin and Globulin Ratio
11. Selector field used to split the data into two sets

Анализируя полученные коэффициенты, можно сделать вывод о том, что наиболее значимыми признаками оказались 1,4,5,6,9.

# Применение автокодера

В главе используется 3 типа автокодера, каждый из которых характеризуется уникальным набором параметров.

После применения автокодера для реализации разреженного слоя полученные данные визуализированы с помощью алгоритма t-SNE (см. Рис. 6).

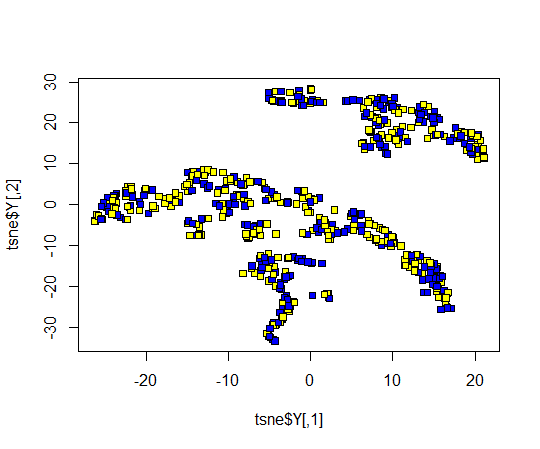


Рис. 6. Визуализация данных с использованием t-SNE

## Автокодер для сокращения размерности

Отличительной особенностью этого автокодера является то, что количество нейронов в скрытом слое должно быть меньше числа признаков. Таким образом, построение автокодера осуществляется со следующими параметрами:

* nl = 3
* N.hidden = 5
* epsilon = 0
* lambda = 0.0002

Преобразованные данные вновь разбиваются на обучающую и тестирующую выборки. После чего строятся классификаторы на основе методов опорных векторов, бэггинга и бустинга. Найденные ошибки классификации представлены в Табл. 3.

|  |  |
| --- | --- |
| **Метод** | **Ошибка классификации** |
| Метод опорных векторов | 0.297 |
| Бэггинг | 0.254 |
| Бустинг | 0.387 |

Табл. 3. Сравнительная таблица значений ошибок классификации

## Автокодер для реализации разреженного скрытого слоя

Отличительной особенностью этого автокодера является то, что количество нейронов в скрытом слое должно быть больше числа признаков. Таким образом, построение автокодера осуществляется со следующими параметрами:

* nl = 3
* N.hidden = 15
* epsilon = 0
* lambda = 0.0002

Преобразованные данные вновь разбиваются на обучающую и тестирующую выборки. После чего строятся классификаторы на основе методов опорных векторов, бэггинга и бустинга. Найденные ошибки классификации представлены в Табл. 4.

|  |  |
| --- | --- |
| **Метод** | **Ошибка классификации** |
| Метод опорных векторов | 0.217 |
| Бэггинг | 0.239 |
| Бустинг | 0.365 |

Табл. 4. Сравнительная таблица значений ошибок классификации

## Зашумленный автокодер

Отличительной особенностью этого автокодера является наличие погрешности, отличной от 0. Таким образом, построение автокодера осуществляется со следующими параметрами:

* nl = 3
* N.hidden = 15
* epsilon = 0.5
* lambda = 0.0002

Преобразованные данные вновь разбиваются на обучающую и тестирующую выборки. После чего строятся классификаторы на основе методов опорных векторов, бэггинга и бустинга. Найденные ошибки классификации представлены в Табл. 5.

|  |  |
| --- | --- |
| **Метод** | **Ошибка классификации** |
| Метод опорных векторов | 0.225 |
| Бэггинг | 0.273 |
| Бустинг | 0.371 |

Табл. 5. Сравнительная таблица значений ошибок классификации

## Выводы

После применения каждого из автокодеров были построены классификаторы, вычислены соответствующие ошибки, а также построены таблицы зависимостей. Анализируя полученные данные, можно сделать вывод о том, что для минимизации ошибки классификации необходимо использовать автокодер с разреженным слоем, а также алгоритм бэггинга (доля ошибочных данных составила 0.239).

# Заключение

В результаты проведённой работы для выбранной базы данных были применены различные методы машинного обучения. Сначала было разработано 3 классификатора, основанные на методах опорных векторов, бэггинга и бустинга. Для данных классификаторов были подобраны параметры, минимизирующие долю ошибки классификации. Наилучшим классификатором оказался бэггинг, который показал ошибку, равную 0.246.

Второй этап обучения состоял в кластеризации. Предварительно удалив метки класса, была проведена кластеризация тремя методами: hierarchical clustering, k-means, clara. Для каждого из методов была выявлена доля ошибочно кластеризованных данных. Наилучшие результаты показали методы k-means и clara, ошибка которых составила 0.556.

После проведенного обучения с учителем и без учителя было осуществлено определение наиболее значимых признаков, влияющих на отнесение объектов к определенному классу, используя логистическую регрессию в рамках метода Лассо. Были выявлены наиболее значимые признаки.

Заключительный этап обучения – использование автокодера. В процессе анализа выявлено, что наименьшую ошибку классификации имеют данные, полученные с использованием автокодера для реализации разреженного скрытого слоя. Полученная ошибка классификации с использованием автокодера оказалась меньше, чем ошибка классификации, полученная в процессе обучения данных с учителем аналогичным методом.

# Приложение. Листинг программы

library(e1071)

library(adabag)

library(rpart)

library(Rtsne)

library(cluster)

library(glmnet)

library(autoencoder)

SourceData <- read.csv("C:/Users/anna\_fox/Documents/UNIVERSE/4 курс/2 semester/Нейронные\_сети/All\_Labs/Kursovoj/Indian Liver Patient Dataset (ILPD).csv",stringsAsFactors = TRUE)

getBestKernel <- function(train, testCheck, kernels){

test <- testCheck[,-10]

errorList <- list()

for (i in 1:length(kernels)){

svmModelLinear <- svm(Select ~ ., data = train, type = "C-classification", cost = 1, kernel = kernels[i])

predictions <- predict(svmModelLinear, test)

myTable <- table(testCheck$"Select", predictions)

errorList[i] <- (myTable[2] + myTable[3]) / sum(myTable)

}

return(kernels[unlist(which.min(errorList))])

}

getBestMFinalBagging <- function(train,test,mfinal){

resList <- list()

for (i in 1:length(mfinal)) {

model <- bagging(Select ~.,data = train, mfinal = mfinal[i], maxdepth = 5)

predictions <- predict.bagging(model, newdata = test)

resList[i] <- predictions$error

}

return(resList)

}

getBestMFinalBoosting <- function(train,test,mfinal){

resList <- list()

for (i in 1:length(mfinal)) {

model <- boosting(Select ~.,data = train, mfinal = mfinal[i], maxdepth = 5)

predictions <- predict.boosting(model, newdata = test)

resList[i] <- predictions$error

}

return(resList)

}

############################################################################################

##Visual

############################################################################################

SourceData <- SourceData[,-10]

n <- dim(SourceData)[1]

idx <- sample(1:n, n\*0.7)

trainData <- SourceData[idx, ]

testData <- SourceData[-idx, ]

train\_data <- unique(trainData)

train\_data$Sex <- as.numeric(as.factor(train\_data$Sex))

tsne <- Rtsne(as.matrix(train\_data))

plot(tsne$Y,pch = 22, bg = c("blue","yellow"))

############################################################################################

##SVM

############################################################################################

#Нашли ядро с наименьшей ошибкой

kernel <- getBestKernel(trainData,testData,c("linear","radial","sigmoid","polynomial"))

#kkkkkk <- getBestKernel2(trainData,testData,c("linear","radial","sigmoid","polynomial"))

svmModelLinear <- svm(Select ~ ., data = trainData, type = "C-classification", cost = 1, kernel = kernel)

predictions <- predict(svmModelLinear, testData[,-10])

myTable <- table(testData$"Select", predictions)

errSvm <- (myTable[2] + myTable[3]) / sum(myTable)

############################################################################################

##BAGGING

############################################################################################

mfinal <- seq(1,6,5)

trainData$Select <- as.factor(trainData$Select)

resList <- getBestMFinalBagging(trainData,testData,mfinal)

plot(mfinal,resList,

col = "red",

xlab = "Number of trees",

ylab = "Classification error")

#Нашли количество деревьев с наименьшей ошибкой

treeNumberBag <- mfinal[unlist(which.min(resList))]

baggingModel <- bagging(Select ~.,data = trainData, mfinal = treeNumberBag, maxdepth = 5)

predictions <- predict.bagging(baggingModel, newdata = testData)

errBagging <- predictions$error

############################################################################################

##BOOSTING

############################################################################################

resList <- getBestMFinalBoosting(trainData,testData,mfinal)

plot(mfinal,resList,

col = "red",

xlab = "Number of trees",

ylab = "Classification error")

#Нашли количество деревьев с наименьшей ошибкой

treeNumberBoost <- mfinal[unlist(which.min(resList))]

boostingModel <- boosting(Select ~.,data = trainData, mfinal = treeNumberBoost, maxdepth = 5)

predictions <- predict.boosting(boostingModel, newdata = testData)

errBoosting <- predictions$error

############################################################################################

##CLUSTER

############################################################################################

cl <- clara(SourceData[,-10], 2,metric = c("euclidean"))

t <- table(cl$clustering,SourceData$Select)

errClara <- sum(diag(t))/sum(t)

plot(cl)

data <- SourceData

data$Sex <- as.numeric(as.factor(data$Sex))

kkk <- kmeans(data[,-10],2,iter.max = 10)

t <- table(kkk$cluster,SourceData$Select)

errKMeans <- sum(diag(t))/sum(t)

# prepare hierarchical cluster

hc = hclust(dist(SourceData[,-10]))

# very simple dendrogram

plot(hc,hang=-1)

groups<-cutree(hc,k=2)

#rect.hclust(hc,k=2,border = "red")

t<-table(groups,SourceData[,10])

(sum(diag(t))/sum(t))

############################################################################################

##LASSO

############################################################################################

x <- as.matrix(data[,-10])

y <- data[,10]

glm = glmnet(x, y, family = "binomial", alpha = 1)

lassoResult <- as.matrix(glm$beta)

############################################################################################

##AUTOENCODER

############################################################################################

testData$Sex <- as.numeric(as.factor(testData$Sex))

trainData$Sex <- as.numeric(as.factor(trainData$Sex))

encoder <- autoencode(as.matrix(data[,-10]),

N.hidden = 15,

epsilon = 0,

lambda = 0.0002,

beta = 6,

rho = 0.9,

unit.type = "tanh",

rescale.flag = TRUE)

predictions <- predict.autoencoder(encoder,as.matrix(data[,-10]),hidden.output = TRUE)

predFrame <- as.data.frame(predictions$X.output)

autoEncodedData <- tibble::add\_column(predFrame,data$Select)

tsne <- Rtsne(as.matrix(unique(autoEncodedData)),perplexity = 20)

plot(tsne$Y,pch = 22, bg = c("blue","yellow"))

#Поменяли имя колонки (класс)

colnames(autoEncodedData)[which(names(autoEncodedData) == "data$Select")] <- "Select"

autoEncodedData$Select <- as.factor(autoEncodedData$Select)

#Разбили выборку на обучающую и тестирующую

idx <- sample(1:n, n\*0.7)

trainData <- autoEncodedData[idx, ]

testData <- autoEncodedData[-idx, ]

#BAGGING

baggingModel <- bagging(Select ~.,data = trainData, mfinal = treeNumberBag, maxdepth = 5)

predictions <- predict.bagging(baggingModel, newdata = testData)

errBagging2 <- predictions$error

#BOOSTING

boostingModel <- boosting(Select ~.,data = trainData, mfinal = treeNumberBoost, maxdepth = 5)

predictions <- predict.boosting(boostingModel, newdata = testData)

errBoosting2 <- predictions$error

#SVM

svmModelLinear <- svm(Select ~ ., data = trainData, type = "C-classification", cost = 1, kernel = kernel)

predictions <- predict(svmModelLinear, testData[,-21])

myTable <- table(testData$"Select", predictions)

errSvm <- (myTable[2] + myTable[3]) / sum(myTable)