1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и технологий
5. Высшая школа искусственного интеллекта

**КУРСОВАЯ РАБОТА**

**«Разработка классификатора для базы данных Pima Indians Diabetes»**

по дисциплине «Машинное обучение, часть1»

1. Выполнил: студент группы
2. 3540201/20302 С.А. Ляхова

*<подпись>*

1. Проверил: Л.В. Уткин
2. д.т.н., профессор
3. *<подпись>*

Санкт-Петербург

2022

**Содержание**

[1. Цель работы 3](#_Toc122226865)

[2. Формулировка задания 3](#_Toc122226866)

[3. Ход работы 5](#_Toc122226867)

[3.1. Вариант задания 5](#_Toc122226868)

[3.2. Описание базы данных 5](#_Toc122226869)

[3.3. Разработка классификаторов 6](#_Toc122226870)

[3.4. Кластеризация данных 10](#_Toc122226871)

[3.5. Определение наиболее значимых признаков 11](#_Toc122226872)

[3.6. Использование автокодера 12](#_Toc122226873)

[4. Вывод 15](#_Toc122226874)

[Приложение 1 16](#_Toc122226875)

[Приложение 2 20](#_Toc122226876)

[Приложение 3 21](#_Toc122226877)

[Приложение 4 22](#_Toc122226878)

# **Цель работы**

Целью данной работы является применение и анализ работы алгоритмов машинного обучение без учителя и с учителем на основе реальной базы данных.

Необходимо разработать алгоритмы классификации, кластеризации и проверить их работу на реальных данных. Кроме того, нужно применить методы логистической регрессии и оценить наиболее значимые параметры выбранного набора данных.

Также необходимо провести классификацию с предварительной обработкой данных, основанной на использовании нейронных сетей – автокодеров.

# **Формулировка задания**

Для базы данных необходимо:

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.

# **Ход работы**

## **Вариант задания**

1. В качестве методов классификации были выбраны следующие методы:
2. Метод опорных векторов
3. Беггинг
4. Бустинг
5. В качестве метода кластеризации был выбран метод k-медоидов.
6. В качестве базы данных была выбрана Pima Indians Diabetes Database (pima-indians-diabetes.data)

## **Описание базы данных**

Pima Indians Diabetes Database – набор данных об индейцах Пима, больных диабетом.

База данных была создана 9 мая 1990 года и принадлежит Национальному институту диабета, желудочно-кишечных и почечных заболеваний (NIDDK). Все пациенты этой базы данных женского пола и старше 20 лет.

База данных содержит 768 записей и 9 атрибутов:

1. Number of times pregnant – количество беременностей пациентки
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test – концентрация глюкозы в плазме пациентки через 2 часа после приема пищи
3. Diastolic blood pressure – диастолическое артериальное давление
4. Triceps skin fold thickness – толщина кожной складки трицепса
5. 2-Hour serum insulin – уровень инсулина через 2 часа после приема пищи
6. Body mass index – индекс массы тела
7. Diabetes pedigree function – значение функции, которая оценивает вероятность диабета на основе семейного анамнеза
8. Age (years) – возраст (в годах)
9. Class variable (0 or 1) – принадлежность к классу диабетиков

Переменная с названием «Class» принимает 2 значения: «0» означает, что пациентка не больна сахарным диабетом, и «1», что имеет такое заболевание. «1» в данном столбце означает, что через 2 часа после приема пищи уровень глюкозы в плазме пациентки был не менее 200 мг/дл.

## **Разработка классификаторов**

Перед построением классификаторов была выполнена визуализация данных с помощью t-SNE. По рисунку №1 можем видеть, что данные визуально разделены на три кластера, причем в каждом из кластеров имеются как здоровые, так и больные пациентки в примерно равном соотношении.

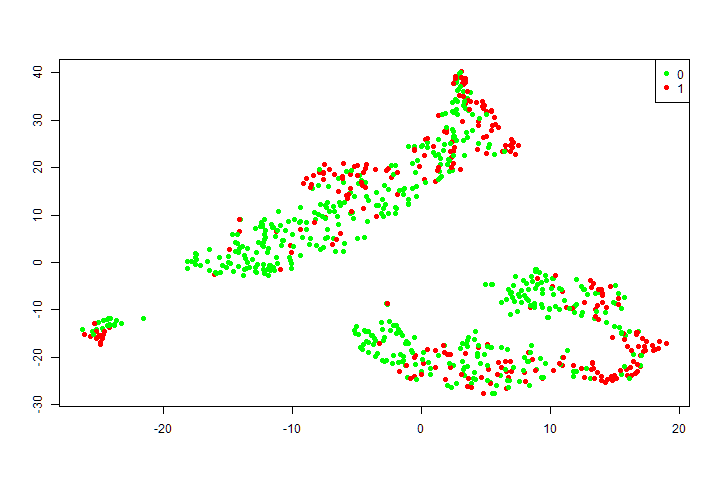


Рисунок 1. Визуализация данных с помощью метода t-SNE

Красными точками изображены больные диабетом, зеленым – здоровые.

*Метод опорных векторов.*

Подбор параметров, минимизирующих ошибку на тестовых данных, осуществлялся в два этапа. Вначале происходил выбор лучшего ядра и параметра штрафа на изучаемом датасете.

80% данных были использованы для обучения, 20% - для тестирования.

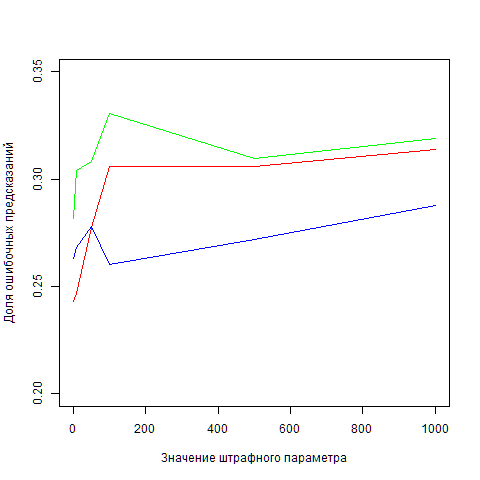


Рисунок 2. Доля ошибочных предсказаний метода SVM (красный – radial, зеленый – sigmoid, синий – poly)

Наименьшую ошибку на тестовых данных показали параметры **радиального ядра с параметром штрафа C=1** (0.2428571).

При этом использовались лишь три ядра: “radial”, “sigmoid” и “poly”, поскольку с линейным ядром алгоритм часто достигал максимального числа итераций и не показывал сходимости.

Далее, учитывая результаты выбора ядра и параметра штрафа, проводился выбор параметра гамма.

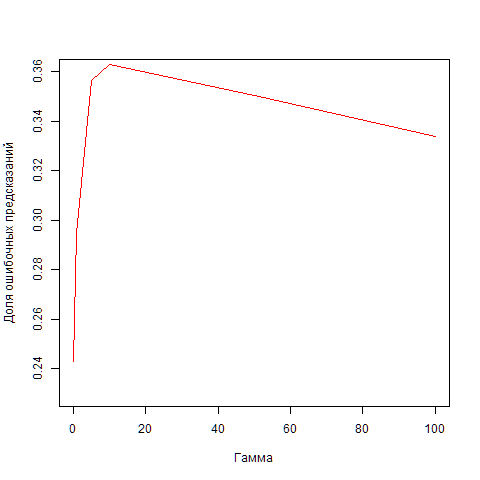


Рисунок 3. Доля ошибочных предсказаний метода SVM для разного параметра γ

Наилучший результат на этом этапе показал **параметр γ=0.1** (0.2357143). Увеличение параметра гамма отрицательно сказывается на результатах.

На каждом этапе было проведено по 10 одинаковых итераций для каждого значения параметров, а затем данные усреднялись.

Наилучшие результаты показала модель с радиальным ядром, с параметром штрафа равным 1, γ=0.1, которая показала 76.4% верно классифицированных данных.

*Бэггинг и Бустинг.*

Алгоритмы запускались с различным количеством деревьев классификации: от 1 до 201 с шагом 10, с одним и тем же параметром 4 раза, затем данные усреднялись.

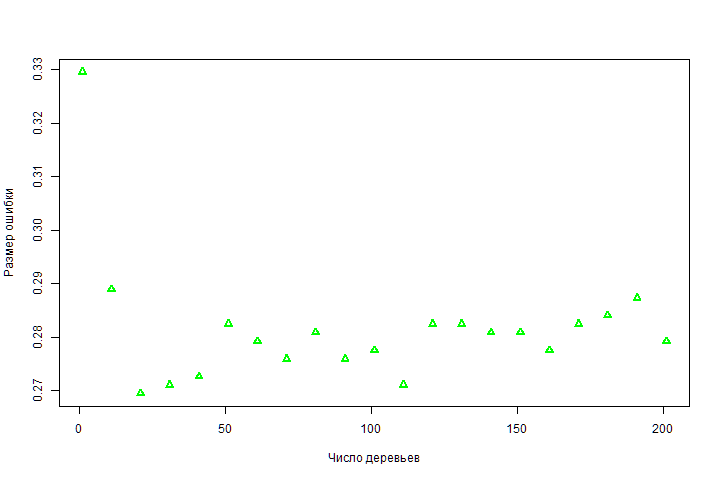


Рисунок 4. Размер ошибки в зависимости от числа деревьев в методе Bagging

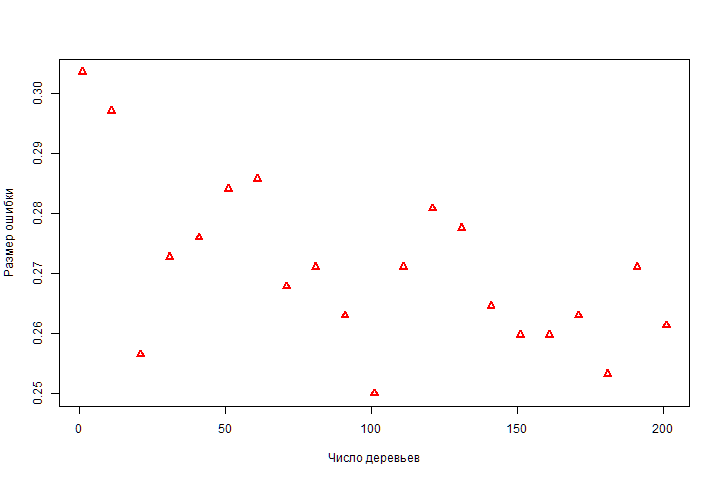


Рисунок 5. Размер ошибки в зависимости от числа деревьев в методе Boosting

Для bagging наилучшим оказалось количество деревьев, равное 21 с ошибкой 0.2694805. Для boosting наименьшую ошибку дало 101 дерево –с ошибкой классификации 0.25.

По рисункам 4-5 видно, что bagging показывает в среднем большую ошибку, чем boosting, однако она примерно одинаковая при различном числе деревьев, в отличие от boosting, где размах значений ошибки гораздо выше.

*Сравнение классификаторов.*

Наилучшим классификатором оказался тот, который был построен методом опорных векторов с ошибкой классификации 23.57%, в то время как ошибка в boosting составила 25%, а в bagging – 26.9%.

Однако нельзя не отметить, что для получения лучших показателей для svm пришлось долго подбирать нужные параметры метода. В общем случае его ошибка примерно равна (или в отдельных кейсах даже выше) ошибке, показанной в среднем методами bagging или boosting. Подходящим более простым в настройке методом можно назвать boosting.

## **Кластеризация данных**

Была построена дендрограмма исходного датасета (произведена иерархическая кластеризация). В качестве метрики было использовано евклидово расстояние.

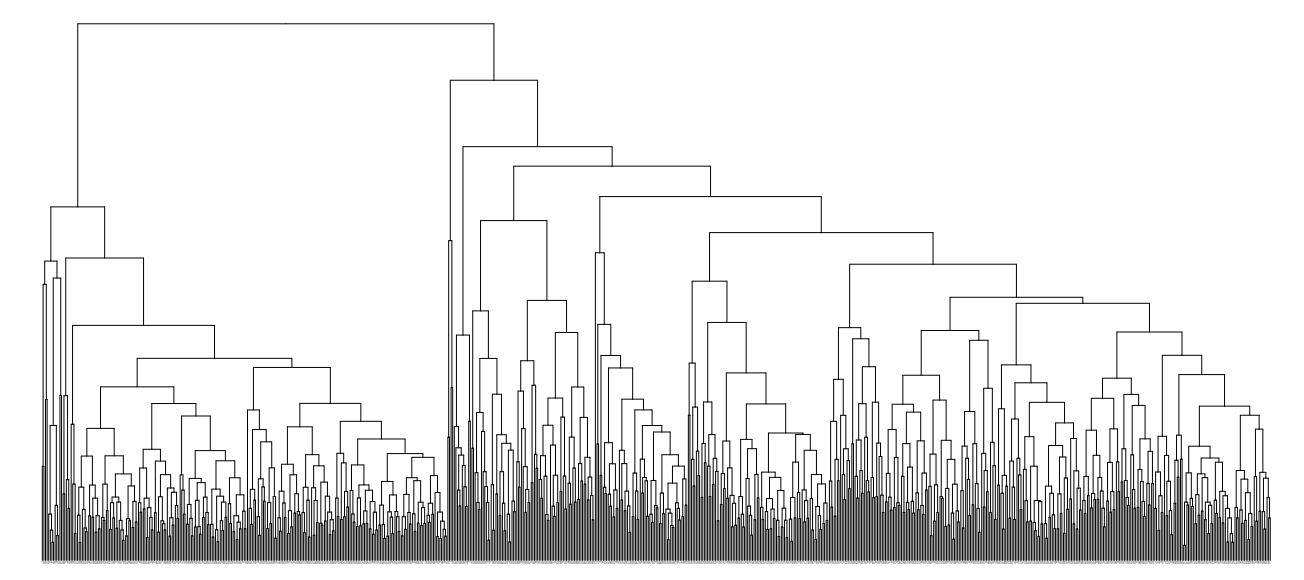


Рисунок 6. Дендрограмма датасета

Кластеризация данных проводилась методом k-медоидов. Была выбрана евклидова метрика без параметра стандартизации, поскольку с таким набором параметров кластеризация точнее. Алгоритм был запущен 10 раз, а затем доля ошибочно кластеризованных данных была усреднена.

В среднем около 310 экземпляров данных из 768 кластеризованны неправильно. Точность кластеризации составляет 59.6%.

Можно сделать вывод, что такой плохой результат получается в следствии схожести данных, что можно увидеть на рисунке 1.

## **Определение наиболее значимых признаков**

С использованием метода Лассо были определены наиболее значимые признаки для данного набора данных.

Наиболее значимые признаки определяют в большей степени принадлежность объекта к тому или иному классу. При этом присутствие неинформативных признаков приводит к снижению точности модели.

Для выполнения использовался стандартный метод glmnet, в качестве аргументов которого передаются матрица обучаемых данных, вектор меток, параметры alpha равный единице (это означает, что используется метод Лассо) и family равный “binomial” (данный выбор сделан в пользу того, что имеется только два вида меток).

Исходя из значений коэффициентов получилась следующая градация признак по возрастанию их значимости:

4. Triceps skin fold thickness

5. 2-Hour serum insulin

3. Diastolic blood pressure

7. Diabetes pedigree function

8. Age

1. Number of times pregnant

6. Body mass index

2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test

## **Использование автокодера**

*Использование автокодера для сокращения размерности*

Для данного задания была использована библиотека h2o, функция h2o.deeplearning() с установленным autoencoder = TRUE. Используется один скрытый слой (размером 2), это сокращает 8 функций до двух измерений. Используется функция активации гиперболического тангенса, которая имеет нелинейную сигмоидальную форму. Чтобы извлечь данные уменьшенной размерности, используется функция h2o.deepfeatures().

Получим визуализацию данных с помощью tSNE.

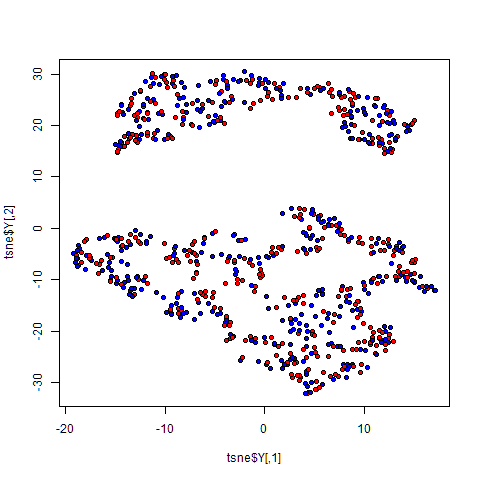


Рисунок 7. Визуализация данных сокращенной размерности с помощью tSNE

Исходя из изображения также трудно разделить «больных» и «здоровых».

Была выполнена классификация с использованием SVM, предварительно были подобраны параметры (kernel = "radial", cost = 10). Ошибка для сокращенной размерности составила – 0.2532468.

*Использование автокодера для реализации разреженного скрытого слоя*

Разреженные автокодеры используются для извлечения наиболее важных представлений функций. Это полезно при попытке понять, каковы наиболее уникальные особенности набора данных, при использовании автокодеров в качестве входных данных для последующих контролируемых моделей.

Включение разреженности заставляет больше нейронов быть неактивными. Это требует, чтобы автокодер представлял каждый вход как комбинацию меньшего числа активаций.

Изменим число нейронов в скрытом слое (hidden = 100).

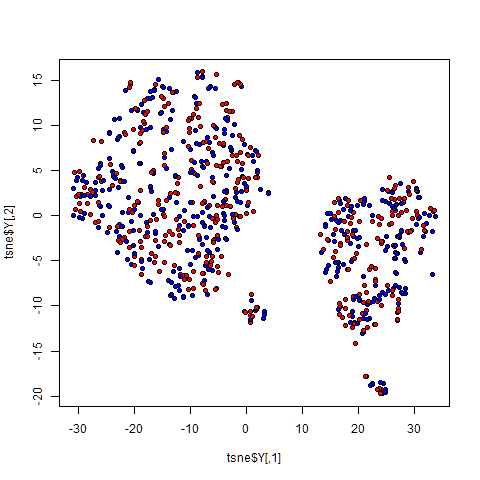


Рисунок 8. Визуализация данных разреженного автокодера

Ошибка для разреженного автокодера составила – 0.2207792.

*Использование зашумленного автокодера*

Шумоподавляющий автокодер — это стохастическая версия автокодера, в которой автокодер обучается восстанавливать входные данные из поврежденной копии входных данных.

Обучение шумоподавляющего автокодера почти такое же, как обучение обычного автоодера. Единственная разница в том, что передаются поврежденные входные данные в training\_frame, а неповрежденные входные данные — в validation\_frame.

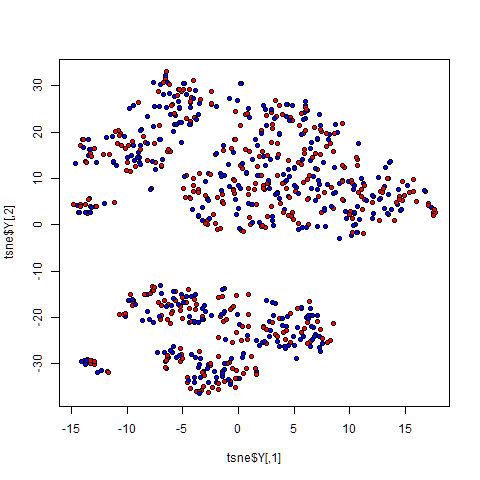


Рисунок 9. Визуализация данных зашумленного автокодера

Ошибка шумоподавляющего автокодера составила – 0.2142857.

Сравнивая полученные результаты с результатами из первого задания можно сделать вывод, что на данной выборке лучше работать с шумоподавляющего автокодером, тогда как автокодер для сокращенной размерности дает результаты хуже, чем исходная модель.

# **Вывод**

В данном роботе для базы данных «Pima Indians Diabetes Database»:

* Реализованы 3 классификатора, основанные на методах обучения с учителем, такие как метод опорных векторов, бэггинг и бустинг. Для данных классификаторов были подобраны параметры для получения наименьшей ошибки классификации. Наилучшим классификатором оказался метод опорных векторов с точностью классификации 76,43%.
* Реализован метод обучения без учителя – метод k-медоидов. Точность кластеризации составляет 59.6%. Можно сделать вывод, что такой плохой результат получается в следствии схожести данных.
* Было осуществлено определение наиболее значимых признаков, влияющих на отнесение объектов к определенному классу, используя логистическую регрессию в рамках метода Лассо.
* Были применены автокодеры. В процессе анализа выявлено, что наибольшую точность классификации имеют данные, полученные с использованием шумоподавляющего автокодера – 80.52%. В качестве метода классификации новых данных был использован метод опорных векторов с определенными оптимальными параметрами. Полученная ошибка классификации с использованием разреженного автокодера больше, чем ошибка классификации, полученная в процессе обучения данных с учителем аналогичным методом.
* Методом t-SNE визуализированы исходные данные, а также данные после прохождения автокодера.

# **Приложение 1**

# Задние 1 -------------------------------------------------------------

require(e1071)

require(tree)

require(maptree)

require(mltools)

diabetes <- read.csv(paste(path, "pima-indians-diabetes.data", sep = ""), sep = ",")

colnames(diabetes) <- c("Pregnant\_times",

"Glucose\_concentration",

"Blood\_pressure",

"Skin\_thickness",

"Insulin",

"BMI",

"Pegigree",

"Age",

"Class")

diabetes$Class <- as.factor(diabetes$Class)

install.packages("Rtsne")

library(Rtsne)

data\_plot <- Rtsne(diabetes)

png(paste(path, "diabetes\_plot.png"), width = 720)

plot(data\_plot$Y, pch = 19, col = c("green", "red")[diabetes$Class], xlab = "", ylab = "")

legend(x="topright", legend = levels(diabetes$Class), col=c("green","red"), pch=19)

dev.off()

# предварительный анализ SVM

n <- nrow(diabetes)

ratio <- 0.8

nt <- as.integer(n \* ratio)

kernels <- c("radial", "sigmoid", "poly")

result <- vector()

for (k in kernels)

{

res <- vector()

for (c in c(1, 10, 50, 100, 500, 1000))

{

tmp\_res <- vector()

for (i in 1:10)

{

diabetes\_rand <- diabetes[order(runif(n)), ]

diabetes\_train <- diabetes\_rand[1: nt, ]

diabetes\_test <- diabetes\_rand[(nt + 1): n, ]

class <- diabetes\_test$Class

diabetes\_test$Class <- NULL

model <- svm(Class ~ ., data=diabetes\_train, type = "C-classification", cost = c, kernel = k)

predicted <- predict(model, diabetes\_test)

tbl <- table(predicted, class)

tmp\_res <- append(tmp\_res, (tbl[1 , 2] + tbl[2, 1])/ length(class))

}

res <- append(res, mean(tmp\_res))

}

print(res)

result <- append(result, res)

}

M <- matrix(result, ncol = 3, nrow = 6)

a<-t(M)

png(paste(path, "Result.png"))

x<-c(1, 10, 50, 100, 500, 1000)

plot(x, a[1,], type = "l", col="red", xlab="Значение штрафного параметра", ylab="Доля ошибочных предсказаний", ylim=c(0.20,0.35))

lines(x, a[2,], type = "l", col="green")

lines(x, a[3,], type = "l", col="blue")

dev.off()

# ---------------------------------------------------------------

gammas <- c(0.1, 0.5, 1, 5, 10, 50, 100)

res <- vector()

for (g in gammas)

{

tmp\_res <- vector()

for (i in 1:10)

{

diabetes\_rand <- diabetes[order(runif(n)),]

diabetes\_train <- diabetes\_rand[1:nt,]

diabetes\_test <- diabetes\_rand[(nt + 1):n,]

class <- diabetes\_test$Class

diabetes\_test$Class <- NULL

model <- svm(

Class ~ .,

data = diabetes\_train,

type = "C-classification",

cost = 1,

kernel = "radial",

gamma = g

)

predicted <- predict(model, diabetes\_test)

tbl <- table(predicted, class)

tmp\_res <- append(tmp\_res, (tbl[1 , 2] + tbl[2, 1]) / length(class))

}

res <- append(res, mean(tmp\_res))

print(res)

}

png(paste(path, "Result2.png"))

x<-c(0.1, 0.5, 1, 5, 10, 50, 100)

plot(x, res, type = "l", col="red", xlab="Гамма", ylab="Доля ошибочных предсказаний", ylim=c(0.23,0.36))

dev.off()

# -----------boosting и bagging----------------------------------------

library(adabag)

diabetes\_rand <- diabetes[order(runif(n)), ]

diabetes\_train <- diabetes\_rand[1: nt, ]

diabetes\_test <- diabetes\_rand[(nt + 1): n, ]

diabetes$Class <- as.factor(diabetes$Class)

tree\_num <- seq(1, 201, 10)

error\_bag <- vector()

error\_boost <- vector()

for (t in tree\_num)

{

err\_bag <- vector()

err\_boost <- vector()

for(i in 1:4)

{

clf1 <- bagging(Class ~ ., data = diabetes\_train, mfinal = t)

err\_bag <- append(err\_bag, predict(clf1, diabetes\_test)$error)

clf2 <- boosting(Class ~ ., data = diabetes\_train, mfinal = t)

err\_boost <- append(err\_boost, predict(clf2, diabetes\_test)$error)

}

error\_bag <- append(error\_bag, mean(err\_bag))

error\_boost <- append(error\_boost, mean(err\_boost))

}

mean(error\_bag)

mean(error\_boost)

error\_bag

error\_boost

png(paste(path, "bagging.png"), width = 720)

plot(x = tree\_num, y = error\_bag, xlab = "Число деревьев", ylab = "Размер ошибки", pch = 2, lwd = 2, col = "green")

dev.off()

png(paste(path, "boosting.png"), width = 720)

plot(x = tree\_num, y = error\_boost, xlab = "Число деревьев", ylab = "Размер ошибки", pch = 2, lwd = 2, col = "red")

dev.off()

# **Приложение 2**

# Задание 3--------------------------------------------------------------

library(dplyr)

library(cluster)

diabetes <- read.csv(paste(path, "pima-indians-diabetes.data", sep = ""), sep = ",")

colnames(diabetes) <- c("Pregnant\_times",

"Glucose\_concentration",

"Blood\_pressure",

"Skin\_thickness",

"Insulin",

"BMI",

"Pegigree",

"Age",

"Class")

diabetes$Class <- NULL

matr <- dist(scale(diabetes))

hc <- hclust(matr)

hcd <- as.dendrogram(hc)

pdf(paste(path, "dendrogramma.pdf"), width = 45, height = 20)

par(cex = 0.3)

plot(hcd, cex = 0.7)

dev.off()

class <- diabetes$Class

diabetes$Class <- NULL

# будем использовать Евклидову метрику без параметра стандартизации, т.к.

# выяснили, что это точнее всего в лабе по кластеризации

tmp <- vector()

for(i in 1:10)

{

cl <- clara(diabetes, 2, rngR = TRUE)

cl$clustering[cl$clustering == 1] <- 0

cl$clustering[cl$clustering == 2] <- 1

tmp <- append(tmp, sum(abs(cl$clustering - class)))

}

1 - mean(tmp)/length(class)

mean(tmp)

sum(tmp)/10

# **Приложение 3**

# Задание 4 ------------------------------------------------------------

install.packages("glmnet")

library(glmnet)

diabetes <- read.csv(paste(path, "pima-indians-diabetes.data", sep = ""), sep = ",")

colnames(diabetes) <- c("Pregnant\_times",

"Glucose\_concentration",

"Blood\_pressure",

"Skin\_thickness",

"Insulin",

"BMI",

"Pegigree",

"Age",

"Class")

x<-as.matrix(diabetes[,-9])

y<-diabetes[,9]

glm<-glmnet(x,y,family="binomial",nlambda =97,alpha =1)

res<-as.matrix(glm$beta)

# **Приложение 4**

# Задание 5 ------------------------------------------------------------

install.packages("h2o")

# Modeling packages

library(h2o) # for fitting autoencoders

diabetes <- read.csv(paste(path, "pima-indians-diabetes.data", sep = ""), sep = ",")

colnames(diabetes) <- c("Pregnant\_times",

"Glucose\_concentration",

"Blood\_pressure",

"Skin\_thickness",

"Insulin",

"BMI",

"Pegigree",

"Age",

"Class")

h2o.init()

features <- as.h2o(diabetes[-9])

ae1 <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = features,

autoencoder = TRUE,

hidden = 2,

activation = 'Tanh'

)

# Extract the deep features

ae1\_codings <- h2o.deepfeatures(ae1, features, layer = 1)

ae1\_codings

data\_coding <- as.data.frame(ae1\_codings)

data\_coding$Class <- diabetes$Class

tsne <- Rtsne(data\_coding)

png(paste(path, "Autoencode.png"))

plot(tsne$Y, pch = 21, bg = c("red", "blue"))

dev.off()

n <- nrow(data\_coding)

rand <- data\_coding[ order(runif(n)),]

nt <- as.integer(n\*0.8)

data\_train <- rand[1:nt, ]

data\_test <- rand[(nt+1):n, ]

kernels <- c("radial", "sigmoid", "poly")

result <- vector()

class <- data\_test$Class

data\_test$Class <- NULL

for (k in kernels)

{

res <- vector()

for (c in c(1, 10, 50, 100, 500, 1000))

{

tmp\_res <- vector()

for (i in 1:10)

{

model <- svm(Class ~ ., data=data\_train, type = "C-classification", cost = c, kernel = k)

predicted <- predict(model, data\_test)

tbl <- table(predicted, class)

tmp\_res <- append(tmp\_res, (tbl[1 , 2] + tbl[2, 1])/ length(class))

}

res <- append(res, mean(tmp\_res))

}

print(res)

result <- append(result, res)

}

auto\_svm <- svm(Class ~ ., data = data\_train, type = "C-classification", cost = 10, kernel = "radial")

pred\_svm <- predict(auto\_svm, data\_test)

tbl <- table(pred\_svm, class)

res <- (tbl[1 , 2] + tbl[2, 1])/ length(class)

# -------------------------------------------------------------------

h2o.init()

features <- as.h2o(diabetes[-9])

ae2 <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = features,

autoencoder = TRUE,

hidden = 100,

activation = 'Tanh'

)

# Extract the deep features

ae2\_codings <- h2o.deepfeatures(ae2, features, layer = 1)

ae2\_codings

data\_coding <- as.data.frame(ae2\_codings)

data\_coding$Class <- diabetes$Class

tsne <- Rtsne(data\_coding)

png(paste(path, "Autoencode2.png"))

plot(tsne$Y, pch = 21, bg = c("red", "blue"))

dev.off()

n <- nrow(data\_coding)

rand <- data\_coding[ order(runif(n)),]

nt <- as.integer(n\*0.8)

data\_train <- rand[1:nt, ]

data\_test <- rand[(nt+1):n, ]

kernels <- c("radial", "sigmoid", "poly")

result <- vector()

class <- data\_test$Class

data\_test$Class <- NULL

for (k in kernels)

{

res <- vector()

for (c in c(1, 10, 50, 100, 500, 1000))

{

tmp\_res <- vector()

for (i in 1:10)

{

model <- svm(Class ~ ., data=data\_train, type = "C-classification", cost = c, kernel = k)

predicted <- predict(model, data\_test)

tbl <- table(predicted, class)

tmp\_res <- append(tmp\_res, (tbl[1 , 2] + tbl[2, 1])/ length(class))

}

res <- append(res, mean(tmp\_res))

}

print(res)

result <- append(result, res)

}

auto\_svm <- svm(Class ~ ., data = data\_train, type = "C-classification", cost = 10, kernel = "radial")

pred\_svm <- predict(auto\_svm, data\_test)

tbl <- table(pred\_svm, class)

res <- (tbl[1 , 2] + tbl[2, 1])/ length(class)

# ----------------------------------------------------------------------

features <- as.h2o(diabetes[-9])

features\_2 <- as.h2o(diabetes[-9] + rnorm(nrow(diabetes), mean = 0, sd = 0.1))

denoise\_ae <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = features\_2,

validation\_frame = features,

autoencoder = TRUE,

hidden = 70,

activation = 'Tanh',

sparse = TRUE

)

denoise\_ae\_codings <- h2o.deepfeatures(denoise\_ae, features, layer = 1)

data\_coding <- as.data.frame(denoise\_ae\_codings)

data\_coding$Class <- diabetes$Class

tsne <- Rtsne(data\_coding)

png(paste(path, "Autoencode3.png"))

plot(tsne$Y, pch = 21, bg = c("red", "blue"))

dev.off()

n <- nrow(data\_coding)

rand <- data\_coding[order(runif(n)),]

nt <- as.integer(n\*0.8)

data\_train <- rand[1:nt, ]

data\_test <- rand[(nt+1):n, ]

kernels <- c("radial", "sigmoid", "poly")

result <- vector()

class <- data\_test$Class

data\_test$Class <- NULL

for (k in kernels)

{

res <- vector()

for (c in c(1, 10, 50, 100, 500, 1000))

{

tmp\_res <- vector()

for (i in 1:10)

{

model <- svm(Class ~ ., data=data\_train, type = "C-classification", cost = c, kernel = k)

predicted <- predict(model, data\_test)

tbl <- table(predicted, class)

tmp\_res <- append(tmp\_res, (tbl[1 , 2] + tbl[2, 1])/ length(class))

}

res <- append(res, mean(tmp\_res))

}

print(res)

result <- append(result, res)

}

auto\_svm <- svm(Class ~ ., data = data\_train, type = "C-classification", cost = 10, kernel = "poly")

pred\_svm <- predict(auto\_svm, data\_test)

tbl <- table(pred\_svm, class)

res <- (tbl[1 , 2] + tbl[2, 1])/ length(class)

res