Министерство образования и науки РФ
Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

КУРСОВОЙ ПРОЕКТ

«Разработка классификаторов для баз данных»
по дисциплине «Машинное обучение»

Выполнила:
студентка гр. 3540201/20301 _____ Климова О. А.
подпись, дата

Проверил:
д.т.н., проф. _____ Уткин Л. В.
подпись, дата

Санкт-Петербург
2022

Содержание

Постановка задачи

В данной курсовой работе 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).

1 Описание датасета

Маммограмма – метод скрининга рака молочной железы. В данной курсовой работе будет использоваться датасет из 961 элемента с 6 параметрами, который может быть использован для диагности доброкачественного или злокачественного образования:

1. BI-RADS: оценка от 1 до 5
2. Age: возраст пациента в годах (целое число)
3. Shape: форма: круглая = 1 овальная = 2 дольчатая = 3 неправильная = 4
4. Margin: граница массы: ограниченная = 1 микродольчатая = 2 скрытая = 3 нечетко очерченная = 4 спикулированная = 5
5. Density: плотность: высокая = 1 изо = 2 низкая = 3 жиросодержащая = 4
6. Severity: серьезность: доброкачественная = 0 или злокачественная = 1

Используемое программное обеспечение: R-Studio.

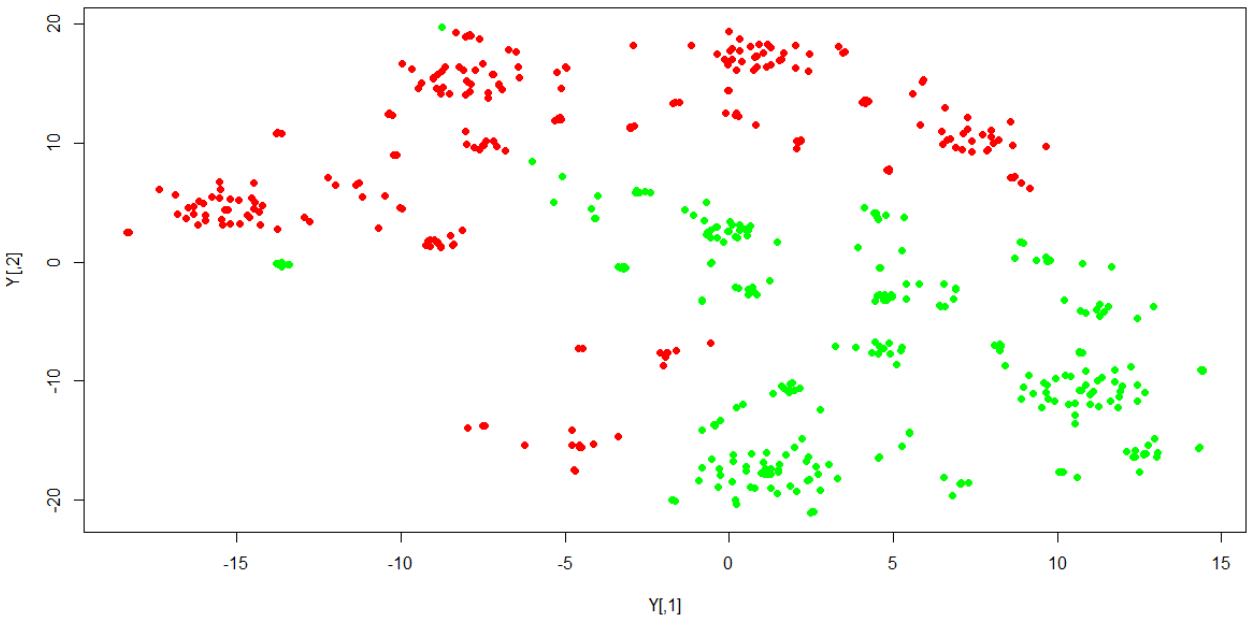
2 Визуализация данных

Датасет имеет следующий вид:

	V1	V2	V3	V4	V5	V6
1	5	67	3	5	3	1
2	4	43	1	1	?	1
3	5	58	4	5	3	1
4	4	28	1	1	3	0
5	5	74	1	5	?	1
6	4	65	1	?	3	0
7	4	70	?	?	3	0
8	5	42	1	?	3	0
9	5	57	1	5	3	1
10	5	60	?	5	1	1

Некоторые значения параметров в датасете пропущены, поэтому они были заменены на средние значения по столбцам. Также, для визуализации данных при помощи метода t-SNE необходимо, чтобы не было повторений, поэтому с помощью функции distinct() было выбрано уникальное подмножество строк. Также, данные столбцов V1, V2, V3, V4, V5 были приведены к типу integer.

Визуализация данных методом t-SNE:



Зеленые – доброкачественные

Красные – злокачественные

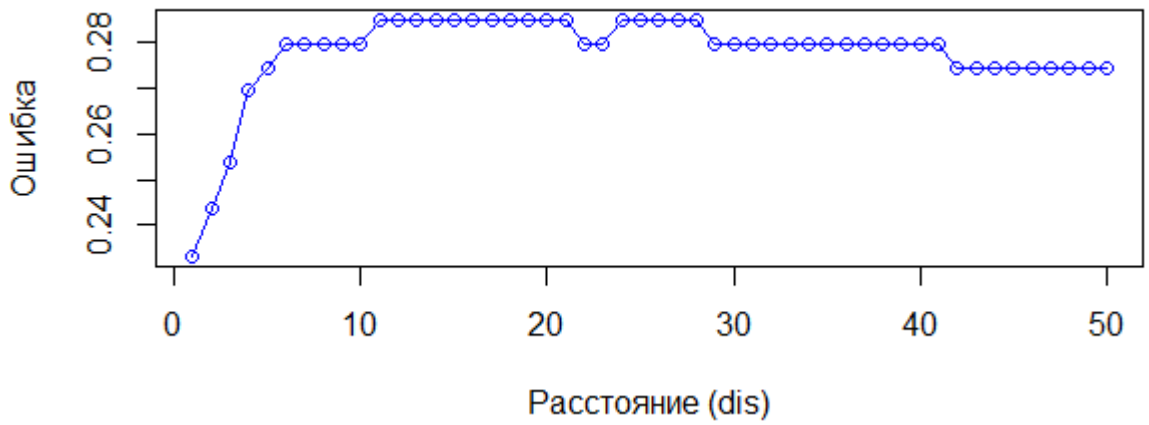
3 Построение классификаторов

1) Был разработан наивный байесовский классификатор, который на тестовых данных (20% от общей выборки) продемонстрировал точность 82.9% со следующими полученными значениями:

bayes_predict	0	1
0	84	18
1	15	76

2) Был разработан классификатор на основе k ближайших соседей с параметрами k – количеством соседей, dis – расстоянием и kernel – ядром. Для получения наилучшего результата параметры были подобраны с целью минимизации ошибки.

- Оптимальное k = 3 при kernel = " triangular".
- Оптимальное dis = 1 при k = 3 и kernel = " triangular":



- Рассмотрев другие ядра при k = 3 и при dis = 1 было получено, что минимальную ошибку дает ядро cos и epanechnikov:

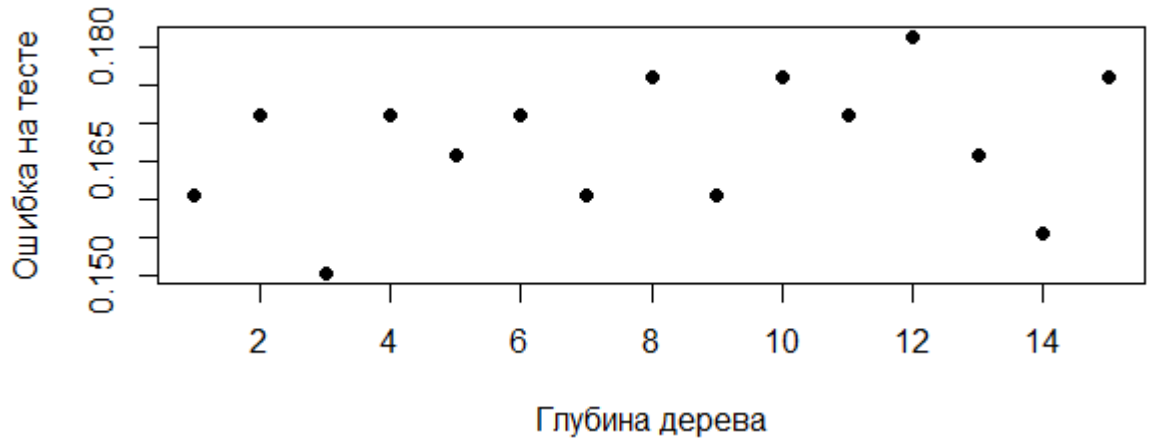
	knn_kernel	knn_err
1	rectangular	0.2383420
2	triangular	0.2331606
3	epanechnikov	0.2279793
4	biweight	0.2383420
5	triweight	0.2538860
6	cos	0.2279793
7	inv	0.2538860
8	gaussian	0.2383420
9	rank	0.2383420
10	optimal	0.2538860

При выборе параметров k = 3, dis = 1 и kernel = cos на тестовых данных была получена точность 82.4%:

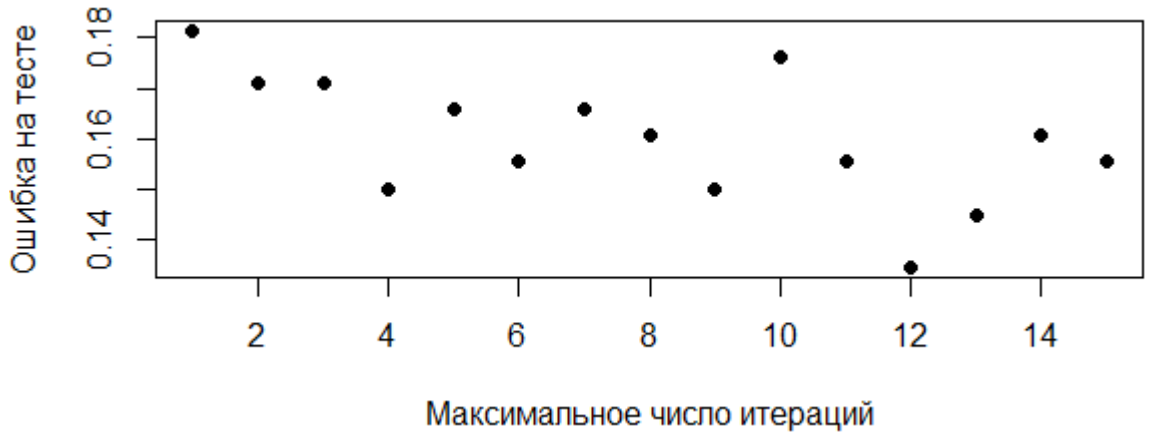
```
[1] 0.8238342
> knn_tab

    0  1
0 92 27
1  7 67
```

3) Был осуществлен бэггинг, наименьшая ошибка классификации была достигнута при параметрах maxdepth = 4 и mfinal = 12.

Зависимость тестовой ошибки от числа деревьев в ансамбле при фиксированном mfinal = 1 имеет вид:



Значит, оптимальной является глубина maxdepth = 3, при данной глубине был построен график зависимости ошибки от максимального числа итераций:



По полученному графику можно сделать вывод, что оптимальное максимальное число итераций mfinal = 12.

При выбранных параметрах беггинга была получена точность 86.5%

```
> bagging_acuracy
[1] 0.865285
```

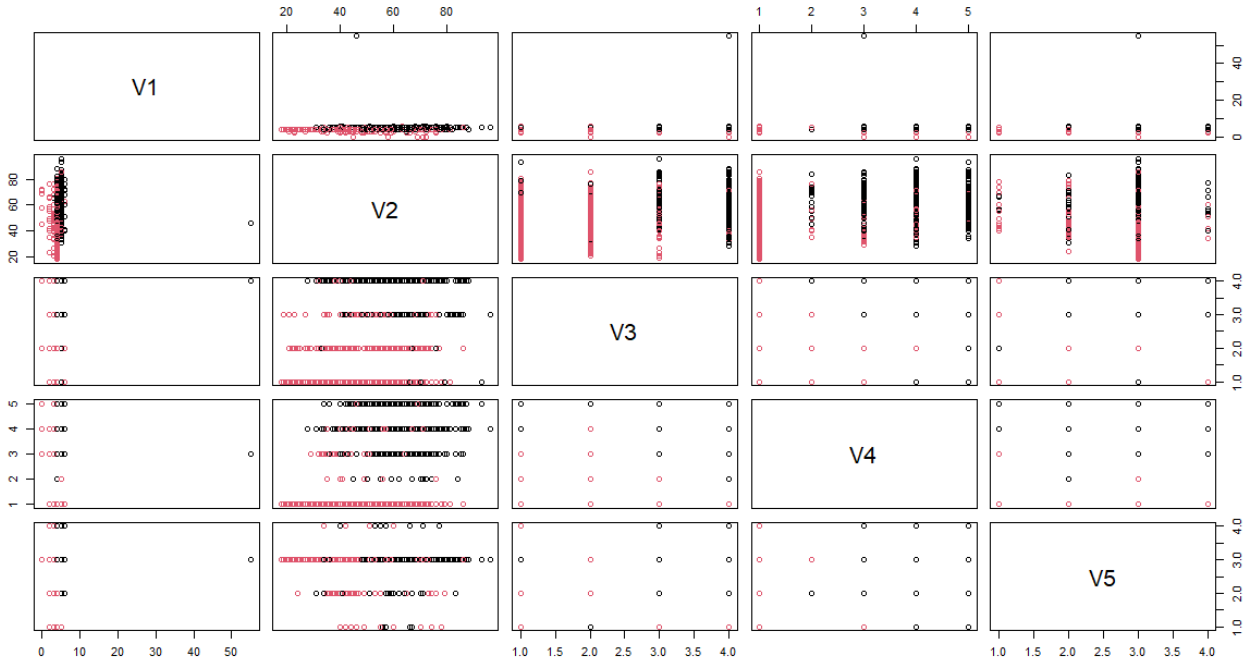
Таким образом, можно сделать вывод, что оптимальнее всего для рассматриваемого датасета работает классификатор на основе беггинга, он дает наименьшую ошибку на тесте – 13.5%.

4 Кластеризация

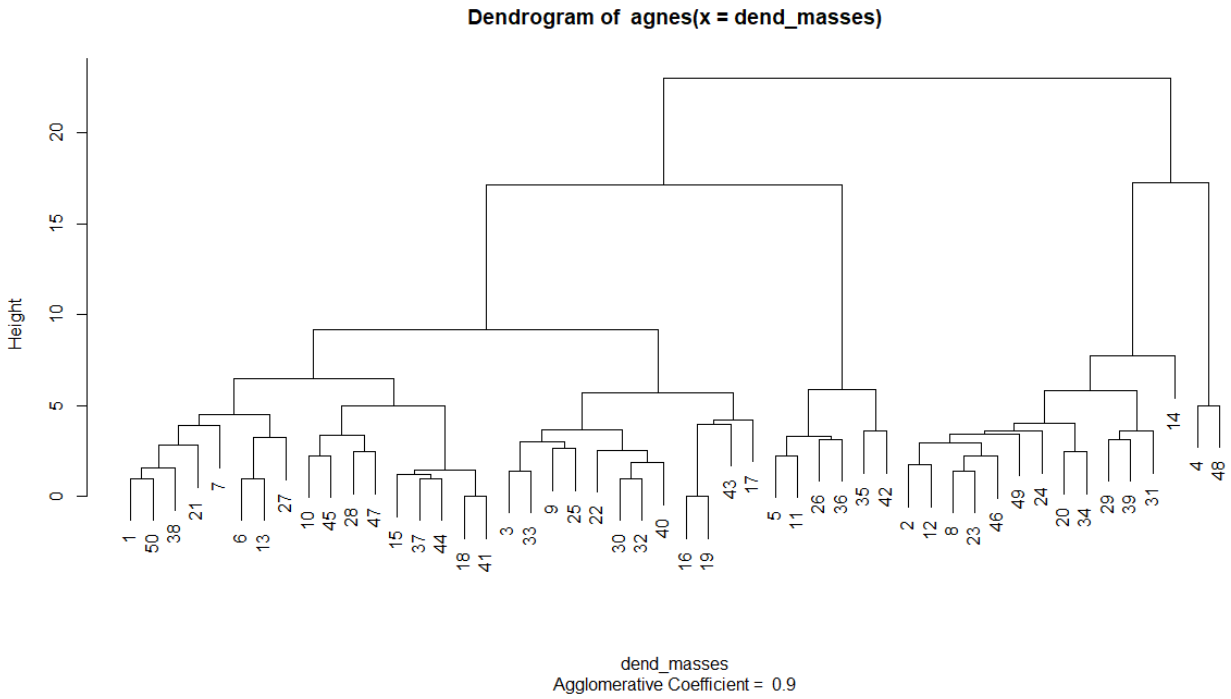
Из базы были удалены метки классов и осуществлена кластеризация данных методом k-медоидов. Параметр k = 2, так как необходимо выделить два кластера. Параметры метрики и использования стандартизации были подобраны путем исследования ошибки при их различных сочетаниях:

1) При metric = "euclidean", stand = FALSE: 34.3% ошибки
2) При metric = "euclidean", stand = TRUE: 17.6% ошибки
3) При metric = "manhattan", stand = FALSE: 30.1% ошибки
4) При metric = "manhattan", stand = TRUE: 18.2% ошибки

Таким образом, можно сделать что для кластеризации оптимальнее всего использовать k = 2, metric = "euclidean", stand = TRUE:

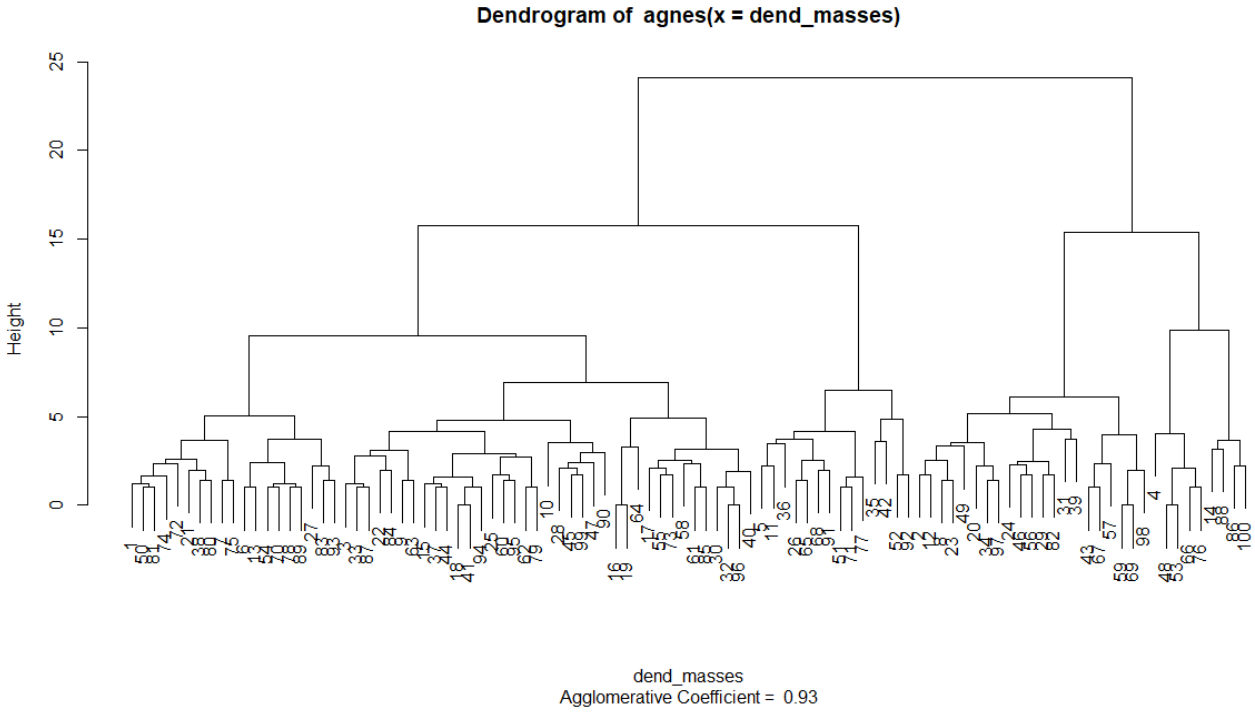


Так как датасет большой, дендрограмма была построена для 50 элементов. По полученной дендрограмме можно видеть, что 35 элементов вошли в один кластер, а 15 в другой.



Полученные результаты были сравнены с реальными метками и была получена ошибка 46%.

Если взять массив для нендрограммы больше, например, 100 элементов, то 69 элементов войдут в один кластер, а 31 в другой:



И ошибка кластеризации составит 41%.

5 Определение наиболее значимых признаков

Наиболее значимый признак – признак, в наибольшей степени влияющий на отнесение объектов к определенному классу. Определение таких признаков будет производиться, используя логистическую регрессию в рамках метода Лассо.

Для выполнения задачи был использован стандартный метод glmnet, в качестве аргументов которого передавались матрица обучаемых данных x и вектор меток y, а также параметр alpha = 1, который означает использовние метода Лоссо.

Наиболее значим по методу Лассо является признак V4 – граница массы, а наименее значимыми V1 и V2 – оценка BI-RADS и плотность:

```
V1          .
V2          0.002968712
V3          0.074914015
V4          0.061609288
V5          .
```

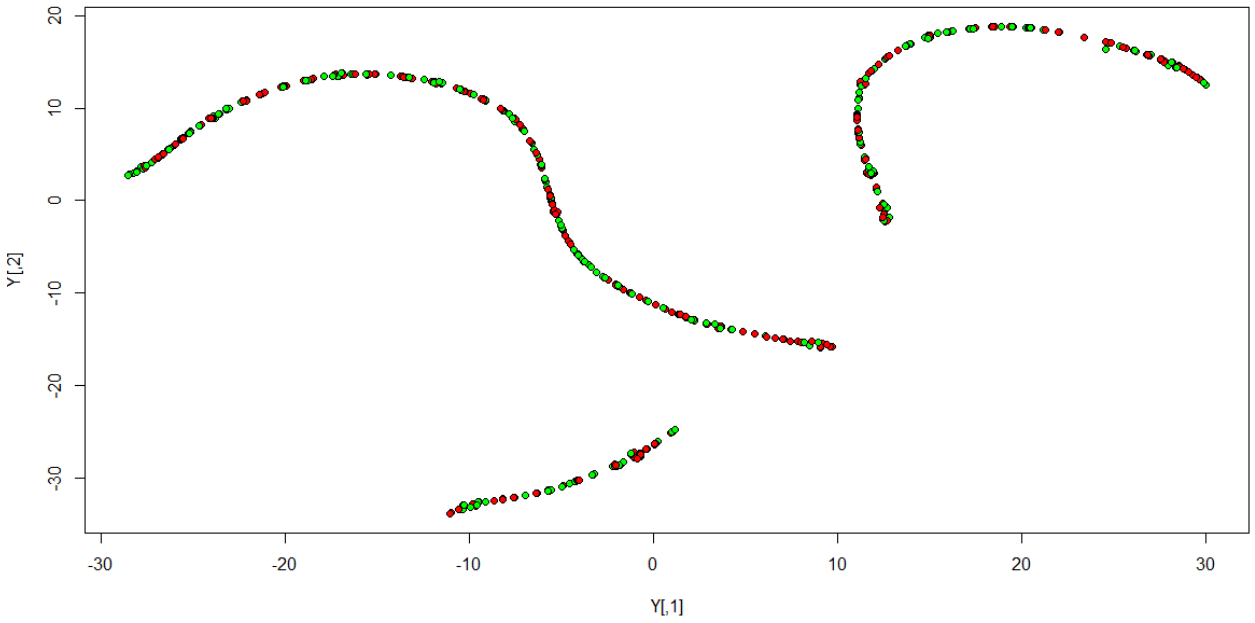
6 Использование автокодера

1) Использование автокодера для сокращения размерности

Выбранные параметры автокодера:

1. Число слоев – 3 (input, hidden, output)
2. Число нейронов в скрытом слое – 4
3. Используемая функция активации – tanh.
4. Погрешность eps = 0

Визуализация при помощи TSNE будет иметь вид:



После преобразования выборки с помощью автокодера была проведена классификация методом k ближайших соседей с параметрами: k = 3; dis = 1 и kernel = " cos". При использовании данного классификатора была получена ошибка для сокращенной размерности: 74%.

2) Использование автокодера для реализации разреженного скрытого слоя.

Выбранные параметры автокодера:

1. Число слоев – 3 (input, hidden, output)
2. Число нейронов в скрытом слое – 20
3. Используемая функция активации – tanh.
4. Погрешность eps = 0

Ошибка для разряженной размерности составила: 71%

3) Использование автокодера для реализации разреженного скрытого слоя.

Выбранные параметры автокодера:

1. Число слоев – 3 (input, hidden, output)
2. Число нейронов в скрытом слое – 20
3. Используемая функция активации – tanh.
4. Погрешность eps = 0.5

Ошибка для разряженной размерности с зашумлением составила: 64%

Визуализация для разряженного и зашумленного автокодера дает, примерно, такую же картину, как и визуализация при автокодере с сокращенной размерностью.

Оценивая полученные результаты, можно сделать вывод, что для данного датасета лучше работать с исходными данными (точность классификации намного выше).

Заключение

В ходе данной курсовой работы:

1) Были разработаны 3 классификатора и осуществлена настройка их параметров для минимизации ошибки классификации на тестовых данных. Также была выполнена визуализацию данных при помощи метода t-SNE.
2) Используемые классификаторы были оценены по полученным точностям классификации и был сделан выбор относительно наилучшего способа классификации.
3) Из базы были удалены метки классов и осуществлена кластеризация данных, была построена дендограмму. Полученные значения были сравнены с реальными метками данных и был сделан вывод о размере ошибки кластеризованных данных.
4) С использованием логистической регрессии в рамках метода Лассо, были определены наиболее значимые признаки, влияющие на отнесение объектов к определенному классу.
5) Был использован автокодер для сокращения размерности и для реализации разреженного скрытого слоя нейронной сети. Преобразовав обучающую выборку при помощи автокодера и осуществив классификацию новых данных, была оценена новая ошибка классификации. Также была выполнена классификация с использованием зашумленного автокодера.

Листинг программы

```
#Загрузка данных и разбиение на обучающую и тестовую выборки
library(e1071)
library(kknn)
library(Rtsne)
library(dplyr)
library(adabag)
library(cluster)
library(glmnet)
library(autoencoder)

masses <- read.table("C:/Users/Unicorn/Desktop/Машинное
Обучение/mammographic_masses.data", sep = ",")

#ПОДГОТОВКА
#_____________________________
#Заполнение пропусков
#Нахождение средних значений по столбцам V1, V2, V3, V4, V5
sum1 = 0
sum2 = 0
sum3 = 0
sum4 = 0
sum5 = 0
for(i in 1:961){
  if(masses$V1[i] == "?"){
  }
  else{
    sum1 = sum1 + as.numeric(masses$V1[i])
  }
}
sum1
V1_mean <- round(sum1/961)

for(i in 1:961){
  if(masses$V2[i] == "?"){
  }
  else{
    sum2 = sum2 + as.numeric(masses$V2[i])
  }
}
sum2
V2_mean <- round(sum2/961)

for(i in 1:961){
  if(masses$V3[i] == "?"){
```

```
  }
  else{
  sum3 = sum3 + as.numeric(masses$V3[i])
  }
}
sum3
V3_mean <- round(sum3/961)
V3_mean
for(i in 1:961){
  if(masses$V4[i] == "?"){
  }
  else{
    sum4 = sum4 + as.numeric(masses$V4[i])
  }
}
sum4
V4_mean <- round(sum4/961)
V4_mean
for(i in 1:961){
  if(masses$V5[i] == "?"){
  }
  else{
    sum5 = sum5 + as.numeric(masses$V5[i])
  }
}
sum5
V5_mean <- round(sum5/961)
V5_mean
#Заполнение
for(i in 1:961){
  if(masses$V1[i] == "?"){
    masses$V1[i]=V1_mean
  }
  if(masses$V2[i] == "?"){
    masses$V2[i]=V2_mean
  }
  if(masses$V3[i] == "?"){
    masses$V3[i]=V3_mean
  }
  if(masses$V4[i] == "?"){
    masses$V4[i]=V4_mean
  }
  if(masses$V5[i] == "?"){
    masses$V5[i]=V5_mean
  }
}
```

```
#masses$V6 = as.factor(masses$V6)
#Сделать остальные параметры в виде чисел
for(i in 1:961){
  masses$V1[i] = as.integer(masses$V1[i])
}
for(i in 1:961){
  masses$V2[i] = as.integer(masses$V2[i])
}
for(i in 1:961){
  masses$V3[i] = as.integer(masses$V3[i])
}
for(i in 1:961){
  masses$V4[i] = as.integer(masses$V4[i])
}
for(i in 1:961){
  masses$V5[i] = as.integer(masses$V5[i])
}
for(i in 1:961){
  masses$V6[i] = as.integer(masses$V6[i])
}
masses$V6 = as.integer(masses$V6)
masses$V1 = as.integer(masses$V1)
masses$V2 = as.integer(masses$V2)
masses$V3 = as.integer(masses$V3)
masses$V4 = as.integer(masses$V4)
masses$V5 = as.integer(masses$V5)
masses

n<-dim(masses)[1]
#Разделим данные случайным образом на обучающую и тестовую выборки
set.seed(12345)
#Для обучения возьмем 80%
n.train<- as.integer(n*0.8)
n.test<- n-n.train
rand <- masses[ order(runif(n)), ]
masses_train <- rand[1:n.train, ]
masses_test <- rand[(n.train+1):n, ]

#TSNE
#_____________________________
#Данные для tsne
tsne_masses <- masses
#Убираем неуникальные строки
tsne_masses = distinct(tsne_masses)
#Визуализация данных при помощи метода Rtsne
```

```
#Добавляем в данные цвета
for(i in 1:length(tsne_masses$V6)){
  if(tsne_masses$V6[i]== 0){
    tsne_masses$V6[i] = "green"
  }
  else{
    tsne_masses$V6[i] = "red"
  }
}
tsne_masses
#TSNE
tsne<-Rtsne(tsne_masses, pca = TRUE, dim = 2)
plot(tsne$Y, pch=19, col = tsne_masses$V6, xlab="Y[,1]", ylab="Y[,2]")
tsne_masses$V6

masses_train$V6 = as.factor(masses_train$V6)
masses_test$V6 = as.factor(masses_test$V6)

#КЛАССИФИКАТОРЫ
#_____________________________
#Наивный Байесовский классификатор
bayes <- naiveBayes(V6~.,data = masses_train)
bayes_predict <- predict(bayes,masses_test)
bayes_tab <- table(bayes_predict, masses_test$V6)
#Точность классификации
bayes_accuracy <- sum(diag(bayes_tab))/sum(bayes_tab)
#Ошибка классификации
bayes_error <- 1-bayes_accuracy
bayes_tab
bayes_accuracy

#k-ближайших соседей
#Нахождение оптимального k
fit.train <- train.kknn(V6 ~ ., masses_train, kmax = 50, distance = 1,
                        kernel = "triangular")
fit.train
#Нахождение оптимального dis
knn_dis = seq(1,50,1)
knn_err = NULL
for(i in knn_dis) {
  knn <- kknn(V6 ~ ., masses_train, masses_test, k = 3, kernel="triangular", distance=i)
  tab = table(fitted(knn), masses_test$V6)
  knn_err <- c(knn_err, 1-(sum(diag(tab))/sum(tab)))
}
res = data.frame(knn_dis, knn_err)
plot(res, xlab="Расстояние (dis)", ylab="Ошибка", type = "o", col="blue")
```

```
res

#Нахождение оптимального ядра
knn_kernel = c("rectangular", "triangular", "epanechnikov","biweight","triweight", "cos", "inv",
"gaussian", "rank", "optimal")
knn_err = NULL
for(i in knn_kernel) {
  knn <- kknn(V6 ~ ., masses_train, masses_test, k = 3, kernel=i, distance=1)
  tab = table(fitted(knn), masses_test$V6)
  knn_err <- c(knn_err, 1-(sum(diag(tab))/sum(tab)))
}
res = data.frame(knn_kernel,knn_err)
res

#Классификатор k-ближайших соседей для k = 3, kernel="cos", distance=1
knn<-kknn(as.factor(V6)~., masses_train, masses_test, k = 3, kernel="cos", distance=1)
knn_tab = table(fitted(knn),masses_test$V6)
knn_accuracy <- sum(diag(knn_tab))/sum(knn_tab)
knn_err <- 1 - knn_accuracy
knn_accuracy
knn_tab

#Бэггинг
#глубина дерева
depths = seq(1, 15, 1)
bagging_err = c()
for(i in 1:length(depths)){
  bagging <- bagging(V6 ~., data=masses_train, mfinal=1,
                     maxdepth=i)
  bagging.pred <- predict.bagging(bagging, masses_test)
  bagging_err <- append(bagging_err, bagging.pred$error)
}
plot(depths, bagging_err, xlab="Глубина дерева", ylab="Ошибка на тесте", pch=19)
bagging_err

#максимальной число итераций
mfinal = seq(1, 15, 1)
maxdepth <- 3
bagging_err = c()
for(i in 1:length(mfinal)){
  bagging <- bagging(V6 ~., data=masses_train, mfinal=mfinal[i],
                         maxdepth = maxdepth)
  bagging.pred <- predict.bagging(bagging, masses_test)
  bagging_err <- append(bagging_err, bagging.pred$error)
}
```

```
plot(mfinal, bagging_err, xlab="Максимальное число итераций", ylab ="Ошибка на тесте", pch=19)

#Бэггинг при maxdepth = 4 и mfinal = 12
bagging_err = c()
bagging <- bagging(V6 ~., data=masses_train, mfinal = 12,
                   maxdepth = 4)
bagging.pred <- predict.bagging(bagging, masses_test)
bagging_err <- append(bagging_err, bagging.pred$error)
#точность
bagging_acuracy = 1 - bagging_err
bagging_acuracy

#КЛАСТЕРИЗАЦИЯ
#_____________________________
#убираем метки
cluster_masses <- masses[,-6]
#нахождение оптимального сочетания параметров
cl_eu1 = clara(cluster_masses, k = 2, metric = "euclidean", stand = FALSE)
tb <- table(cl_eu1$clustering, masses$V6)
cl_eu1.err = sum(diag(tb))/sum(tb)
cl_eu1.err
cl_eu2 = clara(cluster_masses, k = 2, metric = "euclidean", stand = TRUE)
tb <- table(cl_eu2$clustering, masses$V6)
cl_eu2.err = sum(diag(tb))/sum(tb)
cl_eu2.err
cl_ma1 = clara(cluster_masses, k = 2, metric = "manhattan", stand = FALSE)
tb <- table(cl_ma1$clustering, masses$V6)
cl_ma1.err = sum(diag(tb))/sum(tb)
cl_ma1.err
cl_ma2 = clara(cluster_masses, k = 2, metric = "manhattan", stand = TRUE)
tb <- table(cl_ma2$clustering, masses$V6)
cl_ma2.err = sum(diag(tb))/sum(tb)
cl_ma2.err

#кластеризация с оптимальными параметрами
cl_eu2 = clara(cluster_masses, k = 2, metric = "euclidean", stand = TRUE)
plot(cluster_masses, col = cl_eu2$clustering)

#построение дендрограммы
dend_masses <- cluster_masses[1:100,1:5]
plot(agnes(dend_masses))
#сравнение полученных результатов с реальными метками данных
dend <- agnes(dend_masses)
dend$order.lab
#элементы по дентрограмме в одном кластере
```

```
indexes_cl1 = as.integer(dend$order.lab[1:69])
#этот класс 1 (злокачественная) - определено по 1-му элементу в masses
masses
#реальные метки
claster1 <- masses[indexes_cl1,]
ones = rep(1, dim(claster1)[1])
claster1$V6
tab1 <- table(claster1$V6, ones)
tab1
indexes_cl2 = as.integer(dend$order.lab[69:100])
claster2 <- masses[indexes_cl2,]
nulls = rep(0, dim(claster2)[1])
tab2 <- table(claster2$V6, nulls)
tab2
#ошибка
cluster_err = (tab1[1,1]+tab2[2,1])/100
cluster_err

#ОПРЕДЕЛЕНИЕ НАИБОЛЕЕ ЗНАЧИМЫХ ПРИЗНАКОВ
#_____________________________
x = as.matrix(masses[,-6])
y = as.matrix(masses[,6])
lambda_seq <- seq(0.001, 10, 0.001)
cv <- cv.glmnet(x, y, alpha = 1, lambda = lambda_seq, nfolds = 5)
plot(cv)
best_lam <- cv$lambda.min
best_lam
lasso_best <- glmnet(x, y, alpha = 1, lambda = best_lam, label=TRUE)
coef(lasso_best)

#АВТОКОДЕР
#_____________________________
#использование автокодера
#число слоев – 3; число нейронов в скрытом слое – 4; функц. актив. – tanh; eps = 0
masses[,-6]
encoder<-autoencode(as.matrix(masses[,-6]),
                    nl=3,
                    N.hidden = 4,
                    epsilon = 0,
                    unit.type = "tanh",
                    lambda = 0.0002,
                    beta = 6,
                    rho = 0.1,
                    max.iterations = 20000,
                    rescale.flag = TRUE)
decomposition <- predict.autoencoder(encoder, as.matrix(masses[,-6]),hidden.output = TRUE)
```

```
decomposition

#добавляем к полученным данным столбец с метками
decomposition.frame <- as.data.frame(decomposition$X.output)
V5 <- as.numeric(masses$V6)
decomposition.encode.frame <- tibble::add_column(decomposition.frame,V5)
decomposition.encode.frame

#Убираем неуникальные строки
tsne_frame <- distinct(decomposition.encode.frame)

#Визуализация данных при помощи метода Rtsne
tsne <- Rtsne(as.matrix(tsne_frame),pca = TRUE, dim = 2)
plot(tsne$Y, pch=21, bg=c("green","red"), xlab="Y[,1]", ylab="Y[,2]")

#Разделим данные случайным образом на обучающую и тестовую выборки
set.seed(12345)
#Для обучения возьмем 80%
train<- as.integer(n*0.8)
test<- n-train
rand <- decomposition.encode.frame[ order(runif(n)), ]
encode_train <- rand[1:train, ]
encode_test <- rand[(train+1):n, ]
encode_train
#knn
#Нахождение оптимального k
fit.train <- train.kknn(V5~., encode_train, kmax = 50, distance = 1,
                        kernel = "rectangular")
fit.train
#классификация
knn<-kknn(V5~., encode_train, encode_test, k=4, kernel="rectangular", distance=1)
tab <- table(fitted(knn),encode_test$V5)
acuracy <- (sum(diag(tab))/sum(tab))
acuracy
err = 1-(sum(diag(tab))/sum(tab))
err

#разряженная размерность: N.hidden = 20
encoder<-autoencode(as.matrix(masses[,-6]), nl=3, N.hidden = 2,epsilon = 0.5,
                    lambda = 0.0002,
                    beta = 6,
                    rho = 0.9,
                    unit.type = "tanh",
                    max.iterations = 20000,
                    rescale.flag = TRUE)
reconstraction<-predict.autoencoder(encoder,as.matrix(masses[,-6]),hidden.output = FALSE)
```

```
decomposition<-predict.autoencoder(encoder,as.matrix(masses[,-6]),hidden.output = TRUE)
decomposition
#добавляем к полученным данным столбец с метками
decomposition.frame <- as.data.frame(decomposition$X.output)
V21 <- as.numeric(masses$V6)
decomposition.encode.frame <- tibble::add_column(decomposition.frame,V21)
decomposition.encode.frame
#Разделим данные случайным образом на обучающую и тестовую выборки
set.seed(12345)
#Для обучения возьмем 80%
train<- as.integer(n*0.8)
test<- n-train
rand <- decomposition.encode.frame[ order(runif(n)), ]
encode_train <- rand[1:train, ]
encode_test <- rand[(train+1):n, ]
encode_train
#knn
#Нахождение оптимального k
fit.train <- train.kknn(V21~., encode_train, kmax = 50, distance = 1,
                        kernel = "rectangular")
fit.train
#классификация
knn<-kknn(V21~., encode_train, encode_test, k=3, kernel="rectangular", distance=1)
tab <- table(fitted(knn),encode_test$V21)
acuracy <- (sum(diag(tab))/sum(tab))
acuracy
err = 1-(sum(diag(tab))/sum(tab))
err
```