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

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2

«Метод ближайших соседей»
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 рамках данной работы необходимо:

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

Посмотреть 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]**.

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

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

Определить, 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.

Определить, 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, а данные для тестирования – в файле Titanic_test.csv.

# 1 Исследование зависимости точности классификатора от размера выборки

## 1.1 «Крестики-нолики»

Рассмотрим выборку из 958 элементов, каждый из которых для определенного расположения в игре «Крестики-нолики» классифицирует его как выигрышное (positive) или проигрышное (negative) для x:

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 623 | b | b | b | x | x | x | b | o | o | positive |
| 624 | b | b | b | o | o | b | x | x | x | positive |
| 625 | b | b | b | o | b | o | x | x | x | positive |
| 626 | b | b | b | b | o | o | x | x | x | positive |
| 627 | x | x | o | x | x | o | o | b | o | negative |
| 628 | x | x | o | x | x | o | b | o | o | negative |
| 629 | x | x | o | x | x | b | o | o | o | negative |

Отобразим зависимость точности классификатора от размера тестовой и обучающей выборки из датасета «Крестики-нолики». При построении классификатора используем параметры: k = 7 – число ближайших соседей; kernel = «optimal» – ядро; distance = 2 – параметр расстояния Минковского.

Для оценки точности используем метрику accuracy, где TP и TN - верно классифицируемые объекты:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

При обучающей выборке 90%, а тренировочной 10%

```
A_predicted negative positive
   negative       24        5
   positive        7       60
> accuracy
[1] 0.875
```

Данные всех экспериментов приведены в Таблице 1.

Таблица 2 – Зависимость точности от выборки – «Крестики-нолики»

| Процент обучающей выборки о общего объема данных | Точность классификации |
|---|---|
| 90% | 0.875 |
| 80% | 0.8645833 |
| 70% | 0.8854167 |
| 60% | 0.9010417 |
| 50% | 0.9164927 |
| 40% | 0.9113043 |
| 30% | 0.9329359 |
| 20% | 0.8917862 |
| 10% | 0.7949015 |

По полученным данным можно сделать вывод о том, что самая высокая точность классификации при объеме обучающей выборки от 30% до 60%. А также можно сделать вывод о том, что классификатор, построенный с помощью метода k ближайших соседей на датасете «Крестики-нолики» дает более хорошую точность классификации (≈ 0.88), чем при использовании наивного Байесовского классификатора (≈ 0.71).

## 1.2 Классификация спама

Рассмотрим выборку из 4601 элемента, каждый из которых для набора признаков текстового сообщения классифицирует его как спам или не спам:

| charExclamation | charDollar | charHash | capitalAve | capitalLong | capitalTotal | type |
|---|---|---|---|---|---|---|
| 0.778 | 0.000 | 0.000 | 3.756 | 61 | 278 | spam |
| 0.372 | 0.180 | 0.048 | 5.114 | 101 | 1028 | spam |
| 0.276 | 0.184 | 0.010 | 9.821 | 485 | 2259 | spam |
| 0.137 | 0.000 | 0.000 | 3.537 | 40 | 191 | spam |
| 0.135 | 0.000 | 0.000 | 3.537 | 40 | 191 | spam |
| 0.000 | 0.000 | 0.000 | 3.000 | 15 | 54 | spam |
| 0.164 | 0.054 | 0.000 | 1.671 | 4 | 112 | spam |

Отобразим зависимость точности классификатора от размера тестовой и обучающей выборки из датасета «Спам». При построении классификатора используем параметры: k = 7 – число ближайших соседей;

kernel = «optimal» – ядро; distance = 2 – параметр расстояния Минковского.

Для оценки точности используем метрику accuracy, где TP и TN - верно классифицируемые объекты:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

При обучающей выборке 90%, а тренировочной 10%

```
predict   nonspam spam
  nonspam    2359  441
  spam        142 1198
> accuracy2
[1] 0.8591787
```

Данные всех экспериментов приведены в Таблице 2.

Таблица 2 – Зависимость точности от выборки – «Спам»

| Процент обучающей выборки о общего объема данных | Точность классификации |
|---|---|
| 90% | 0.8591787 |
| 80% | 0.8752717 |
| 70% | 0.8947205 |
| 60% | 0.8938406 |
| 50% | 0.8978261 |
| 40% | 0.9152174 |
| 30% | 0.9086957 |
| 20% | 0.9206522 |
| 10% | 0.9260871 |

По полученным результатам можно сделать вывод о том, что точность работы классификатора наименьшая при объеме обучающей выборки от 80% до 90%. Также можно видеть, что и для датасета «Спам» метод k ближайших соседей работает лучше, чем наивный Байесовский классификатор.

Код для всех реализуемых выше экспериментов представлен в Приложении 1.

# 2 Классификатор для датасета «Glass»

Датасет Glass содержит 214 строк:

| | Id | RI | Na | Mg | Al | Si | K | Ca | Ba | Fe | Type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1.52101 | 13.64 | 4.49 | 1.10 | 71.78 | 0.06 | 8.75 | 0.00 | 0.00 | 1 |
| 2 | 2 | 1.51761 | 13.89 | 3.60 | 1.36 | 72.73 | 0.48 | 7.83 | 0.00 | 0.00 | 1 |
| 3 | 3 | 1.51618 | 13.53 | 3.55 | 1.54 | 72.99 | 0.39 | 7.78 | 0.00 | 0.00 | 1 |
| 4 | 4 | 1.51766 | 13.21 | 3.69 | 1.29 | 72.61 | 0.57 | 8.22 | 0.00 | 0.00 | 1 |
| 5 | 5 | 1.51742 | 13.27 | 3.62 | 1.24 | 73.08 | 0.55 | 8.07 | 0.00 | 0.00 | 1 |
| 6 | 6 | 1.51596 | 12.79 | 3.61 | 1.62 | 72.97 | 0.64 | 8.07 | 0.00 | 0.26 | 1 |

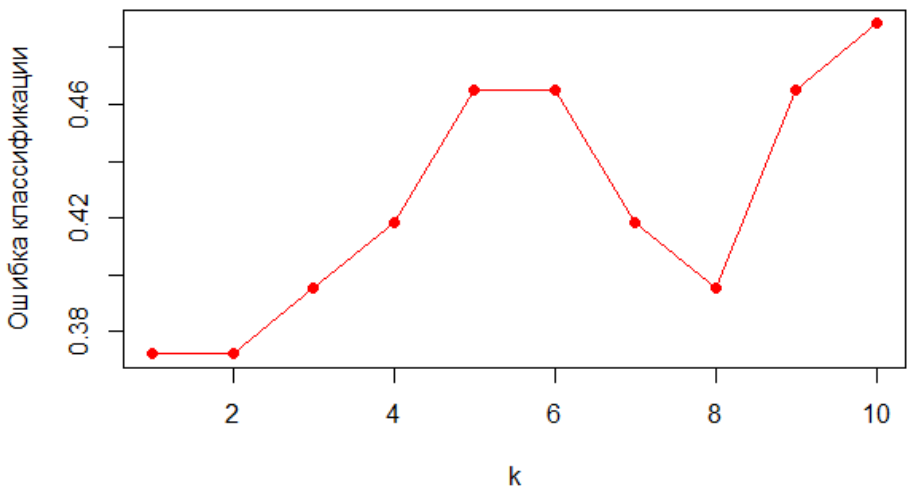
Удалим первый столбец, так как он не несет информации для классификации:

| | RI | Na | Mg | Al | Si | K | Ca | Ba | Fe | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.52101 | 13.64 | 4.49 | 1.10 | 71.78 | 0.06 | 8.75 | 0.00 | 0.00 | 1 |
| 2 | 1.51761 | 13.89 | 3.60 | 1.36 | 72.73 | 0.48 | 7.83 | 0.00 | 0.00 | 1 |
| 3 | 1.51618 | 13.53 | 3.55 | 1.54 | 72.99 | 0.39 | 7.78 | 0.00 | 0.00 | 1 |
| 4 | 1.51766 | 13.21 | 3.69 | 1.29 | 72.61 | 0.57 | 8.22 | 0.00 | 0.00 | 1 |
| 5 | 1.51742 | 13.27 | 3.62 | 1.24 | 73.08 | 0.55 | 8.07 | 0.00 | 0.00 | 1 |
| 6 | 1.51596 | 12.79 | 3.61 | 1.62 | 72.97 | 0.64 | 8.07 | 0.00 | 0.26 | 1 |

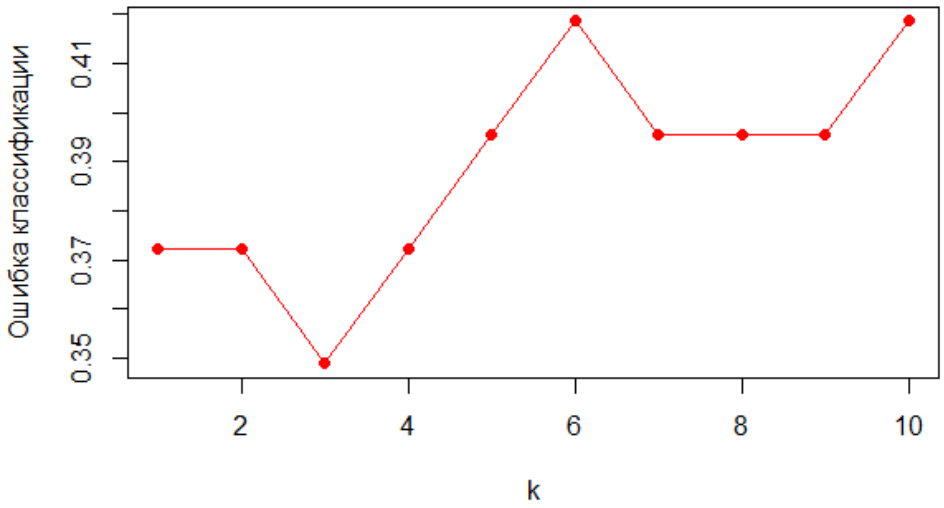
Построим классификатор с помощью метода k ближайших соседей.

1) Проведем исследование зависимости ошибки классификации от числа соседей (рассмотрим k от 1 до 10) при разных значениях ядер, расстояние distance возьмем постоянное и равное 2.

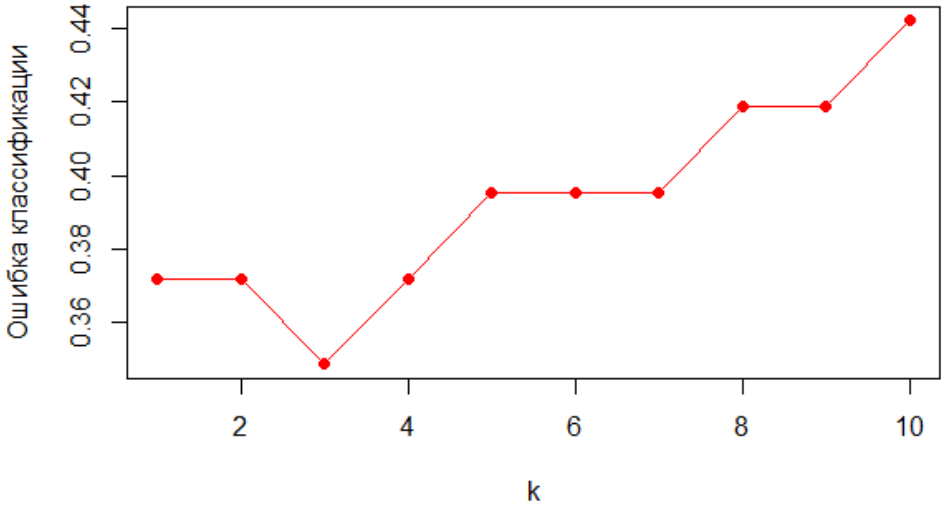
**Kernel = «rectangular»:**



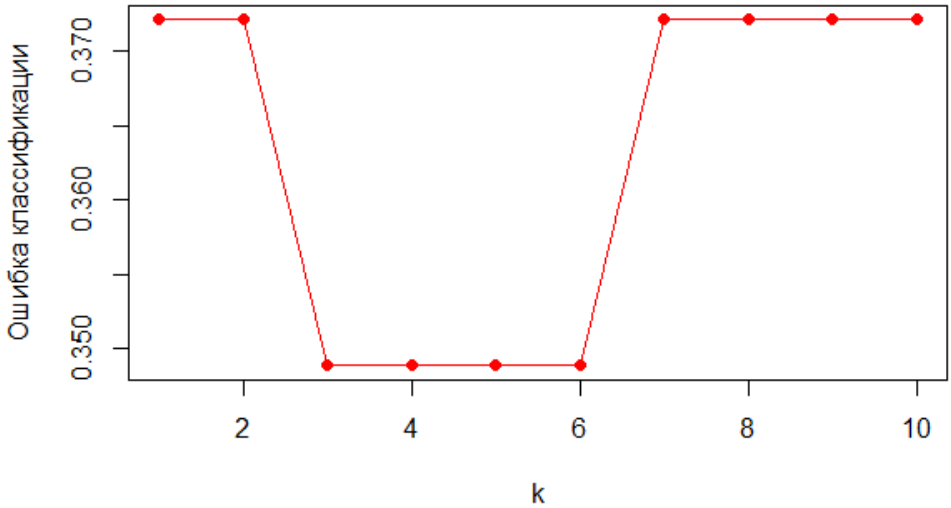
**Kernel = «triangular»:**



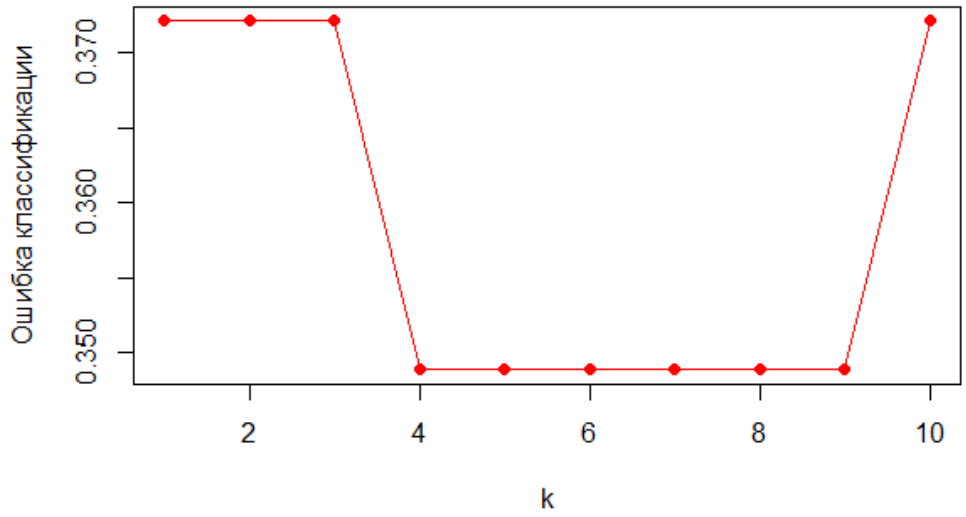

**Kernel = «epanechnikov»:**



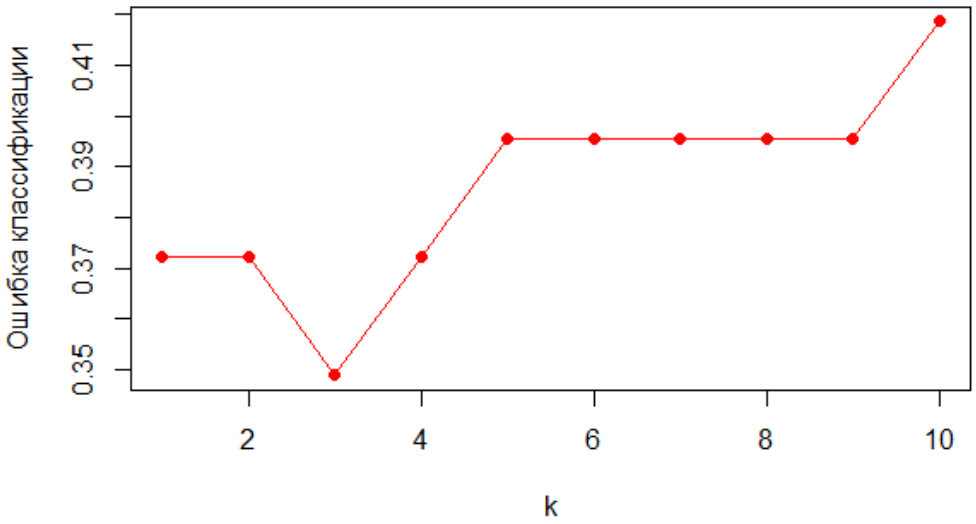

**Kernel = «biweight»:**



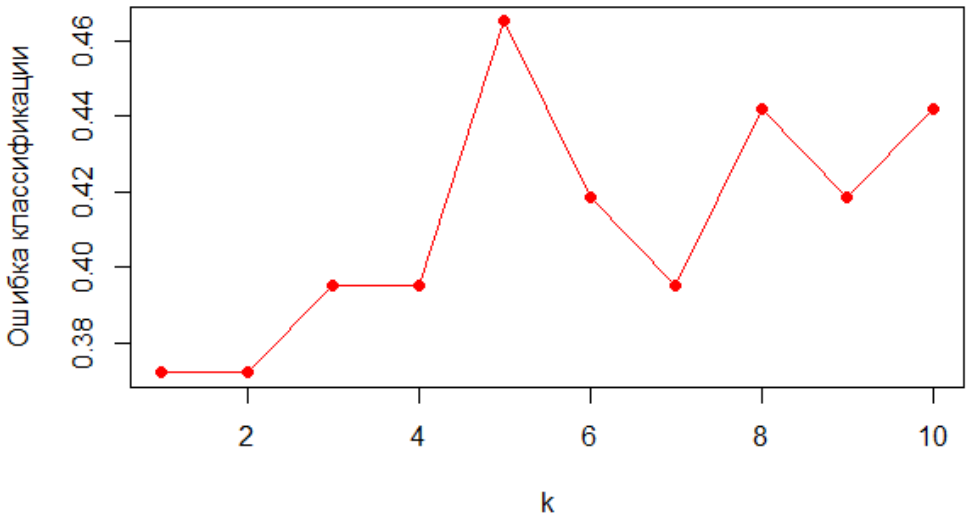
**Kernel = «triweight»:**



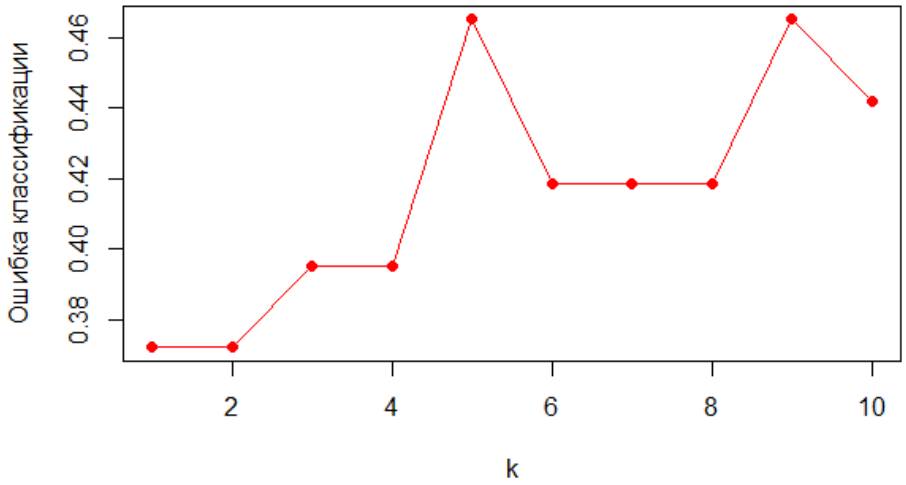

**Kernel = «cos»:**



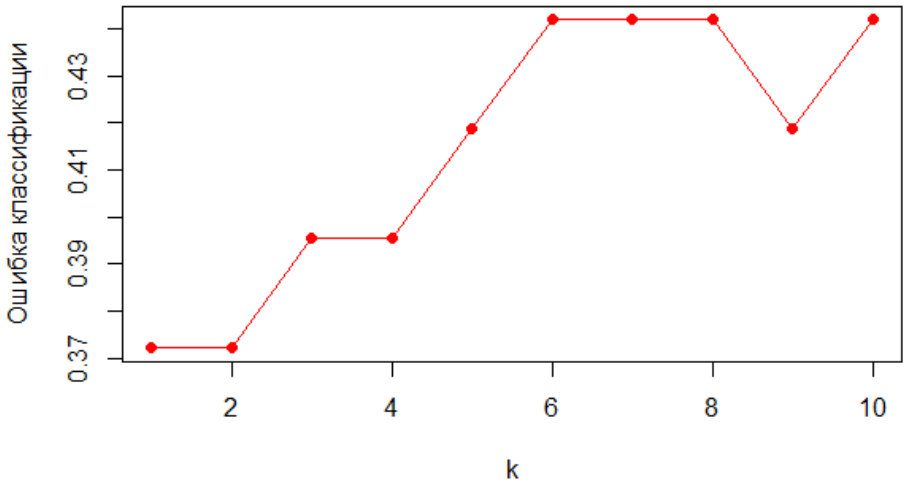

**Kernel = «inv»:**



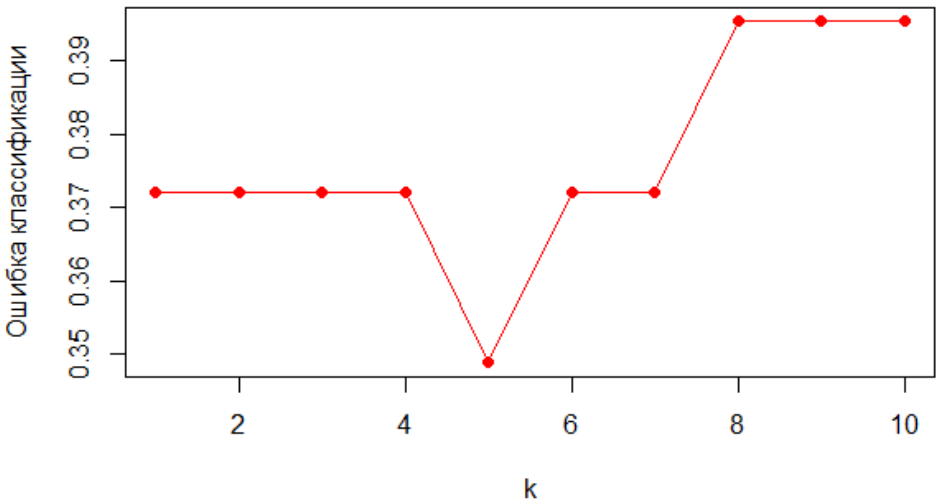
**Kernel = «gaussian»:**




**Kernel = «rank»:**




**Kernel = «optimal»:**



По полученным данным можно сделать вывод об оптимальном значении k для каждого из ядер:

| Ядро | Число k, при котором ошибка классификации наименьшая |
|---|---|
| rectangular | 1 |
| triangular | 3 |
| epanechnikov | 3 |
| biweight | 4 |
| triweight | 4 |
| cos | 3 |
| inv | 1 |
| gaussian | 1 |
| rank | 1 |
| optimal | 5 |

Для используемого датасета ошибка минимальна (≈ 0.345) при k = 5 и ядре «optimal».

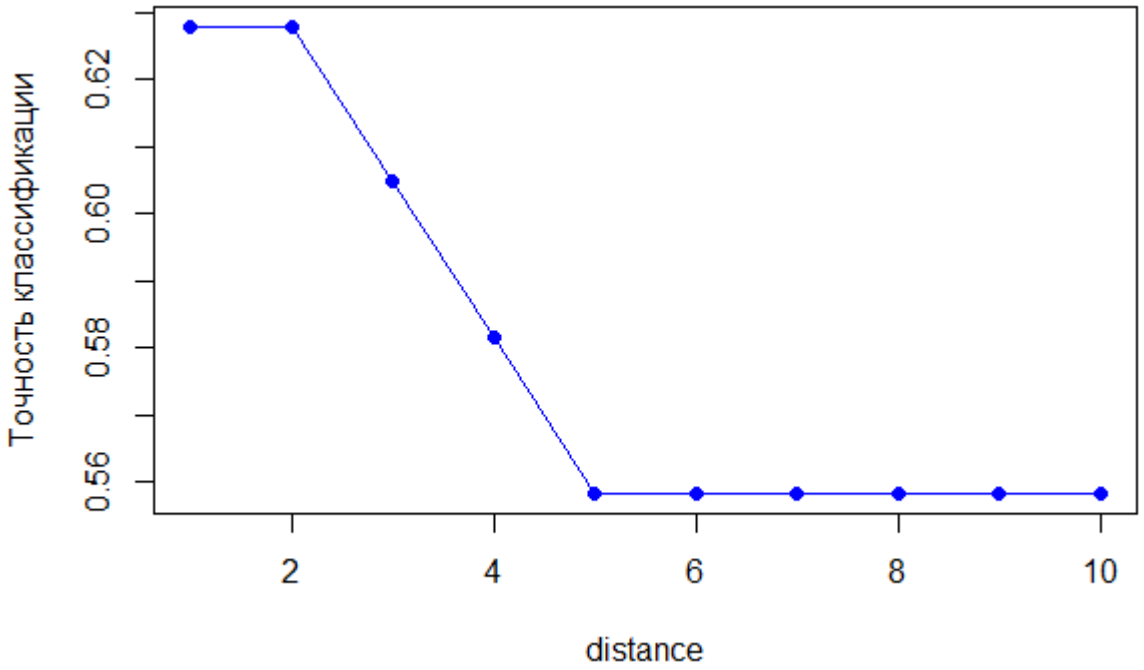
2) Исследуем влияние параметра distance на точность классификации.

В таблице 3 представлены результаты точности, полученные при различных значениях расстояния при использовании классификатора с параметрами: k = 7; kernel = «optimal».

Таблица 3 – Зависимость точности от параметра distance

| Значение параметра distance | Точность классификации |
|---|---|
| 1 | 0.6279070 |
| 2 | 0.6279070 |
| 3 | 0.6046512 |
| 4 | 0.5813953 |
| 5 | 0.5581395 |
| 6 | 0.5581395 |
| 7 | 0.5581395 |
| 8 | 0.5581395 |
| 9 | 0.5581395 |
| 10 | 0.5581395 |

График зависимости точности от расстояния distance:




Самая высокая точность классификации (≈ 63%) при distance = 1. При увеличении параметра distance точность классификации уменьшается.

3) Определим 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.

Данный экземпляр с вероятностью:

- 0.68 относится к 5-му классу (контейнеры);
- 0.23 относится ко 2-му классу (окна зданий, не плавильная обработка);
- 0.07 относится к 1-му классу (окна зданий, плавильная обработка);
- 0.02 относится к 3-му классу (автомобильные окна, плавильная обработка).

4) Исследуем как признаки влияют на точность классификации, путем последовательного исключения каждого признака:

| Признак, который исключен | Точность классификации |
|---|---|
| 1) RI | 0.6976744 |
| 2) Na | 0.7906977 |
| 3) Mg | 0.6976744 |
| 4) Al | 0.7441861 |

| | |
|---|---|
| 5) Si | 0.6976744 |
| 6) K | 0.6744186 |
| 7) Ca | 0.7209302 |
| 8) Ba | 0.7441861 |
| 9) Fe | 0.7906977 |

По результатам исследования можно видеть, что самая высокая точность классификатора (≈ 79%) при исключении признака Fe, значит данный признак в меньшей степени влияет на результат классификации.

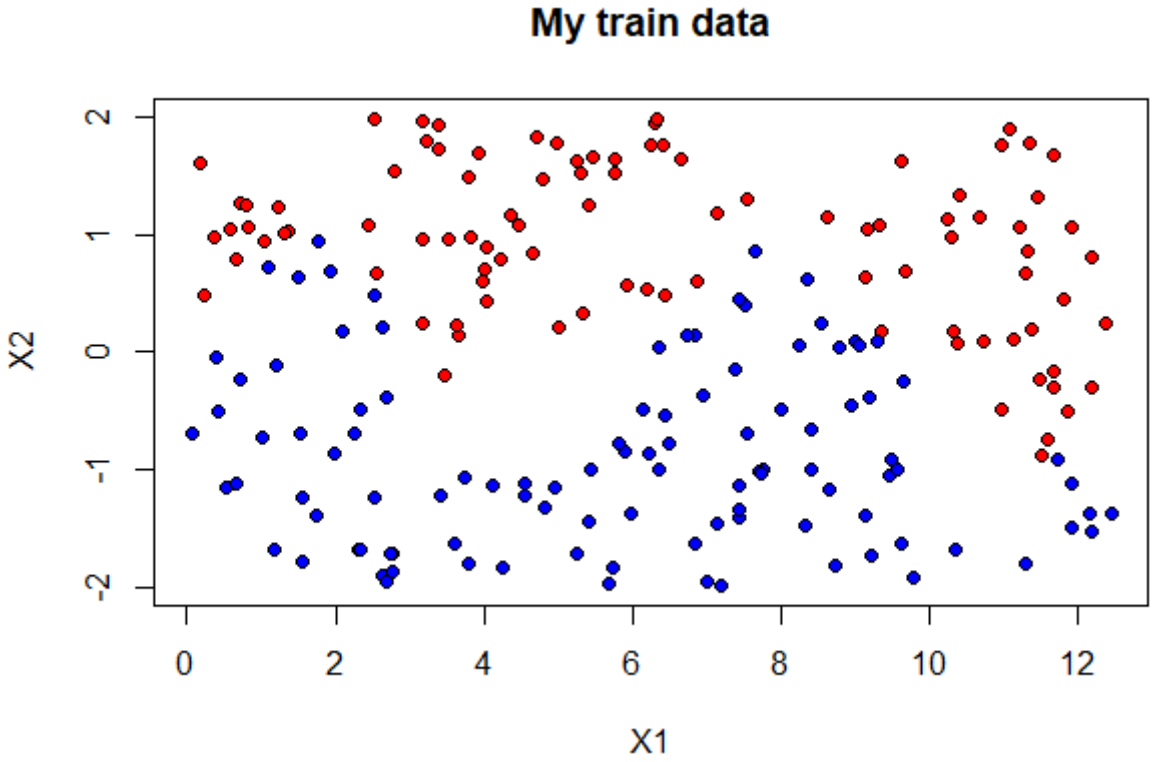
Код для всех реализуемых выше экспериментов представлен в Приложении 2.

# 4 Классификатор для svmdata4

Был построен классификатор по заранее сгенерированным обучающим и тестовым выборкам, хранящимся в файлах svmdata4.txt, svmdata4test.txt.

Было найдено оптимальное значение k = 8, обеспечивающее наименьшую ошибку классификации.

Данные используемого датасета представляются в следующем виде:




Код, используемый для работы с датасетом svmdata4 представлен в Приложении 3.

# 4 Классификатор для датасета «Титаник»

Разработан классификатор на основе метода ближайших соседей для данных Титаник (Titanic dataset).

Для обучения удалим из датасета данные, которые не несут обучающей информации, такие как: id, имя пассажира, каюта, порт посадки и номер билета.

Далее было определено оптимальное значение k ближайших соседей: 37.

Для построения классификатора необходимо, чтобы все данные были заполнены, поэтому для параметров Age и Fire пропуски были заполнены средними значениями.

Затем был построен классификатор на основе метода k ближайших соседей с параметрами: k = 37, kernel = «optimal», distance = 2. Точность данного классификатора составила ≈ 79%.

Код, на основе которого был построен классификатор для датасета «Титаник» приведен в Приложении 4.

# Приложение 1. Исследование точности классификатора k ближайших соседей от объема данных

```
##КРЕСТИКИ-НОЛИКИ
library(kknn)
A_raw <- read.table("Tic_tac_toe.txt", sep = ",", stringsAsFactors = TRUE)
#Число строк в базе
n <- dim(A_raw)[1]
# Устанавливаем базу генерации случайных чисел и рандомизируем выборку
set.seed(12345)
A_rand <- A_raw[ order(runif(n)), ]
#Разделим данные на обучающие и тестирующие (90% обучающая)
nt <- as.integer(n*0.9)
A_train <- A_rand[1:nt, ]
A_test <- A_rand[(nt+1):n, ]
#Процент классов из V10 в обучающей и тестовой выборке
prop.table(table(A_train$V10))
prop.table(table(A_test$V10))
#Строим классификатор (k = 7, distance = 2, kernel = "optimal")
A_classifier <- kknn(V10~., A_train, A_test, k = 7, distance = 2, kernel = "optimal")
A_predicted <- fitted(A_classifier)
tab1 = table(A_predicted, A_test$V10)
#вычисляем точность классификации
accuracy = (tab1[1,1] + tab1[2,2]) / (tab1[1,1] + tab1[2,2] + tab1[1,2] + tab1[2,1])
tab1
accuracy

##СПАМ
library(kernlab)
library(kknn)
data(spam)
#Случайным образом выбираем 10% сообщений для тестирования
idx <- sample(1:dim(spam)[1], 4601*0.1);
spamtrain <- spam[-idx, ];
spamtest <- spam[idx, ];
#Обучаем классификатор
A_classifier <- kknn(type ~ ., spamtrain, spamtest, k = 7, distance = 2, kernel = "optimal")
predict <- fitted(A_classifier)
tab2 = table(predict, spamtest$type)
#вычисляем точность классификации
accuracy2 = (tab2[1,1] + tab2[2,2]) / (tab2[1,1] + tab2[2,2] + tab2[1,2] + tab2[2,1])
tab2
accuracy2
```

# Приложение 2. Код, используемый для построения классификатора на основе датасета «Glass»

```
##GLASS
library(kknn)
library(rcompanion)
data(glass)
glass <- glass[, -1]
#Число строк в базе
n <- dim(glass)[1]
# Устанавливаем базу генерации случайных чисел и рандомизируем выборку
set.seed(12345)
Glass_rand <- glass[ order(runif(n)), ]
#Разделим данные на обучающие и тестирующие (80% обучающая)
nt <- as.integer(n*0.8)
Glass_train <- Glass_rand[1:nt, ]
Glass_test <- Glass_rand[(nt+1):n, ]

Depk = matrix(c(1:10, 1:10),nrow = 10, ncol = 2, byrow = TRUE)
#Строим классификатор для выявления зависимости ошибки от k (distance = 2, kernel =
"optimal")
for (k in 1:10){
Glass_classifier <- kknn(Type ~ ., Glass_train, Glass_test, k = k, distance = 2, kernel =
"optimal")
fit <- fitted(Glass_classifier)
tab3 = table(fit, Glass_test$Type)
sum1 = 0
sum2 = 0
#вычисляем суммы числителя и знаменателя для нахождения точности классификации
for(i in 1:6){
  sum1 = sum1 + tab3[i,i]
  for(j in 1:6){
    sum2 = sum2 + tab3[i,j]
  }
}
#вероятность ошибочной классификации
mistake3 = 1 - sum1/sum2
Depk[k, 1] = k
Depk[k, 2] = mistake3
}
plot(Depk[,1],Depk[,2], col="red", ylab="Ошибка классификации", xlab="k", pch = 19,
type="o")


Depd = matrix(c(1:10, 1:10),nrow = 10, ncol = 2, byrow = TRUE)
```

```
#Строим классификатор для выявления зависимости ошибки от distance (k = 7, kernel =
"optimal")
for (d in 1:10){
  Glass_classifier <- kknn(Type ~ ., Glass_train, Glass_test, k = 7, distance = d, kernel =
"optimal")
  fit <- fitted(Glass_classifier)
  tab3 = table(fit, Glass_test$Type)
  sum1 = 0
  sum2 = 0
  #вычисляем суммы числителя и знаменателя для нахождения точности классификации
  for(i in 1:6){
    sum1 = sum1 + tab3[i,i]
    for(j in 1:6){
      sum2 = sum2 + tab3[i,j]
    }
  }
  #точность классификации
  accuracy3 = sum1/sum2
  Depd[d, 1] = d
  Depd[d, 2] = accuracy3
}
plot(Depd[,1],Depd[,2], col="blue", ylab="Точность классификации", xlab="distance", pch =
19, type="o")

#определеям тип стекла
example <- 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)
Glass_classifier <- kknn(Type~., Glass_train, example,  k = 7, distance = 2, kernel = "optimal")
#выведем матрицу вероятносте прогнозируемых классов
Glass_classifier$prob

#определим наименнее значимый признак для классификации
library(dplyr)
glass
#удаляем признак RI
Newglass <- glass %>% select(-Fe)
n <- dim(Newglass)[1]
NewGlass_rand <- Newglass[ order(runif(n)), ]
#Разделим данные на обучающие и тестирующие (80% обучающая)
nt <- as.integer(n*0.8)
NewGlass_train <- NewGlass_rand[1:nt, ]
NewGlass_test <- NewGlass_rand[(nt+1):n, ]
NewGlass_classifier <- kknn(Type ~ ., NewGlass_train, NewGlass_test, k = 7, distance = 2,
kernel = "optimal")
fit <- fitted(NewGlass_classifier)
tab3 = table(fit, NewGlass_test$Type)
```

```
sum1 = 0
sum2 = 0
#вычисляем суммы числителя и знаменателя для нахождения точности классификации
for(i in 1:6){
  sum1 = sum1 + tab3[i,i]
  for(j in 1:6){
    sum2 = sum2 + tab3[i,j]
  }
}
#точность классификации
accuracy3 = sum1/sum2
accuracy3
```

# Приложение 3. Код нахождения оптимального k для обучающего множества svmdata4

```
##SVMDADA4
svmdata4_train <- read.table("svmdata4.txt", sep = "\t", stringsAsFactors = TRUE)
svmdata4_test <- read.table("svmdata4test.txt", sep = "\t", stringsAsFactors = TRUE)
fit.train <- train.kknn(Colors ~ ., svmdata4_train, kmax = 15, distance = 2,
                        kernel = "optimal")
plot(svmdata4_train$X1, svmdata4_train$X2, pch=21, xlab = "X1", ylab="X2",
    bg=c("red","blue") [unclass(svmdata4_train$Colors)], main="My train data")

#Лучшее значение k = 8
fit.train
```

# Приложение 4. Построение классификатора на основе метода k ближайших соседей для датасета «Титаник»

```
##Титаник
library(kknn)
library(dplyr)
#Загрузка обучающей выборки
Titanic_train <- read.csv("Titanic_train.csv", header = TRUE, sep = ",", dec = ".",
                          stringsAsFactors = FALSE)
#Загрузка тестовой выборки
Titanic_test <- read.csv("Titanic_test.csv", header = TRUE, sep = ",", dec = ".",
                         stringsAsFactors = FALSE)
#Удаление столбцов, ненесущих информации для обучения
Titanic_train <- Titanic_train %>% select(-PassengerId)
Titanic_train <- Titanic_train %>% select(-Name)
Titanic_train <- Titanic_train %>% select(-Ticket)
Titanic_train <- Titanic_train %>% select(-Cabin)
Titanic_train <- Titanic_train %>% select(-Embarked)
Titanic_train

#Нахождение оптимального k
fit.train <- train.kknn(Survived ~ ., Titanic_train, kmax = 40, distance = 2,
                        kernel = "optimal")
fit.train

#Заполнение пропусков
#Нахождение средних значений
Age_mean <- summary(Titanic_test$Age)[4]
Fare_mean <- summary(Titanic_test$Fare)[4]
for(i in 1:dim(Titanic_test)[1]){
  if(is.na(Titanic_test$Age[i])){
    Titanic_test$Age[i]=Age_mean
  }
  if(is.na(Titanic_test$Fare[i])){
    Titanic_test$Fare[i]=Fare_mean
  }
}

Titanic_classifier <- kknn(Survived ~ ., Titanic_train, Titanic_test, k = 37, distance = 2, kernel =
"optimal")
fit <- round(fitted(Titanic_classifier))
Gen_sub = read.csv("gender_submission.csv", header = TRUE, sep = ",", dec = ".",
                   stringsAsFactors = FALSE)
```

```
tab4 = table(fit, Gen_sub$Survived)
tab4
accuracy4 = (tab4[1,1] + tab4[2,2]) / (tab4[1,1] + tab4[2,2] + tab4[1,2] + tab4[2,1])
accuracy4
```