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

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Метод поиска ближайшего соседа»**

по дисциплине «Машинное обучение»

1. Выполнил: студент группы
2. 3540201/20302 А.И. Обидина

*<подпись>*

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

Санкт-Петербург

2022

**Задание**

1. Исследовать, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации или на вероятность ошибочной классификации в примере крестики-нолики и примере о спаме e-mail сообщений.

2. Построить классификатор для обучающего множества **Glass**, данные которого характеризуются 10-ю признаками:

1. Id number: 1 to 214; 2. RI: показатель преломления; 3. Na: сода (процент содержания в соотвествующем оксиде); 4. Mg; 5. Al; 6. Si; 7. K; 8. Ca; 9. Ba; 10. Fe.

Классы характеризуют тип стекла:

(1) окна зданий, плавильная обработка

(2) окна зданий, не плавильная обработка

(3) автомобильные окна, плавильная обработка

(4) автомобильные окна, не плавильная обработка (нет в базе)

(5) контейнеры

(6) посуда

(7) фары

Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки. Это выполняется командой **glass <- glass[,-1]**.

Постройте графики зависимости ошибки классификации от значения k и от типа ядра.

Исследуйте, как тип метрики расстояния (параметр **distance**) влияет на точность классификации.

Определите, к какому типу стекла относится экземпляр с характеристиками

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

Определите, какой из признаков оказывает наименьшее влияние на определение класса путем последовательного исключения каждого признака.

3. Для построения классификатора используйте заранее сгенерированные обучающие и тестовые выборки, хранящиеся в файлах svmdata4.txt, svmdata4test.txt. Найдите оптимальное значение k, обеспечивающее наименьшую ошибку классификации. Посмотрите, как выглядят данные на графике, используя функцию

**plot(mydata.train$X1, mydata.train$X2, pch=21, bg=c("red","blue") [unclass(mydata.train$Colors)], main="My train data")**

4. Разработать классификатор на основе метода ближайших соседей для данных **Титаник (Titanic dataset) -** <https://www.kaggle.com/c/titanic>

Исходные обучающие данные для классификации – в файле Titanic\_train.csv

Данные для тестирования – в файле Titanic\_test.csv

# Ход работы

## **Пункт 1**

1. Размер датасета «Крестики-нолики» - 958.

Для того, чтобы выявить, как объем обучающей выборки и количество тестовых данных из набора «Крестики-нолики» влияют на точность классификации или на вероятность ошибочной классификации, был проведён циклический эксперимент. Объем обучающей выборки увеличивался с 0,1 части выборки до 0,9 части выборки с шагом 0,05 выборки. Для каждой итерации метод knn был повторен 50 раз со значением , ядро – triangular, параметр расстояния Минковского – 1. Точность классификации вычисляется как отношение верных предсказаний к произведению разности общего количества данных и отобранных данных на текущей итерации, а также 50. На рисунке 1 приведен полученный результат.

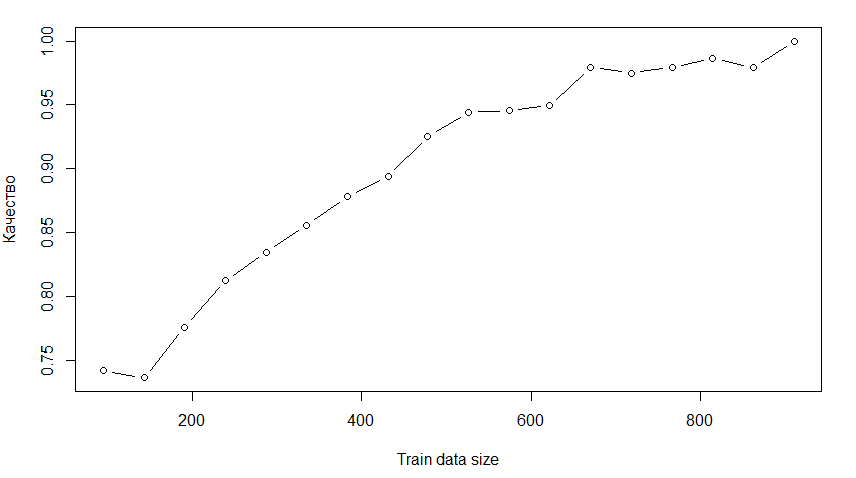


Рисунок 1 – Зависимость качества модели от размера обучающей выборки

Из полученного графика видно, что точность почти прямо пропорционально зависит от размера обучающей выборки. Наилучшая точность в данном эксперименте была достигнута при объёме обучающей выборки 1000 (в этом случае точность равна 100%). Самые лучшие показатели точности – в интервале от 0,7 до 1,0 части выборки.

Проанализировав полученные данные, можно сказать, что модель KNN справилась с задачей лучше, чем наивный Байесовский классификатор из лабораторной №1 для данного датасета.

1. Для того, чтобы выявить, как объем обучающей выборки и количество тестовых данных из набора «Спам» влияют на точность классификации или на вероятность ошибочной классификации, был проведён циклический эксперимент. Объем тестовой выборки увеличивался со 100 записей базы данных до 1000 с шагом в 100 записей. Для каждой итерации метод knn использован 50 раз со значением , ядро – triangular, параметр расстояния Минковского – 1. Точность классификации вычисляется как отношение верных предсказаний к произведению разности общего количества данных и отобранных данных, а также 50. На рисунке 2 приведен полученный результат.

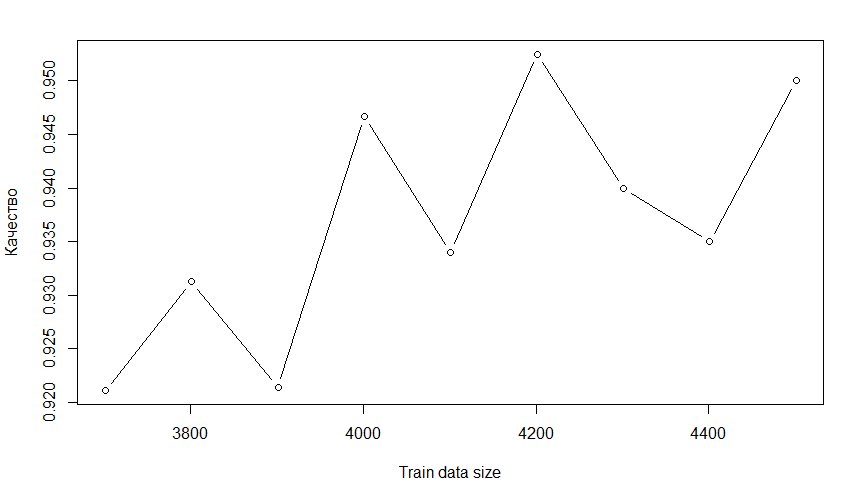


Рисунок 2 – Зависимость качества модели от размера обучающей выборки

По графику нельзя выявить точную зависимость качества классификации от объёма входных данных. При этом лучшая точность наблюдается при использовании 4201 записи в качестве обучающей выборки (точность равна 0,953).

Проанализировав полученные данные, можно сказать, что модель KNN справилась с задачей лучше, чем наивный Байесовский классификатор из лабораторной №1 для данного датасета.

**Код:**

library(kknn)

first\_data <- read.table("D:\\Tic\_tac\_toe.txt", sep = ",", stringsAsFactors = TRUE)

first\_data

strings\_num<-dim(first\_data)[1]

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

rand\_data

train<-c()

prediction<-c()

acc<-c()

start<-0.1

finish<-1

for\_one\_volume<-function(incr, idx, fl){

pr<-0

tr<-0

for(i in seq(1, repeats\_num)){

if (fl == 0) {

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

train\_data<-rand\_data[1:incr, ]

train\_data

A\_classifier<-kknn(V10 ~ .,

train = train\_data,

test = test\_data,

distance = 1,

k=round(sqrt(incr), digits = 0),

kernel = "triangular")

A\_predicted<-A\_classifier$fitted.values

res <- table(A\_predicted, test\_data$V10)

pr <- pr + res[1] + res[4]

}

else{

test\_data <- spam[idx,]

train\_data <- spam[-idx,]

model <- kknn(type ~ .,

train = train\_data,

test = test\_data,

distance = 1,

k=round(sqrt(incr), digits = 0),

kernel = "triangular")

res <- table(model$fitted.values, test\_data$type)

tr <- dim(train\_data)[1]

tr

pr <- pr + res[1] + res[4]

pr

}

}

return(list(tr,pr))

}

#tic-tac-toe

repeats\_num<-50

s<-start

k<-1

while (s<finish) {

incr<-as.integer(strings\_num\*s)

train[k]<-incr

tr\_pr = as.numeric(for\_one\_volume(incr,0,0))

prediction[k]<-tr\_pr[2]

acc[k] <- prediction[k] / (repeats\_num\*(strings\_num-incr))

s<-s+0.05

k<-k+1

}

train

acc

plot(train, acc, type = "b", xlab = "Train data size", ylab = "Качество")

#spam

library(kernlab)

library(e1071)

data(spam)

train<-c()

prediction<-c()

acc<-c()

spam[0:1,]

start<-100

finish<-1000

repeats\_num<-50

k<-1

s<-start

while (s<finish) {

idx <- sample(1:dim(spam)[1], s)

test\_data <- spam[idx,]

tr\_pr<-as.numeric(for\_one\_volume(s,idx,1))

train[k]<-tr\_pr[1]

prediction[k]<-tr\_pr[2]

acc[k] <- prediction[k] / (repeats\_num\*dim(test\_data)[1])

s<-s+100

k<-k+1

}

plot(train, acc, type = "b", xlab = "Train data size", ylab = "Качество")

## **Пункт 2**

С помощью метода train.kknn были получены графики зависимости ошибки классификации от типа ядра, параметра k и значения расстояния Минковского (представлены на рисунках 3 - 7). Были рассмотрены все возможные варианты ядер, значения расстояний Минковского – 1, 2, 3, 7, 10. Так же для каждого из полученных классификаторов выбран наилучшее ядро и значение параметра k.

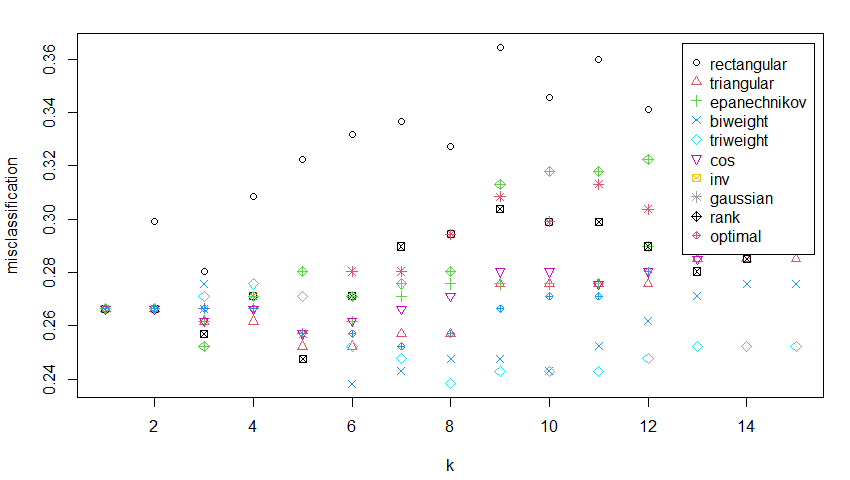




Рисунок 3 – Зависимость ошибки от типа ядра и параметра k, расстояние Минковского = 1

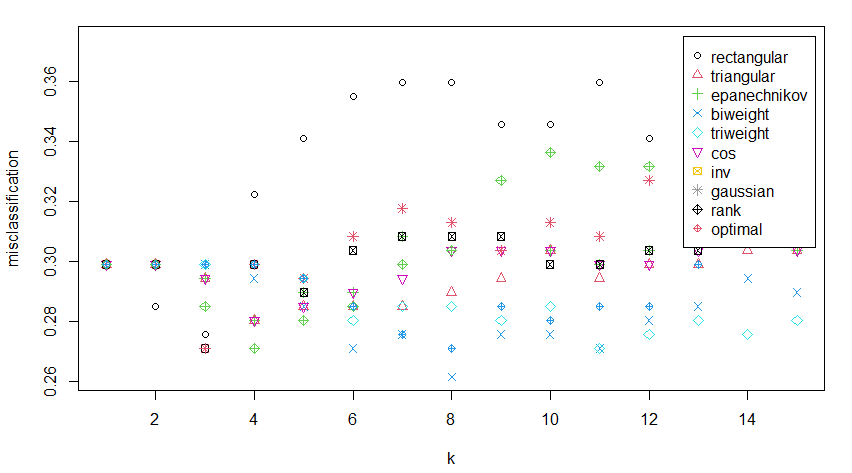
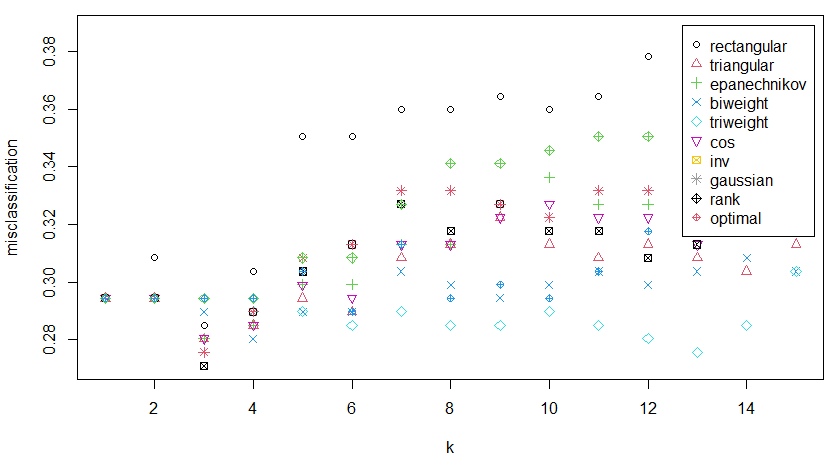




Рисунок 4 – Зависимость ошибки от типа ядра и параметра k, расстояние Минковского = 2



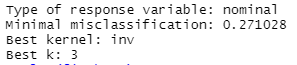
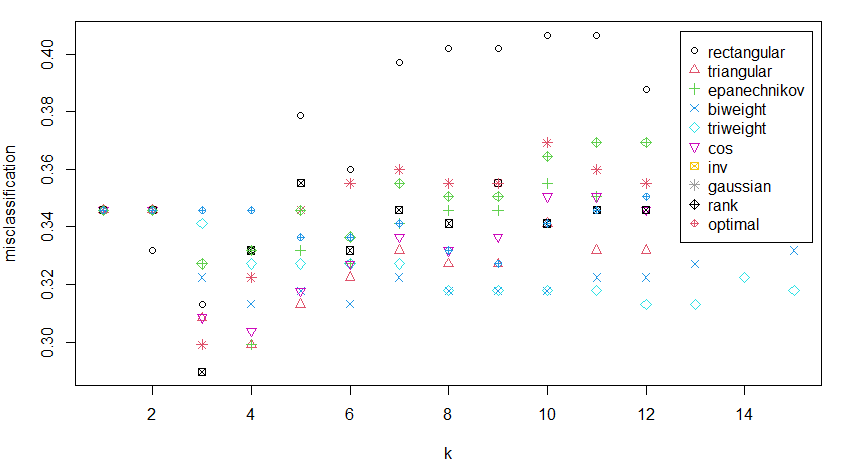


Рисунок 5 – Зависимость ошибки от типа ядра и параметра k, расстояние Минковского = 3



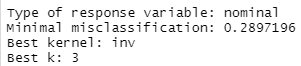


Рисунок 6 – Зависимость ошибки от типа ядра и параметра k, расстояние Минковского = 7

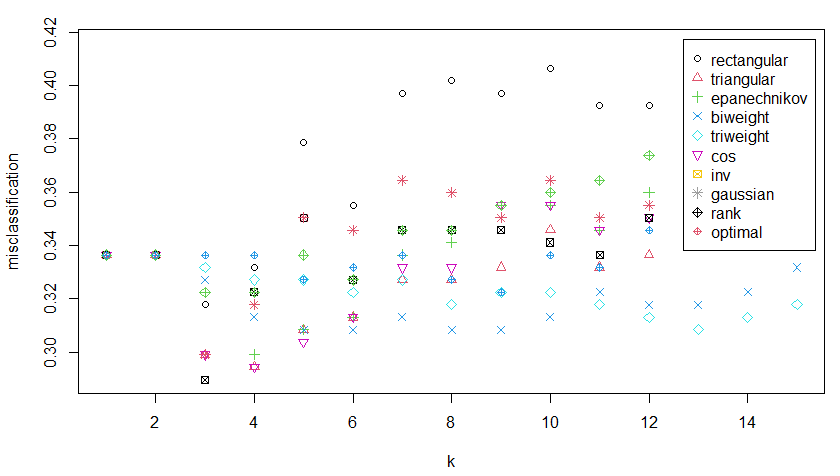
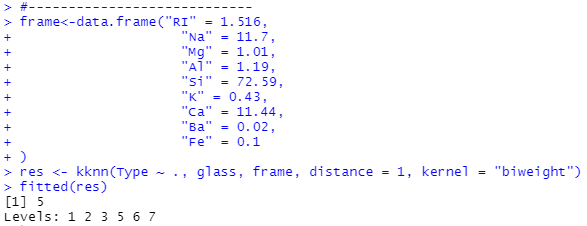




Рисунок 7 – Зависимость ошибки от типа ядра и параметра k, расстояние Минковского = 10

Типы ядер biweight и triweight показывают наименьшую ошибку вне зависимости от параметра k и значений расстояния Минковского. Ядро rectangular показывает наибольшие значения ошибки (с увеличением k). Конкретной зависимости для остальных типов ядер не было выявлено. Самое оптимальное расстояние Минковского – 2.

Стекло с заданными параметрами было определено как стекло 5-го типа – контейнерное:



Из набора данных glass были последовательно исключены признаки, после чего вычислялось качество классификации модели. Исключение признаков процентного содержания железа и натрия не оказало влияния на определение класса (рисунок 8). Наиболее значимый параметр – кальций.

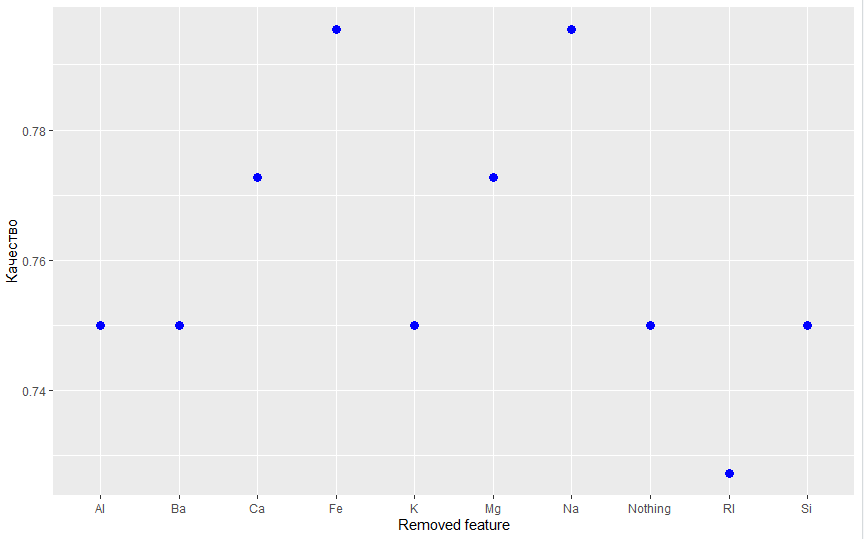


Рисунок 8 – График зависимости качества классификации от удаления признаков

**Код:**

library(kknn)

library(kernlab)

data(glass)

glass<-glass[,-1]

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

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

ker = c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal")

fit.knn1<-train.kknn(Type ~.,glass, kmax=round(sqrt(strings\_num), digits = 0), kernel=ker, distance=1)

fit.knn1

plot(fit.knn1)

fit.knn2<-train.kknn(Type ~.,glass, kmax=round(sqrt(strings\_num), digits = 0), kernel=ker, distance=2)

fit.knn2

plot(fit.knn2)

fit.knn3<-train.kknn(Type ~.,glass, kmax=round(sqrt(strings\_num), digits = 0), kernel=ker, distance=3)

fit.knn3

plot(fit.knn3)

fit.knn4<-train.kknn(Type ~.,glass, kmax=round(sqrt(strings\_num), digits = 0), kernel=ker, distance=7)

fit.knn4

plot(fit.knn4)

fit.knn5<-train.kknn(Type ~.,glass, kmax=round(sqrt(strings\_num), digits = 0), kernel=ker, distance=10)

fit.knn5

plot(fit.knn5)

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

frame<-data.frame("RI" = 1.516,

"Na" = 11.7,

"Mg" = 1.01,

"Al" = 1.19,

"Si" = 72.59,

"K" = 0.43,

"Ca" = 11.44,

"Ba" = 0.02,

"Fe" = 0.1

)

res <- kknn(Type ~ ., glass, frame, distance = 1, kernel = "biweight")

fitted(res)

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

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

train\_data<-glass[1:170,]

test\_data<-glass[(170+1):strings\_num,]

acc<-c()

features1<-c("Nothing", "RI","Na","Mg","Al","Si","K","Ca","Ba","Fe")

features<-c(0,1,2,3,4,5,6,7,8,9)

for(j in 1:10)

{

i=j-1

i

pr=0

train<-train\_data

test<-test\_data

if (i > 0) {

train<-train\_data[,-i]

test<-test\_data[,-i]

}

knn<-kknn(Type ~ .,train,test,k=round(sqrt(strings\_num),digits=0),distance=1,kernel="biweight")

glass\_fit<-fitted(knn)

for(k in 1:length(glass\_fit)){

if(glass\_fit[k]==test$Type[k])

pr=pr+1

}

print(pr/dim(test)[1])

acc[j]<-pr/dim(test)[1]

}

acc

features

plot(features, acc, type = "b", xlab = "Removed feature", ylab = "Качество")

library(ggplot2)

ggplot(data.frame(features1,acc)) +

labs(x = "Removed feature", y = "Качество") +

geom\_point(aes(x = features1, y = acc), color = "blue", lwd = 3)

## **Пункт 3**

Обучающая выборка была визуализирована, ее распределение представлено на рисунке 9. Видно, что разделение классов довольно явное.

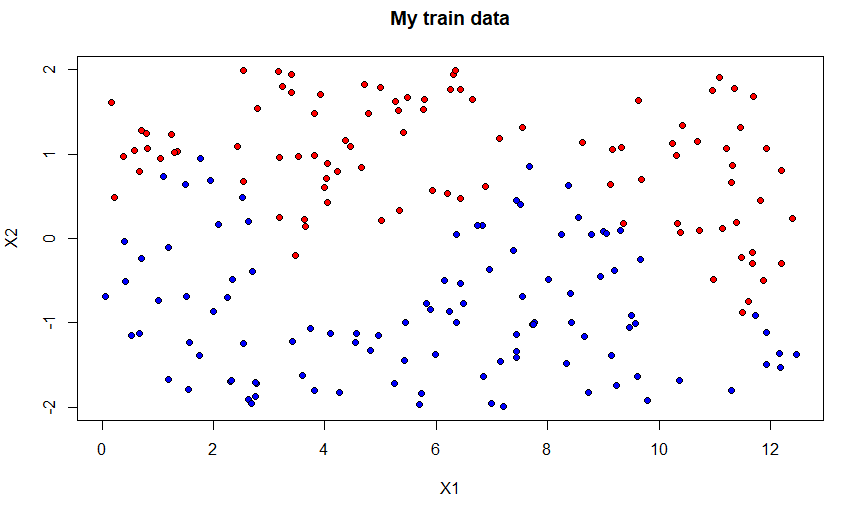


Рисунок 9 – Распределение обучающей выборки

С помощью метода train.kknn был получен график зависимости ошибки классификации от параметра k (представлен на рисунке 10). Наименьшее значение ошибки было получено при количестве ближайших соседей, равном 8.

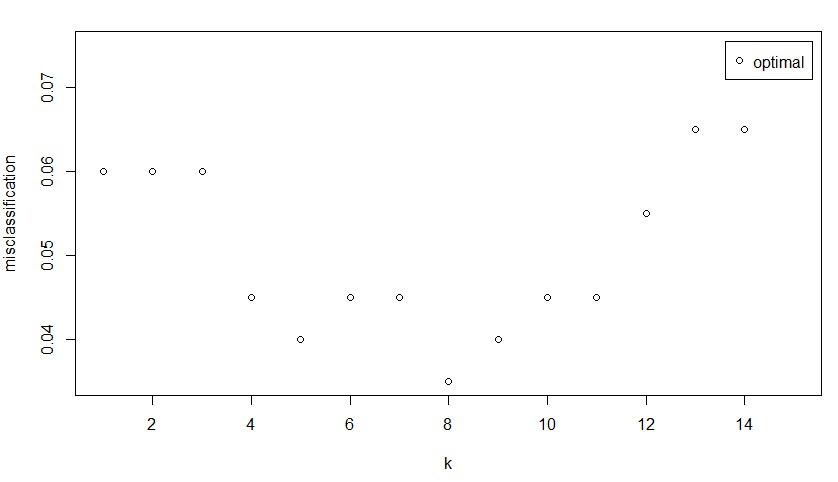


Рисунок 10 – График зависимости ошибки классификации от параметра k

**Код:**

library(kknn)

train\_data<-read.table("D:\\svmdata4.txt",stringsAsFactors = TRUE)

test\_data<-read.table("D:\\svmdata4test.txt",stringsAsFactors = TRUE)

strings\_number<-dim(train\_data)[1]

plot(train\_data$X1, train\_data$X2, pch=21, bg=c("red","blue") [unclass(train\_data$Colors)], main="My train data", xlab = "X1", ylab="X2")

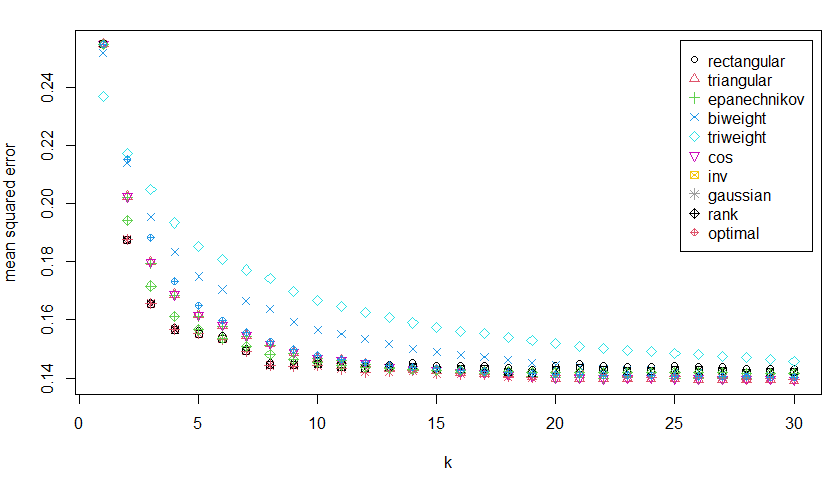
knn<-train.kknn(Colors ~ .,train\_data,kmax=round(sqrt(strings\_num),digits=0),distance=2,kernel="optimal")

knn

plot(knn)

## **Пункт 4**

С помощью метода train.kknn был реализован классификатор данных для датасета Titanic, график зависимости ошибки классификации от типа ядра и параметра k, а также наилучшие значения изменяемых гиперпараметров классификатора представлены на рисунке 11.



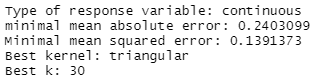


Рисунок 11 – График зависимости ошибки классификации от параметра k и типа ядра

Код:

library(kknn)

train\_data<-read.csv(file="D:\\All\_Labs\\train.csv")

train\_data

test\_data<-read.csv(file="D:\\All\_Labs\\test.csv")

strings\_num<-dim(train\_data)[1]

knn<-train.kknn(Survived ~ ., train\_data,

kmax=round(sqrt(strings\_num),digits=0),

kernel=c("rectangular",

"triangular",

"epanechnikov",

"biweight",

"triweight",

"cos",

"inv",

"gaussian",

"rank",

"optimal"),

distance=2)

knn

plot(knn)

## **Вывод**

В ходе выполнения данной лабораторной работы на практике был исследован метод ближайших соседей, а также зависимость качества классификации с помощью этого метода от значений гиперпараметров, объема обучающей выборки. Данный метод достаточно прост в реализации, но хорош в использовании. При этом, используя данный метод, необходимо хранить целиком всю обучающую выборку, также он достаточно трудоёмок в вычислительном плане (трудоемкость растет квадратично с увеличением числа обучающих примеров).