1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —

Институт компьютерных наук и технологий

Высшая школа искусственного интеллекта

**КУРСОВАЯ РАБОТА**

**«Разработка классификатора для базы данных Mammographic masses»**

Выполнила

студентка гр. 3540201/20302 Обидина А.И.

Проверил

профессор Уткин Л.В.

Санкт-Петербург

2022

# **Цель работы**

Целью данной работы является применение алгоритмов машинного обучение без учителя и с учителем для разработки классификаторов для реальной базы данных.

Необходимо разработать алгоритмы классификации, кластеризации и проверить их работу на реальных данных.

Методы классификации:

1. Метод k ближайших соседей.
2. Беггинг.
3. Бустинг.

Методы кластеризации:

1. Метод k средних.

Также для оценки наиболее значимых параметров выбранного набора данных нужно применить методы логистической регрессии.

Также необходимо провести классификацию с предварительным преобразованием данных с помощью 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**

## **Описание базы данных**

Этот набор данных может использоваться для прогнозирования тяжести (доброкачественного или злокачественного) поражения по признакам BI-RADS и возрасту пациента. Данные содержат оценку BI-RADS, возраст пациента и три признака BI-RADS с диагнозами (злокачественное или доброкачественное поражение). База, полученная в Институте радиологии Университет Эрланген-Нюрнберг в 2003–2006 гг., содержит 516 доброкачественных и 445 злокачественных образований (всего 961 случаев).

Каждый случай имеет 6 атрибутов:

* Оценка BI-RADS: 1-5 (порядковый)
* Возраст: возраст пациента в годах (целое число).
* Форма: массовая форма: круглая = 1, овал = 2, лобулярная = 3, нерегулярная = 4 (номинальный).
* Видимость: ограничена = 1, микролобуляция = 2, затемнена = 3, плохо определена = 4, ясная = 5 (номинальный).
* Плотность: высокая = 1, средняя = 2, низкая = 3, жиросодержащая = 4 (номинальный)
* Тяжесть: доброкачественная=0 или злокачественная=1 (биноминальный).

На рисунке 1 представлена визуализация данных с помощью t-SNE.

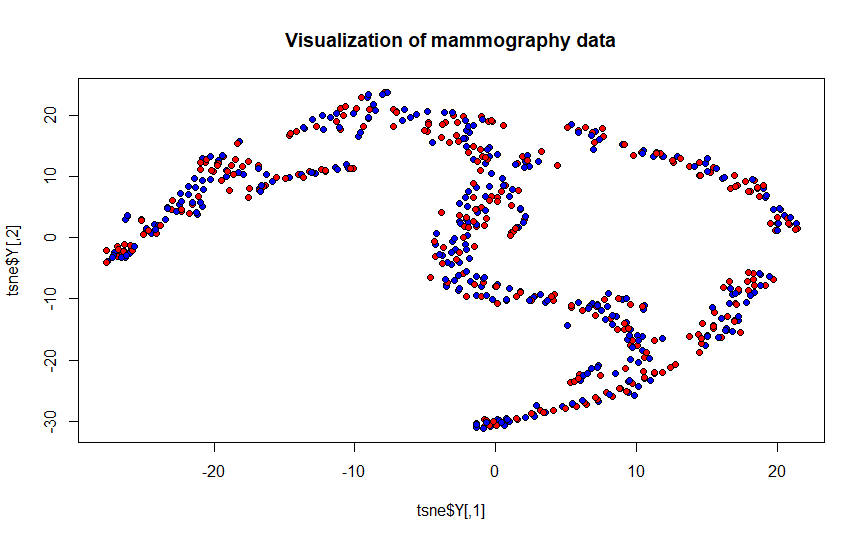


Рисунок 1 - Визуализация данных с помощью метода t-SNE

Красными точками изображены злокачественные случаи, синим – доброкачественные.

## **Разработка классификаторов**

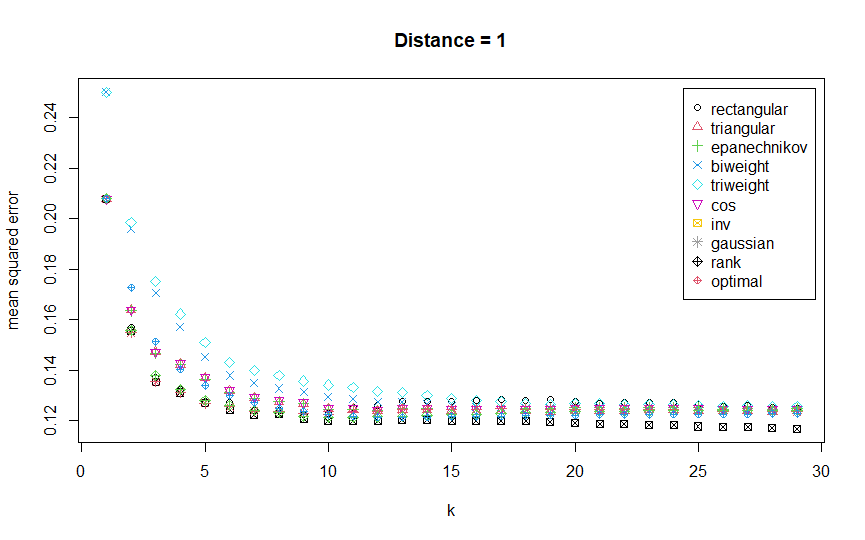
Были разработаны классификаторы с помощью трех методов – k ближайших соседей, бустинга и беггинга.

**Метод k ближайших соседей.**

Была проведена классификация методом k ближайших соседей c разным расстоянием Минковского: 1, 3 и 5, а также с разными типами ядер, максимальное значение параметра k – корень из количества записей в наборе данных.

Результаты представлены на рисунках 2 – 4.

Наилучший тип ядра – инверсивное, минимальная среднеквадратичная ошибка равна 0.1169409, при этом k = 29.



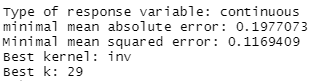
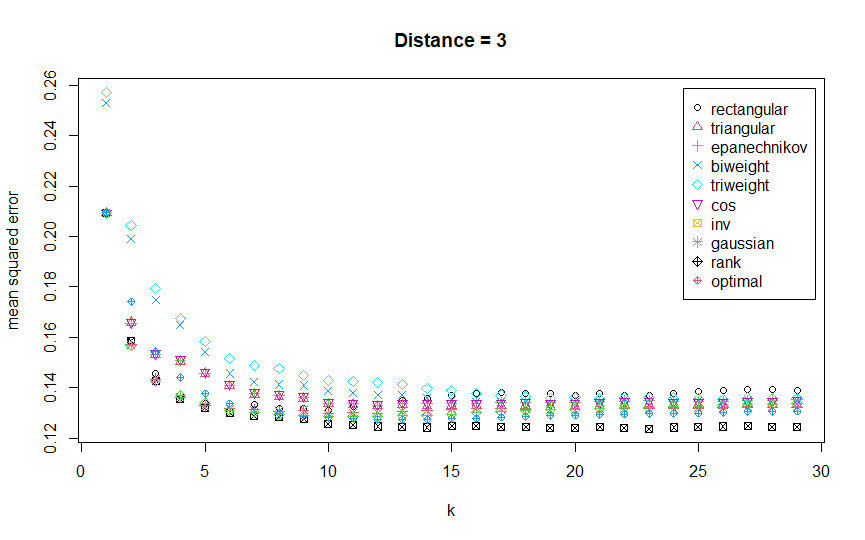


Рисунок 2 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 1



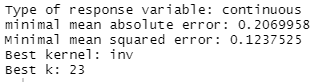
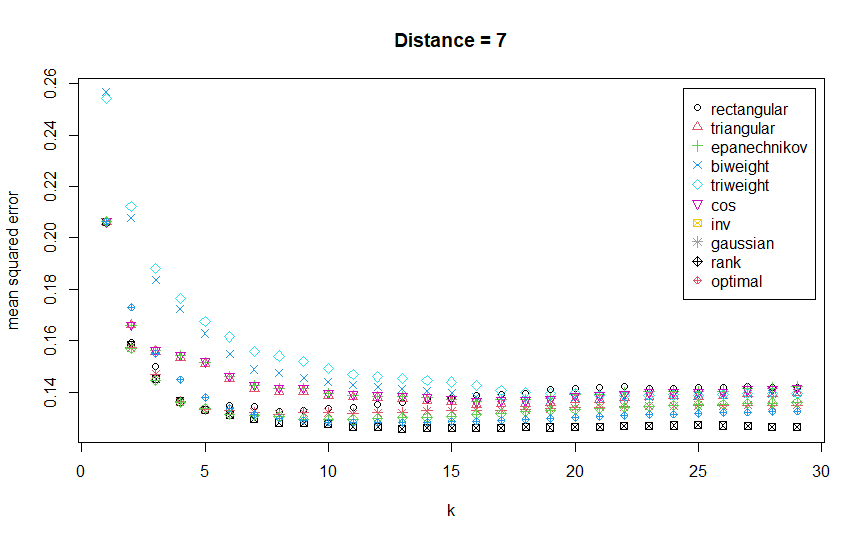


Рисунок 3 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 3



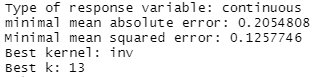


Рисунок 4 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 7

**Беггинг.**

Классификатор с алгоритмом беггинга запускался с различным количеством слабых классификаторов (деревьев) – от 1 до 201 с шагом 10. Результаты приведены на рисунке 5.

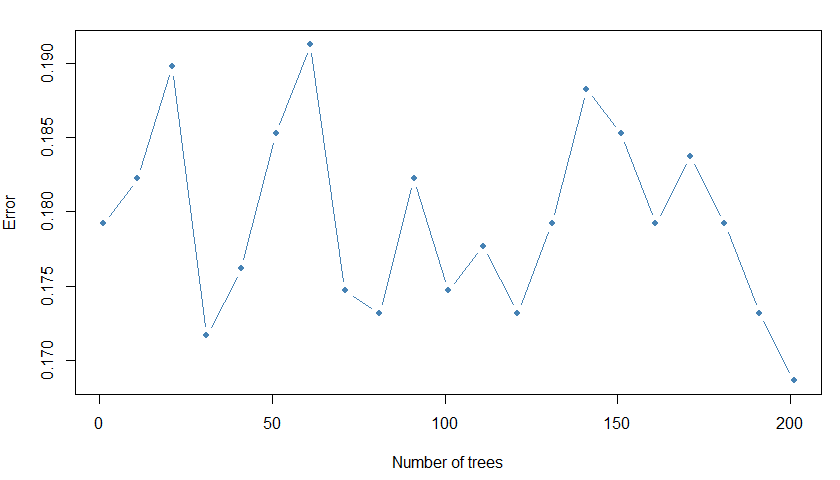


Рисунок 5 – Зависимость ошибки классификации от количества деревьев (метод bagging)

Наименьшая ошибка классификации (0,0967) обнаружилась при количестве деревьев, равном 201.

**Бустинг.**

Классификатор с алгоритмом бустинга запускался с различным количеством слабых классификаторов (деревьев) – от 1 до 201 с шагом 10. Результаты приведены на рисунке 6.

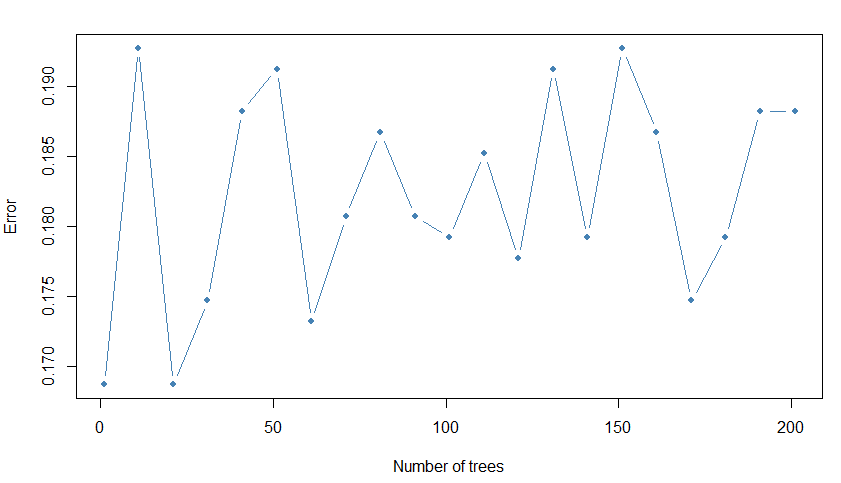


Рисунок 6 – Зависимость ошибки классификации от количества деревьев (метод boosting)

Наименьшая ошибка классификации (0,0845) обнаружилась при количестве деревьев, равном 1 и 21.

С помощью полученных результатов можно сделать вывод, что ошибки, полученные при классификации с помощью методов bagging и boosting, в среднем примерно одинаковые.

## **Кластеризация данных**

Была проведена кластеризация набора данных методом k-средних. Алгоритм был запущен 100 раз, в результате чего была получена ошибка кластеризации, равная 0,3144.

Кроме того, была построена дендрограмма набора данных (представлена на рисунке 7). Данные не очень схожи, можно сделать вывод, что именно поэтому получилась не очень большая ошибка кластеризации.

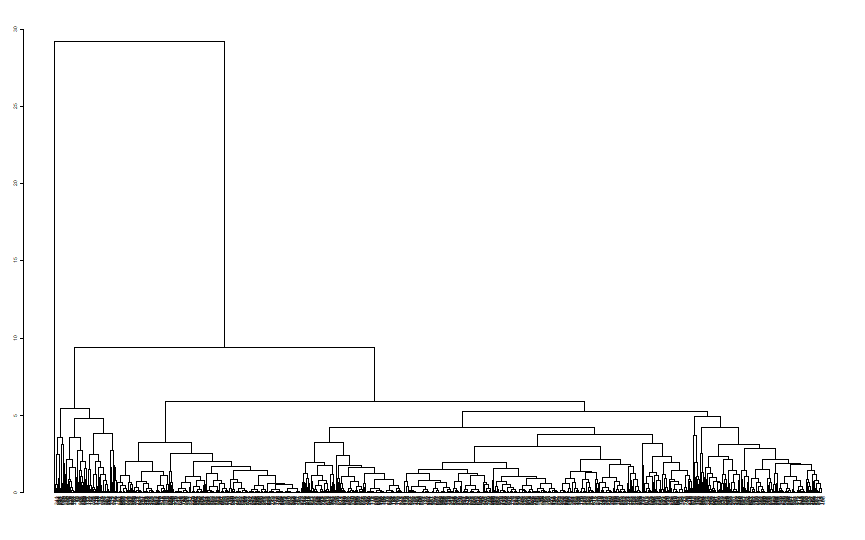


Рисунок 7 – Дендрограмма набора данных

## **Определение наиболее значимых признаков**

Наиболее значимые признаки для набора данных были определены с помощью метода Лассо.

Выделение наиболее информативных признаков – достаточно важная задача, так как присутствие признаков, вносящих малый вклад в результат, может снизить точность модели.

Для оценки значимости признаков был использован метод glmnet (принимает такие аргументы, как матрица данных, значения классов, alpha = 1 – метод Лассо, family = “binomial” – задача бинарной классификации).

Порядок признаков согласно их значимости:

* BI-RADS
* Shape
* Margin
* Age
* Density

## **Использование автокодера**

Для реализации автокодеров использовалась библиотек h2o (h2o.deeplearning(), где autoencoder = TRUE).

**Автокодер, сокращающий размерность**

Для сокращения размерности был добавлен скрытый слой из трех узлов, при создании автокодера была использована функция активации Rectifier по умолчанию.

Визуализация данных меньшей размерности представлена на рисунке 8. Видно, что разделение на классы по полученной визуализации получить довольно трудно.

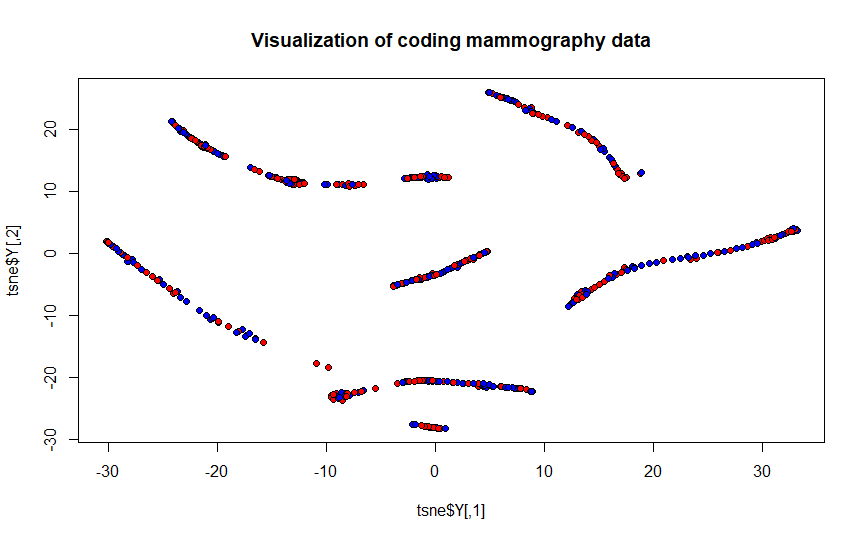
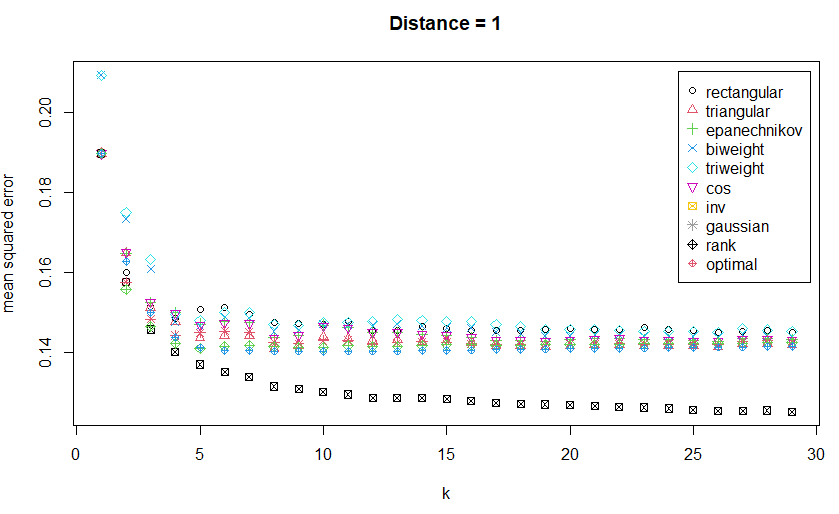


Рисунок 8 – Визуализация данных меньшей размерности

С помощью метода k ближайших соседей была выполнена классификация данных меньшей размерности, результаты представлены на рисунках 9 – 11.

Наилучший тип ядра – инверсивное, минимальная среднеквадратичная ошибка равна 0,1251519, при этом k = 29.



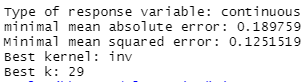
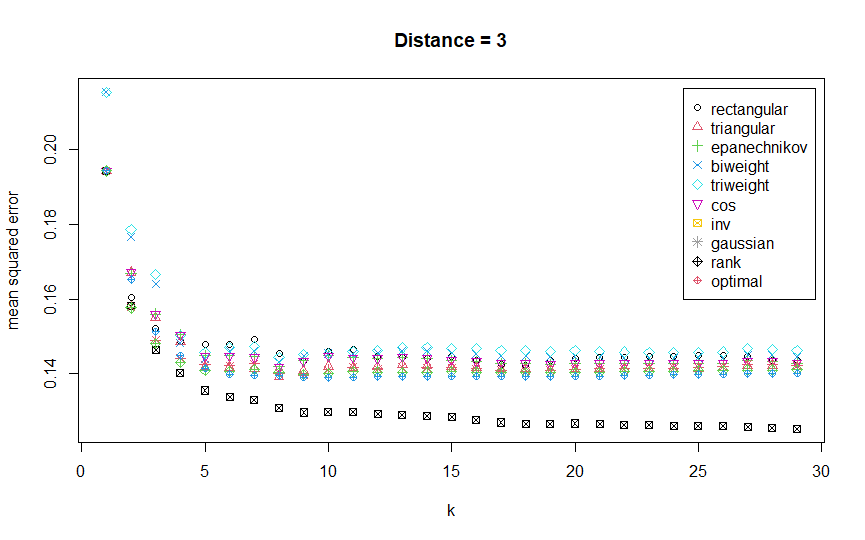


Рисунок 9 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 1



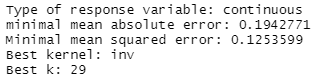
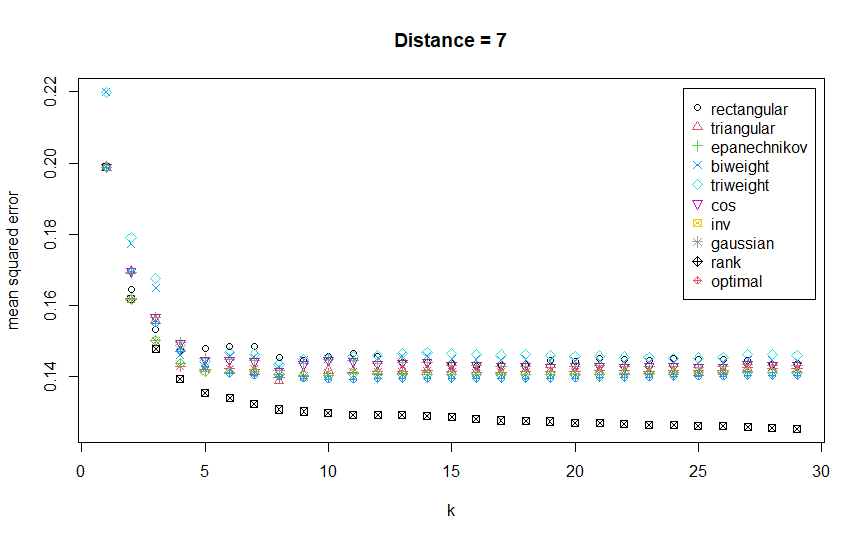


Рисунок 10 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 3



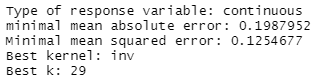


Рисунок 11 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 7

**Автокодер для реализации разреженного скрытого слоя**

Для того, чтобы извлечь наиболее важные представления функции, используются разреженные автокодеры. Разреженность – некоторое количество нейронов перестает быть активность, их веса обнуляются, то есть разреженный автокодер представляет каждый входные данные как комбинацию меньшего числа активаций.

Данный автокодер отличается от предыдущего большим количеством нейронов в скрытом слое (было увеличено до 100).

Визуализация полученных данных представлена на рисунке 12. Видно, что разделение на классы по полученной визуализации получить довольно трудно.

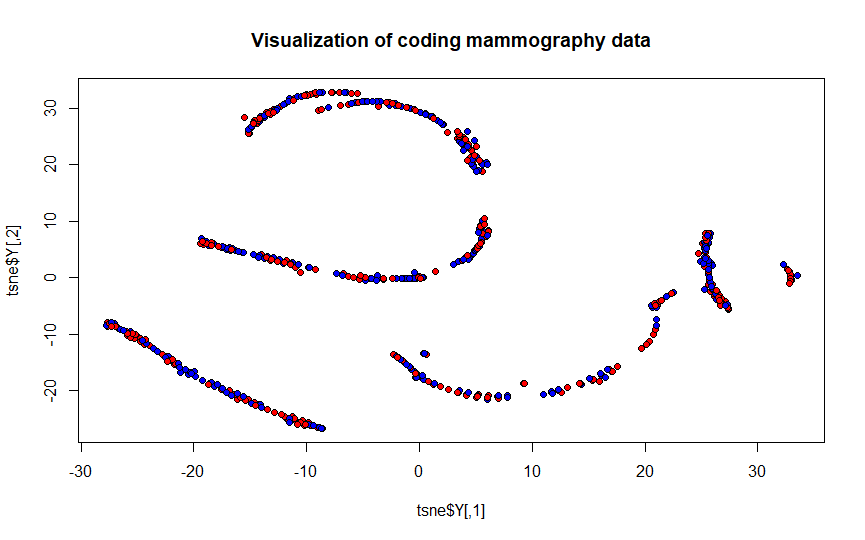
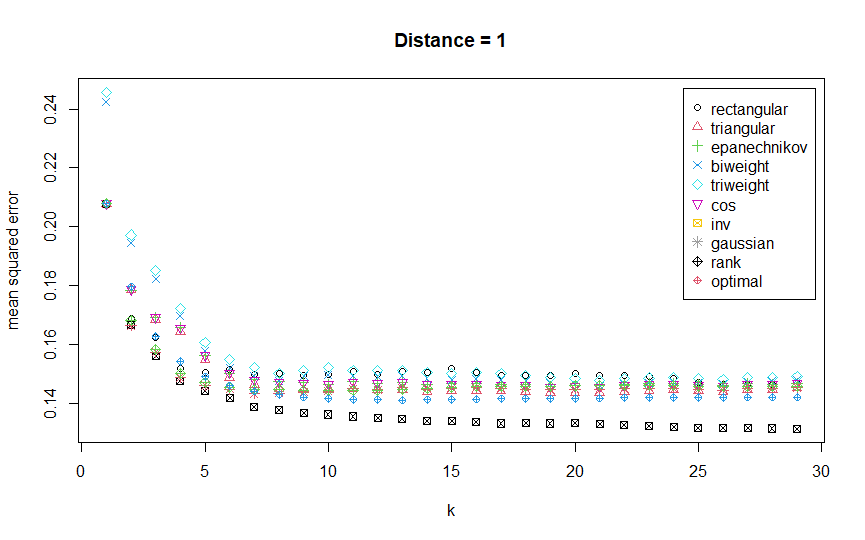


Рисунок 12 – Визуализация полученных данных

С помощью метода k ближайших соседей была выполнена классификация данных, результаты представлены на рисунках 13 – 15.

Наилучший тип ядра – инверсивное, минимальная среднеквадратичная ошибка равна 0,1313169, при этом k = 27.



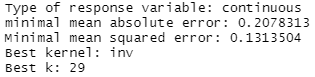
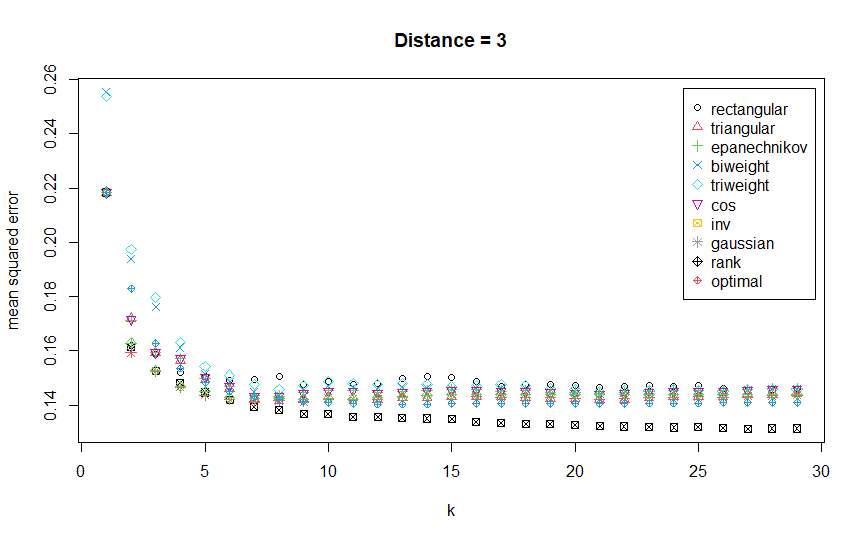


Рисунок 13 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 1



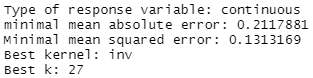
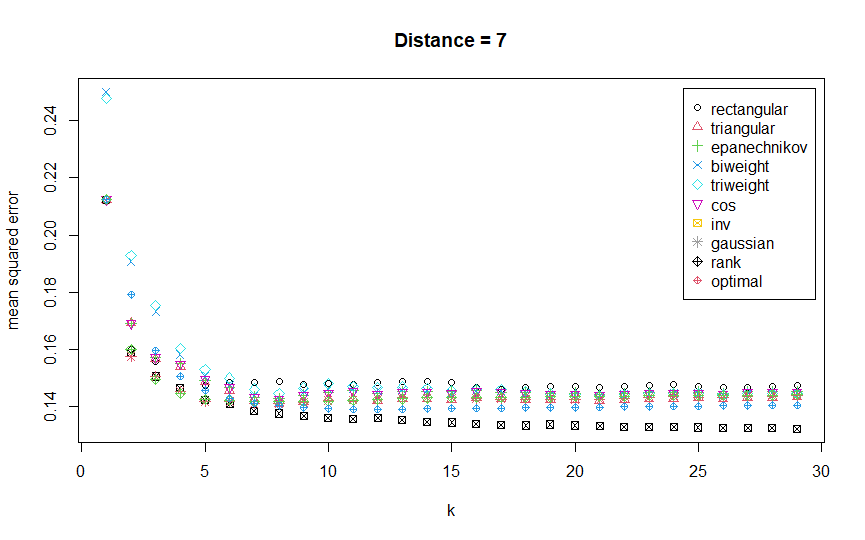


Рисунок 14 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 3



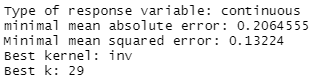


Рисунок 15 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 7

**Зашумленный автокодер**

Зашумленный (шумоподавляющий) автокодер – это такая модель автокодера, которая восстанавливает входные данные из зашумленной копии входных данных.

Обучение автокодера отличается от обучения предыдущих автокодеров тем, что в training\_frame подаются зашумленные данные, а в validation\_frame – первоначальные входные данные.

Визуализация полученных данных представлена на рисунке 16. Видно, что разделение на классы по полученной визуализации получить довольно трудно.

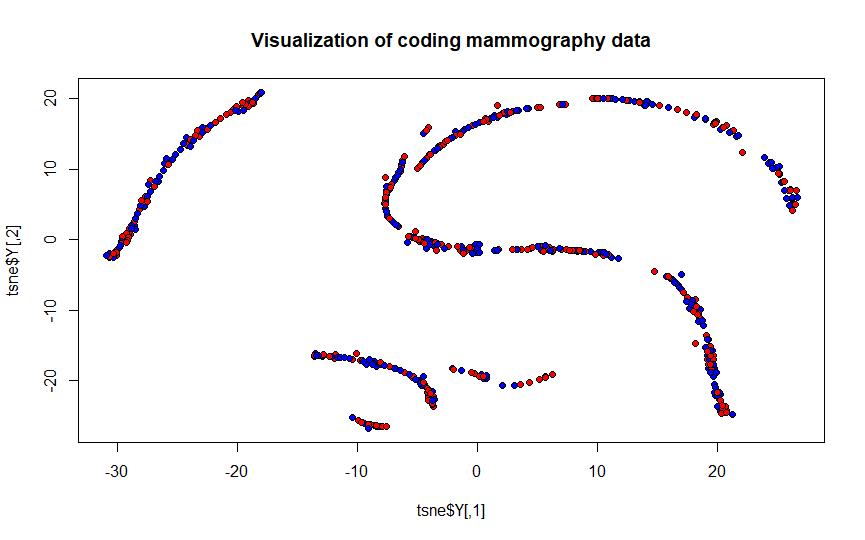
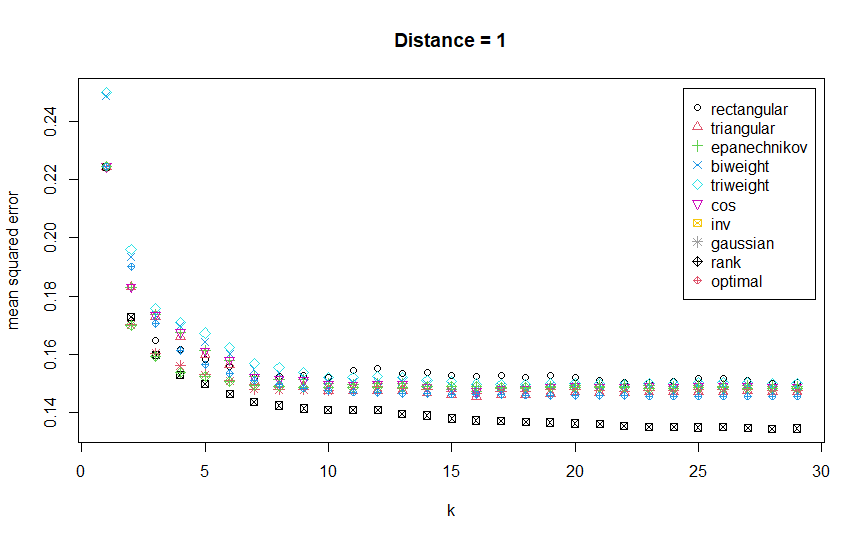


Рисунок 16 – Визуализация полученных данных

С помощью метода k ближайших соседей была выполнена классификация данных, результаты представлены на рисунках 17 – 19.

Наилучший тип ядра – инверсивное, минимальная среднеквадратичная ошибка равна 0,1345594, при этом k = 28.



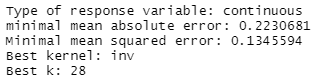
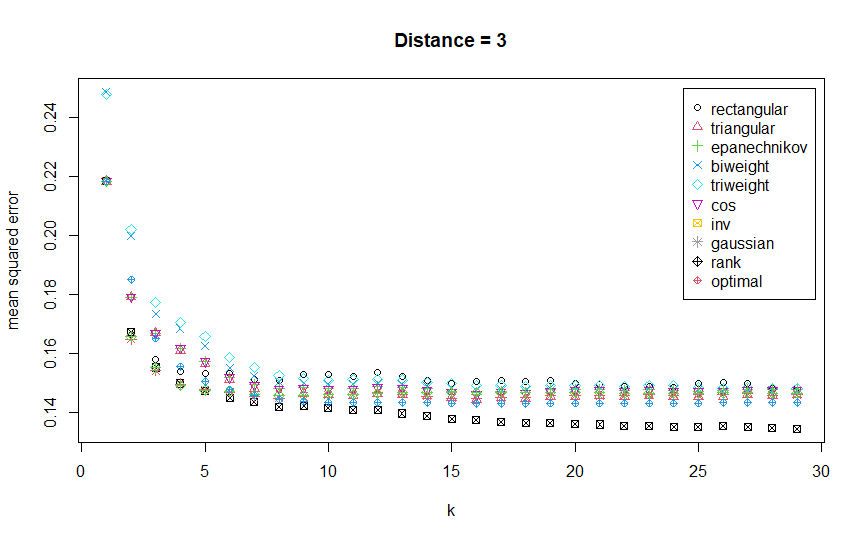


Рисунок 17 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 1



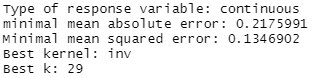
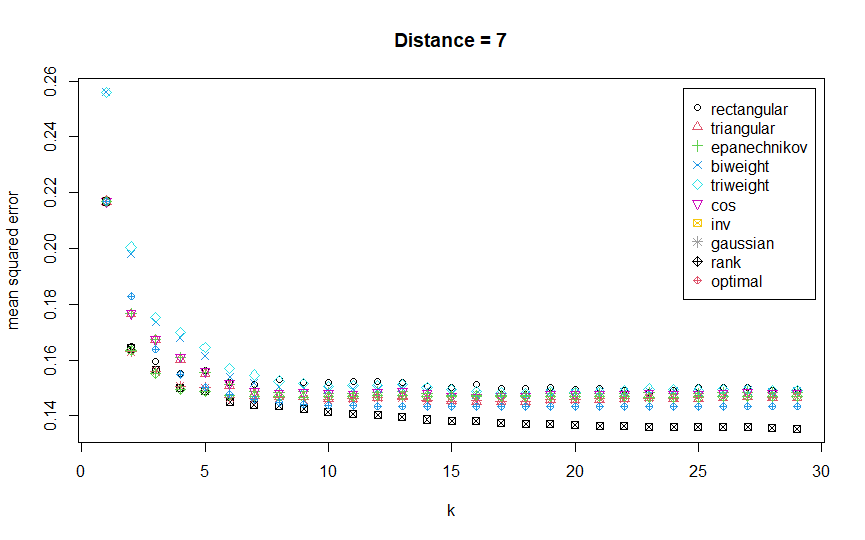


Рисунок 18 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 3



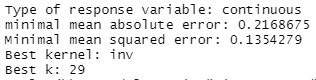


Рисунок 19 – Зависимость среднеквадратичной ошибки от типа ядра и параметра k, distance = 7

При сравнении полученных результатов автокодеров, можно сказать, что лучшие результаты показал автокодер для сокращенной размерности, но эти результаты несколько хуже показаний исходной модели k ближайших соседей.

# **Вывод**

При выполнении данной курсовой работы для базы данных «Mammographic masses»:

* Были реализованы 3 классификатора, основанные на методах k ближайших соседей, беггинге и бустинге. Наилучшие результаты показал бустинг (ошибка классификации - 0,0845).
* Был реализован метод k-средних - обучения без учителя, ошибка кластеризации – 0,3144.
* Были определены наиболее значимые признаки набора данных с помощью метода Лассо.
* Были релизованы три автокодера – сокращения размерности, «разрежеватель» и шумоподавляющий. Наилучшие результаты показал автокодер для сокращенной размерности (ошибка - 0,1251519).
* Была осуществлена визуализация набора данных.

# **Приложение 1**

library(Rtsne)

library(kknn)

library(mlbench)

library(adabag)

library(rpart)

library(cluster)

library(glmnet)

col\_names=c("BI-RADS","Age","Shape","Margin","Density","Severity")

mammography<-read.table("D:\\mammographic\_masses\_data.txt", sep=',', header=F,col.names=col\_names,na.strings='?')

mammography

#remove rows with NA value in any column data frame

mammography<-mammography[complete.cases(mammography),]

mammography

strings\_num<-dim(mammography)[1]

data<-mammography[order(runif(strings\_num)),]

data

tsne<-Rtsne(as.matrix(data),pca=TRUE,dim=2,check\_duplicates = FALSE)

plot(tsne$Y,pch=21,bg=c("Blue","Red"),main="Visualization of mammography data")

#knn

train\_data<-data[1:(0.8\*strings\_num),]

test\_data<-data[(0.8\*strings\_num+1):strings\_num,]

ker = c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal")

kknn\_model1<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=1)

kknn\_model2<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=3)

kknn\_model3<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=7)

kknn\_model1

plot(kknn\_model1, main="Distance = 1")

kknn\_model2

plot(kknn\_model2, main="Distance = 3")

kknn\_model3

plot(kknn\_model3, main="Distance = 7")

#bagging

train\_data<-data[1:(0.8\*strings\_num),]

test\_data<-data[((0.8\*strings\_num)+1):strings\_num,]

train\_data

test\_data

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

misClass <- c()

it\_num <- 4

k=1

for (i in trees\_num)

{

misCl <- c()

for(j in 1:it\_num)

{

bagging\_model <- bagging(Severity ~ ., data = train\_data, mfinal = i)

pr <- predict(bagging\_model, test\_data)

misCl[j] <- pr$error

}

misClass[k] <- mean(misCl)

k=k+1

}

mean(misClass)

misClass

print(min(misClass))

plot(seq(1, 201, 10), misClass, col = "steelblue", xlab="Number of trees", ylab="Error", pch = 20, cex = 1.2, type="b")

#boosting

train\_data<-data[1:(0.8\*strings\_num),]

test\_data<-data[(0.8\*strings\_num+1):strings\_num,]

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

misClass <- c()

it\_num <- 4

k=1

for (i in trees\_num)

{

misCl <- c()

for(j in 1:it\_num)

{

boosting\_model <- boosting(Severity ~ ., train\_data, mfinal = i)

pr <- predict(boosting\_model, test\_data)

misCl[j] <- pr$error

}

misClass[k] <- mean(misCl)

k=k+1

}

mean(misClass)

misClass

print(min(misClass))

plot(seq(1, 201, 10), misClass, col = "steelblue", xlab="Number of trees", ylab="Error", pch = 20, cex = 1.2, type="b")

#k-means

cluster\_data<-data[,-6]

model<-kmeans(cluster\_data, 2, iter.max = 100)

model$cluster[model$cluster == 1] <- 0

model$cluster[model$cluster == 2] <- 1

misCl = 0

for (i in 1:strings\_num)

{

if (model$cluster[i] != data$Severity[i])

misCl = misCl + 1

#print(model$cluster[i])

#print(data$Severity[i])

}

misCl<-misCl/strings\_num

misCl

plot(agnes(cluster\_data))

matr <- dist(scale(cluster\_data))

hc <- hclust(matr)

hcd <- as.dendrogram(hc)

plot(hcd, cex = 0.7)

#main features

X<-as.matrix(data[,-6])

Y<-data[,6]

glm<-glmnet(X,Y,family="binomial",nlambda =97,alpha =1)

res<-as.matrix(glm$beta)

res

#autocoder 1

library(h2o)

h2o.init()

features <- as.h2o(data[-6])

ae <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = features,

autoencoder = TRUE,

hidden = 3, # 6 to 3 features

activation = 'Rectifier'

)

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

ae\_codings

data\_2 <- as.data.frame(ae\_codings)

data\_2$Severity <- data$Severity

tsne<-Rtsne(as.matrix(data\_2),pca=TRUE,dim=2,check\_duplicates = FALSE)

plot(tsne$Y,pch=21,bg=c("Blue","Red"),main="Visualization of coding mammography data")

strings\_num <- dim(data\_2)[1]

rand\_data <- data\_2[ order(runif(strings\_num)),]

train\_data<-rand\_data[1:(0.8\*strings\_num),]

test\_data<-rand\_data[(0.8\*strings\_num+1):strings\_num,]

ker = c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal")

kknn\_model1<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=1)

kknn\_model2<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=3)

kknn\_model3<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=7)

kknn\_model1

plot(kknn\_model1, main="Distance = 1")

kknn\_model2

plot(kknn\_model2, main="Distance = 3")

kknn\_model3

plot(kknn\_model3, main="Distance = 7")

#autocoder 2

library(h2o)

h2o.init()

features <- as.h2o(data[-6])

ae <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = features,

autoencoder = TRUE,

hidden = 100, # 6 to 100 features

activation = 'Rectifier'

)

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

ae\_codings

data\_2 <- as.data.frame(ae\_codings)

data\_2$Severity <- data$Severity

tsne<-Rtsne(as.matrix(data\_2),pca=TRUE,dim=2,check\_duplicates = FALSE)

plot(tsne$Y,pch=21,bg=c("Blue","Red"),main="Visualization of coding mammography data")

strings\_num <- dim(data\_2)[1]

rand\_data <- data\_2[ order(runif(strings\_num)),]

train\_data<-rand\_data[1:(0.8\*strings\_num),]

test\_data<-rand\_data[(0.8\*strings\_num+1):strings\_num,]

ker = c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal")

kknn\_model1<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=1)

kknn\_model2<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=3)

kknn\_model3<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=7)

kknn\_model1

plot(kknn\_model1, main="Distance = 1")

kknn\_model2

plot(kknn\_model2, main="Distance = 3")

kknn\_model3

plot(kknn\_model3, main="Distance = 7")

#autocoder 3

library(h2o)

h2o.init()

features <- as.h2o(data[-6])

noise\_features <- as.h2o(data[-6] + rnorm(nrow(data), mean = 0, sd = 0.1))

denoise\_ae <- h2o.deeplearning(

x = seq\_along(features),

training\_frame = noise\_features,

validation\_frame = features,

autoencoder = TRUE,

hidden = 60,

activation = 'Rectifier',

sparse = TRUE

)

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

ae\_codings

data\_2 <- as.data.frame(ae\_codings)

data\_2$Severity <- data$Severity

tsne<-Rtsne(as.matrix(data\_2),pca=TRUE,dim=2,check\_duplicates = FALSE)

plot(tsne$Y,pch=21,bg=c("Blue","Red"),main="Visualization of coding mammography data")

strings\_num <- dim(data\_2)[1]

rand\_data <- data\_2[ order(runif(strings\_num)),]

train\_data<-rand\_data[1:(0.8\*strings\_num),]

test\_data<-rand\_data[(0.8\*strings\_num+1):strings\_num,]

ker = c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal")

kknn\_model1<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=1)

kknn\_model2<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=3)

kknn\_model3<-train.kknn(Severity ~., train\_data, kmax=round(sqrt(strings\_num),digits=0), kernel=ker, distance=7)

kknn\_model1

plot(kknn\_model1, main="Distance = 1")

kknn\_model2

plot(kknn\_model2, main="Distance = 3")

kknn\_model3

plot(kknn\_model3, main="Distance = 7")