Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого
—
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

ЛАБОРАТОРНАЯ РАБОТА №2

«Метод поиска ближайшего соседа»

по дисциплине «Машинное обучение, часть1»

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

<подпись>

Проверил: Л.В. Уткин
д.т.н., профессор

<подпись>

Санкт-Петербург
2022

Содержание

1. Цель работы

Исследовать метод k ближайших соседей пакета *kknn* языка R, выполнив поставленные задачи и проанализировав результаты.

2. Формулировка задания

1. Исследуйте, 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. Постройте 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

3. Ход работы

Задание №1

Размер датасета «крестики-нолики» - 958. Так как число классов четное, нельзя брать четное число ближайших соседей. Выбор параметра k противоречив. С одной стороны, увеличение его значения повышает достоверность классификации, но при этом границы между классами становятся менее четкими.

Для данного примера было взято от 10% исходной выборки до 95% с шагом в 1%. Для того, чтобы испытание было более точным, для каждой процентной доли бралось по 10 испытаний, а затем полученные результаты усреднялись.

Метод ближайших соседей был использован с параметрами:

- Число ближайших соседей – k=7 (по умолчанию)
- Ядро – “triangular”
- Параметр расстояния Минковского - 1

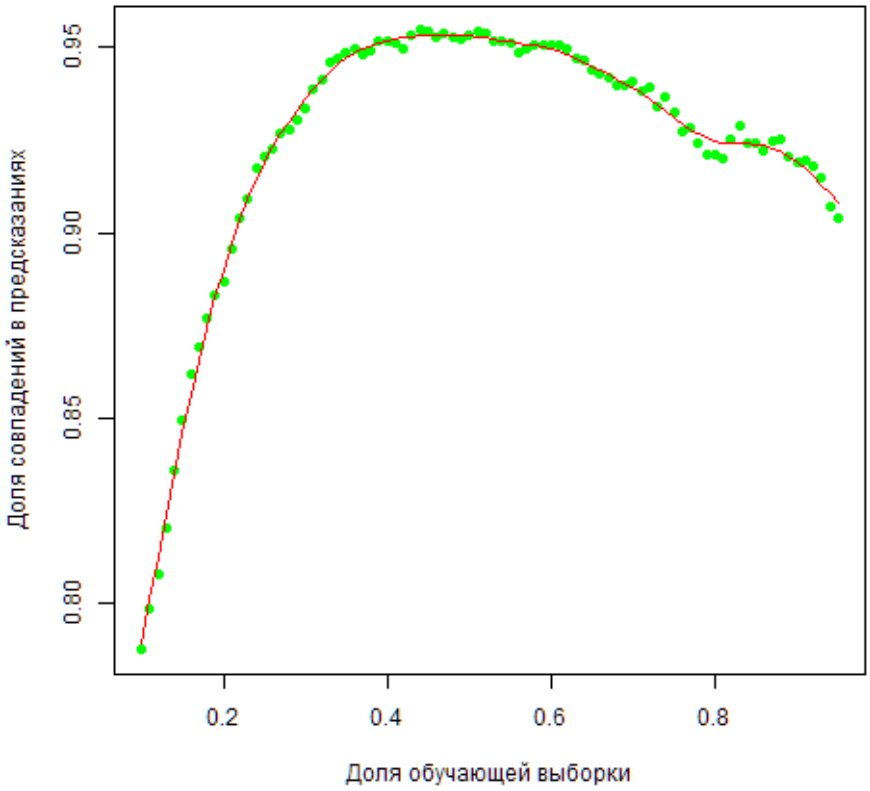

Рисунок 1. Зависимость точности классификации от объема обучающих/тестовых данных в примере «Крестики-нолики» с k=7

По рисунку 1 можно выделить зоны недообучения в 10-25% и переобучения – 80-95%. По графику отчетливо видно, что недообучение оказывает наиболее сильное негативное влияние на результат, чем переобучение.

Средний процент успешных предсказаний для данного датасета составил 92.4%, максимальный – 95.5%.

Для эксперимента значение k было изменено и для каждого объема выборки считается как $k = \sqrt{n}$.

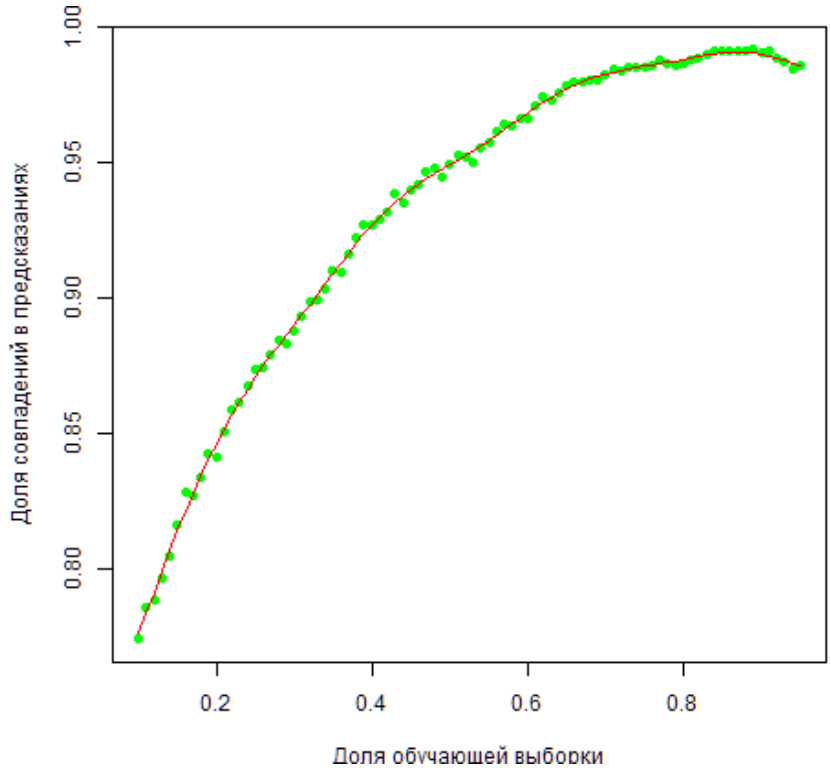

Рисунок 2. Зависимость точности классификации от объема обучающих/тестовых данных в примере «Крестики-нолики» с $k = \sqrt{n_i}$

Удалось добиться лучших показателей - средний процент успешных предсказаний составил 93.3%, максимальный – 99.1%, который достигнут при соотношении обучающей и тестовой выборки 90%/10%.

Можно сказать, что модель KNN справилась с задачей лучше, чем наивный Байесовский классификатор из лабораторной №1.

Размер датасета «спам» - 4601.

Для примера «спам» было взято от 10 до 1000 экземпляров обучающих данных с шагом 10. Данные также усреднялись после 10 испытаний для каждого количества экземпляров.

Метод ближайших соседей был использован с параметрами:

- Число ближайших соседей – k=7 (по умолчанию)
- Ядро – “triangular”
- Параметр расстояния Минковского - 1

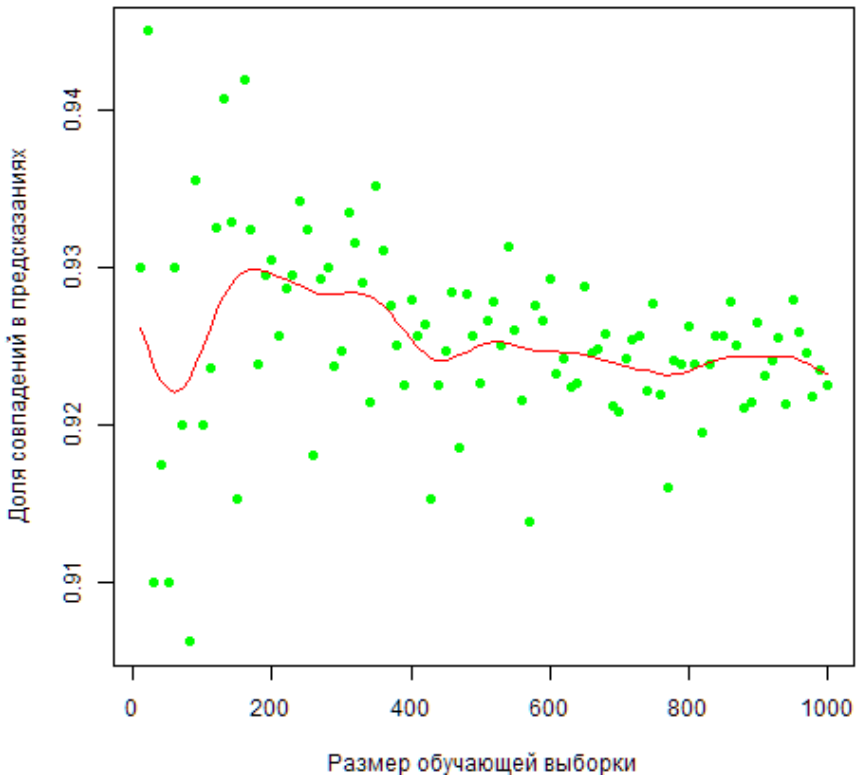

Рисунок 3. Зависимость точности классификации от объема обучающих/тестовых данных в примере «Спам»

По рисунку 3 можем заметить, что на данном датасете результаты заметно разбросаны, особенно на участке 10-400. Процент успешных предсказаний для данного датасета составил 92.5%.

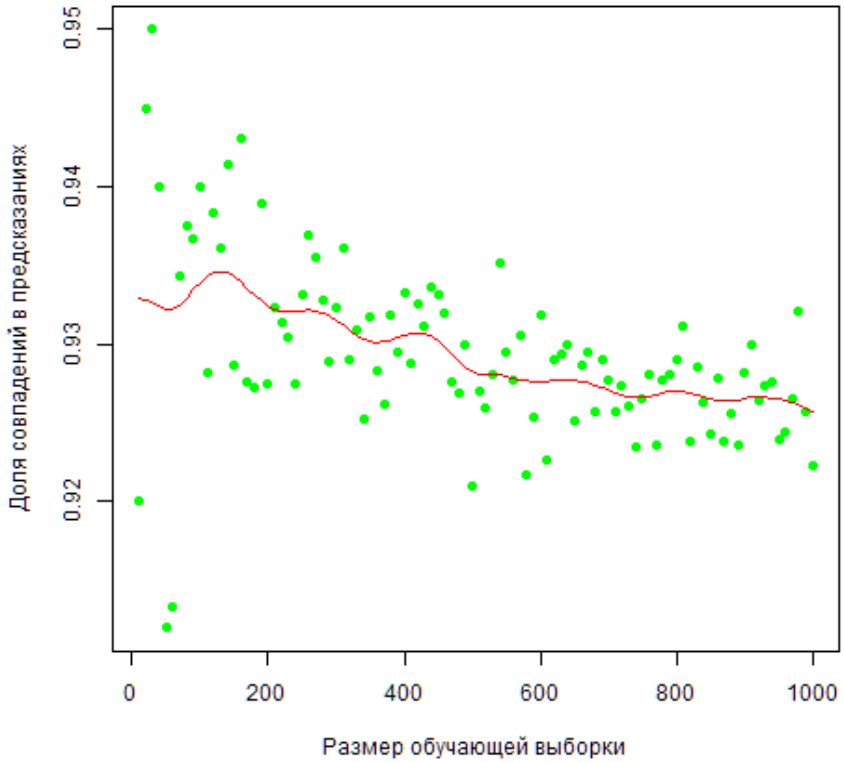

Рисунок 4. Зависимость точности классификации от объема обучающих/тестовых данных в примере «Спам» с $k = \sqrt{n_i}$

Для эксперимента значение k было изменено и для каждого объема выборки считается как $k = \sqrt{n}$. Средний процент успешных предсказаний составил 93%, максимальный – 95%. Результаты на рисунке 4 по сравнению с рисунком 3 стали менее разбросаны.

Также и для этого датасета можно сказать, что модель KNN справилась с задачей лучше, чем наивный Байесовский классификатор из лабораторной №1.

Задание №2

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

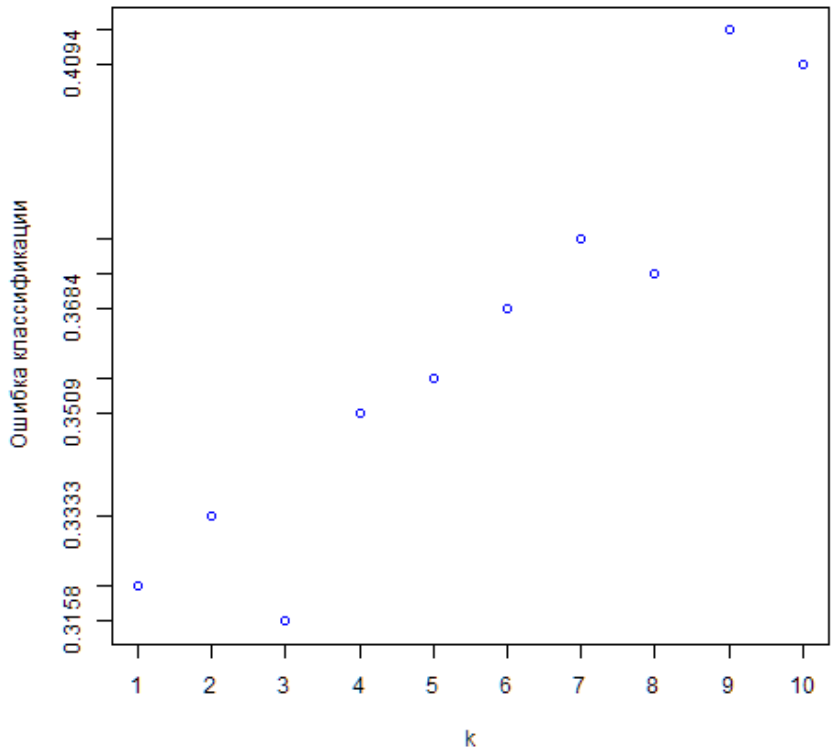

Рисунок 5. Зависимость ошибки классификации от числа ближайших соседей для ядра Rectangular

На рисунке 5 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Rectangular. Для данного ядра оптимальным числом ближайших соседей является k=3, где ошибка составляет 31.58%.

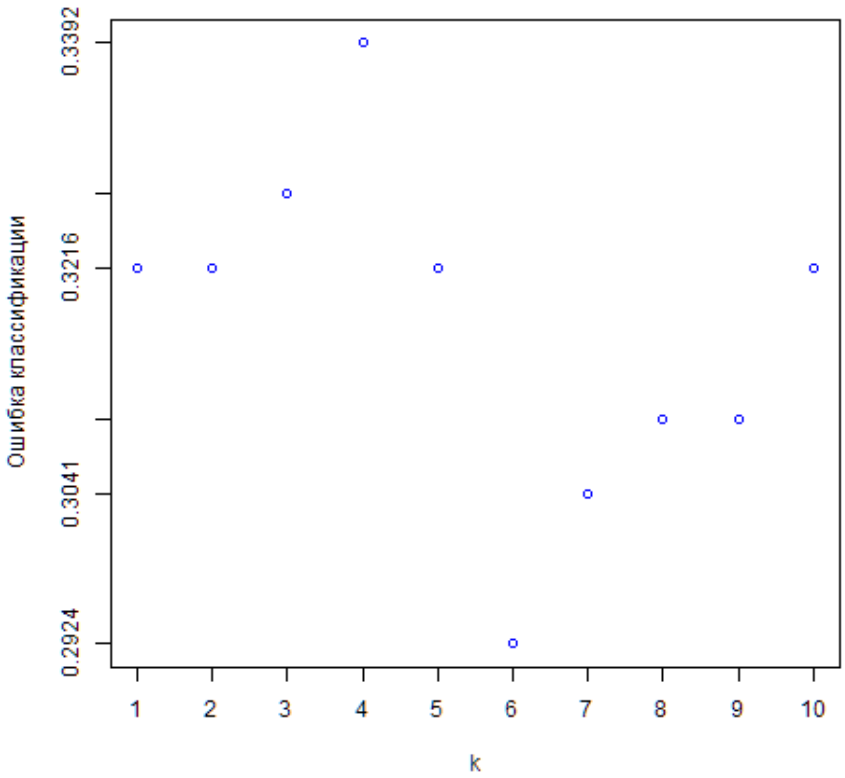

Рисунок 6. Зависимость ошибки классификации от числа ближайших соседей для ядра Triangular

На рисунке 6 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Triangular. Для данного ядра оптимальным числом ближайших соседей является k=6, где ошибка составляет 29,24%.

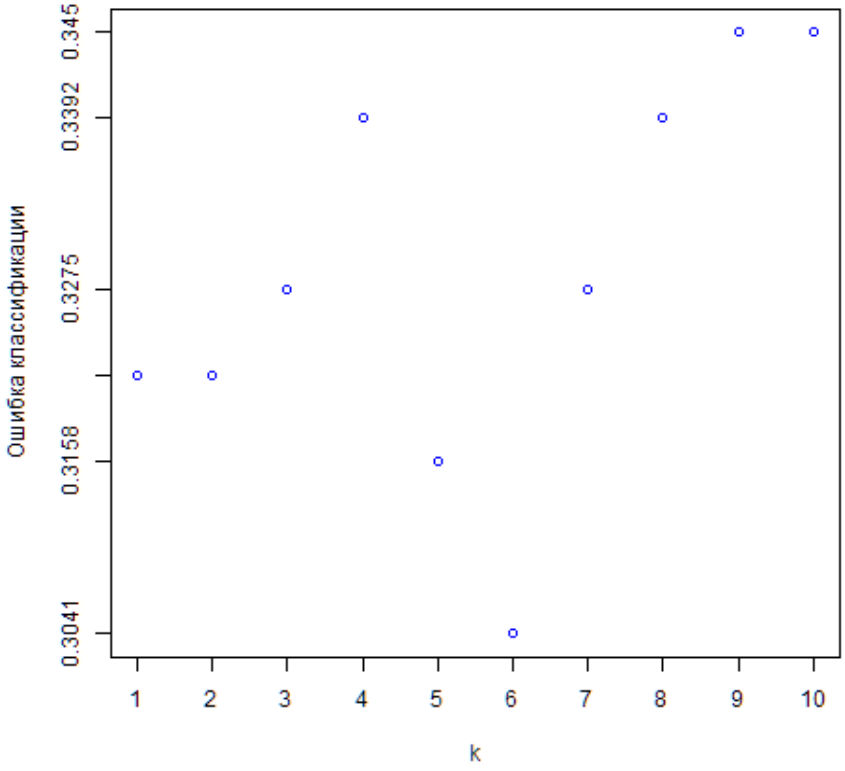

Рисунок 7. Зависимость ошибки классификации от числа ближайших соседей для ядра Epanechnikov

На рисунке 7 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Epanechnikov. Для данного ядра оптимальным числом ближайших соседей является k=6, где ошибка составляет 30,41%.

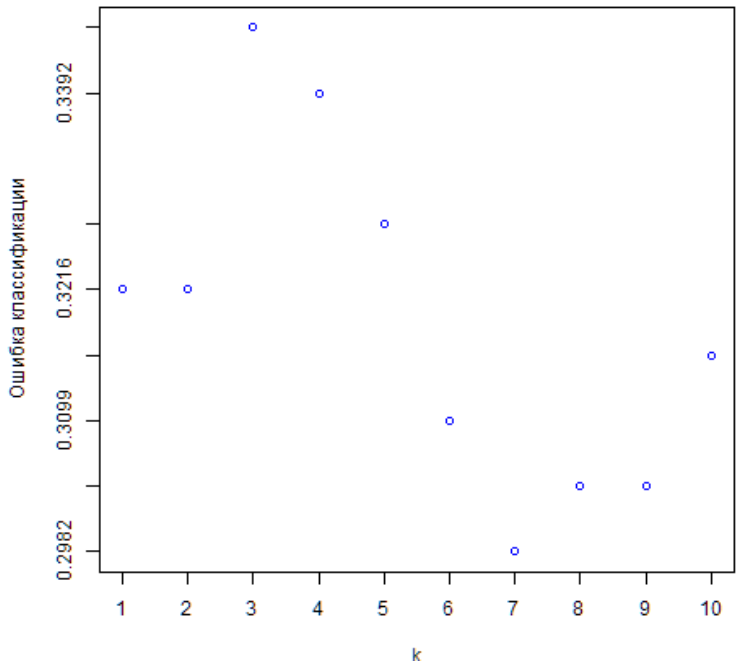

Рисунок 8. Зависимость ошибки классификации от числа ближайших соседей для ядра Biweight

На рисунке 8 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Biweight. Для данного ядра оптимальным числом ближайших соседей является k=7, где ошибка составляет 29,82%.

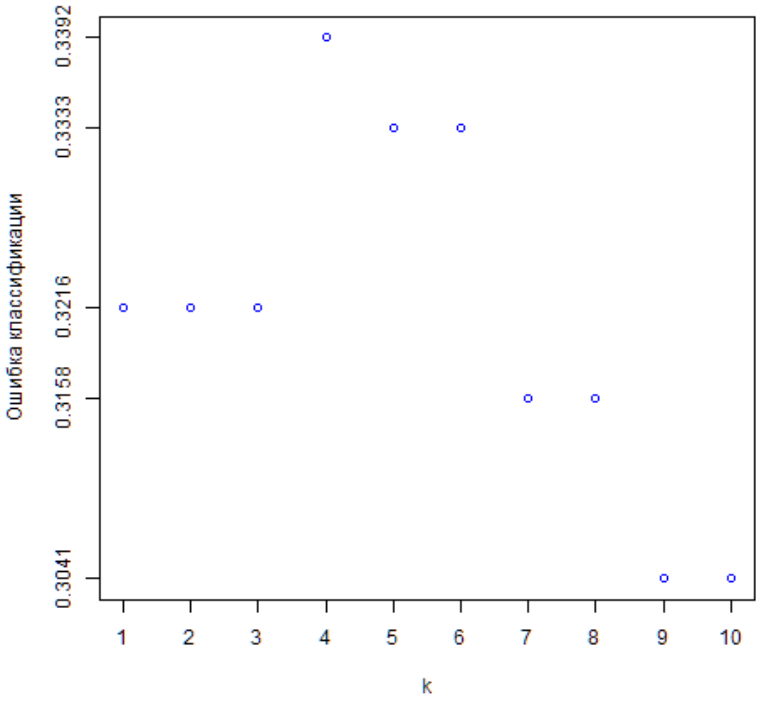

Рисунок 9. Зависимость ошибки классификации от числа ближайших соседей для ядра Triweight

На рисунке 9 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Triweight. Для данного ядра оптимальным числом ближайших соседей является k=9, 10, где ошибка составляет 30,41%.

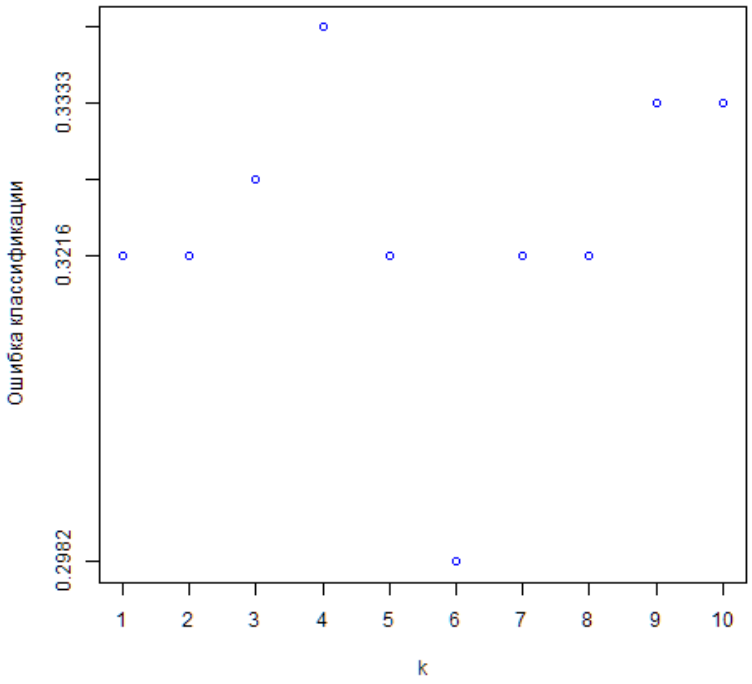

Рисунок 10. Зависимость ошибки классификации от числа ближайших соседей для ядра Cos

На рисунке 10 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Cos. Для данного ядра оптимальным числом ближайших соседей является k=6, где ошибка составляет 29,82%.

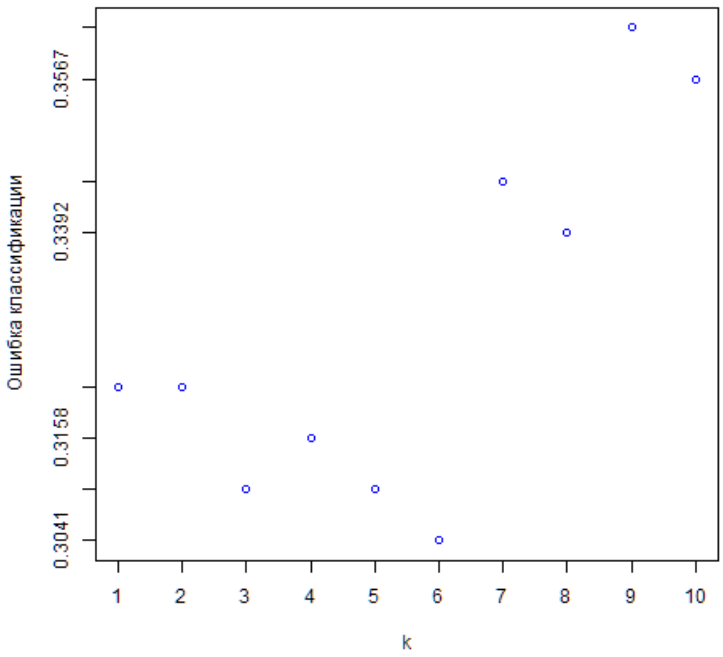

Рисунок 11. Зависимость ошибки классификации от числа ближайших соседей для ядра Inv

На рисунке 11 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Inv. Для данного ядра оптимальным числом ближайших соседей является k=6, где ошибка составляет 30,41%.

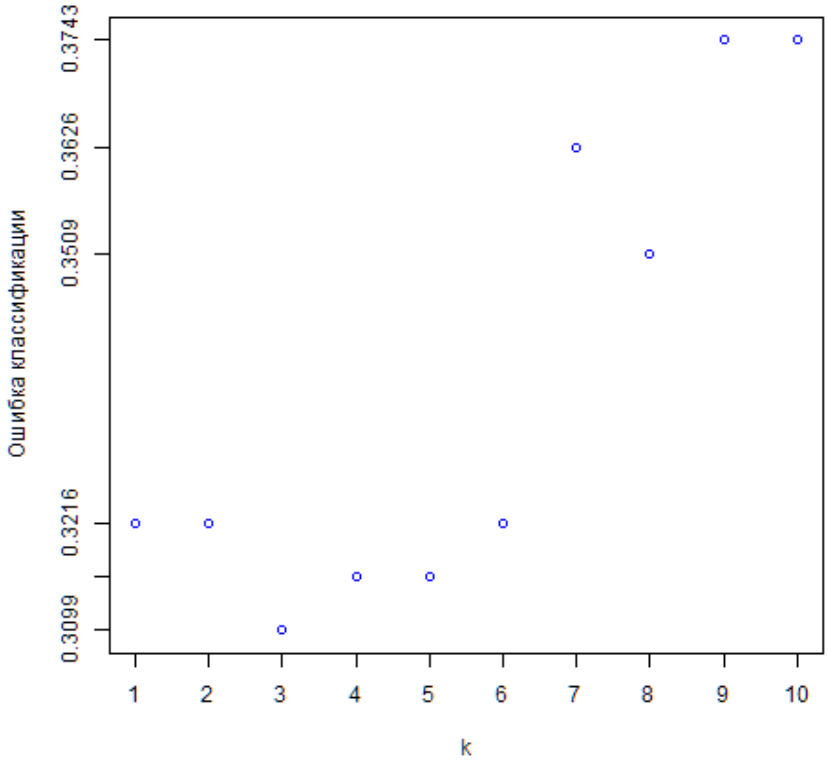

Рисунок 12. Зависимость ошибки классификации от числа ближайших соседей для ядра Gaussian

На рисунке 12 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Gaussian. Для данного ядра оптимальным числом ближайших соседей является k=3, где ошибка составляет 30,99%.

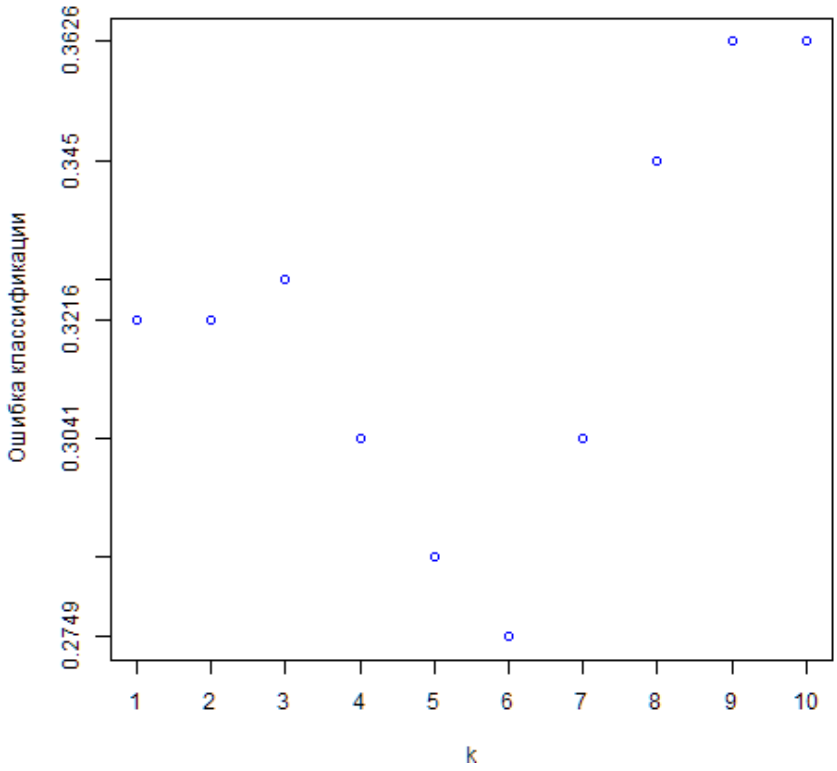

Рисунок 13. Зависимость ошибки классификации от числа ближайших соседей для ядра Rank

На рисунке 13 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Rank. Для данного ядра оптимальным числом ближайших соседей является k=6, где ошибка составляет 27,49%.

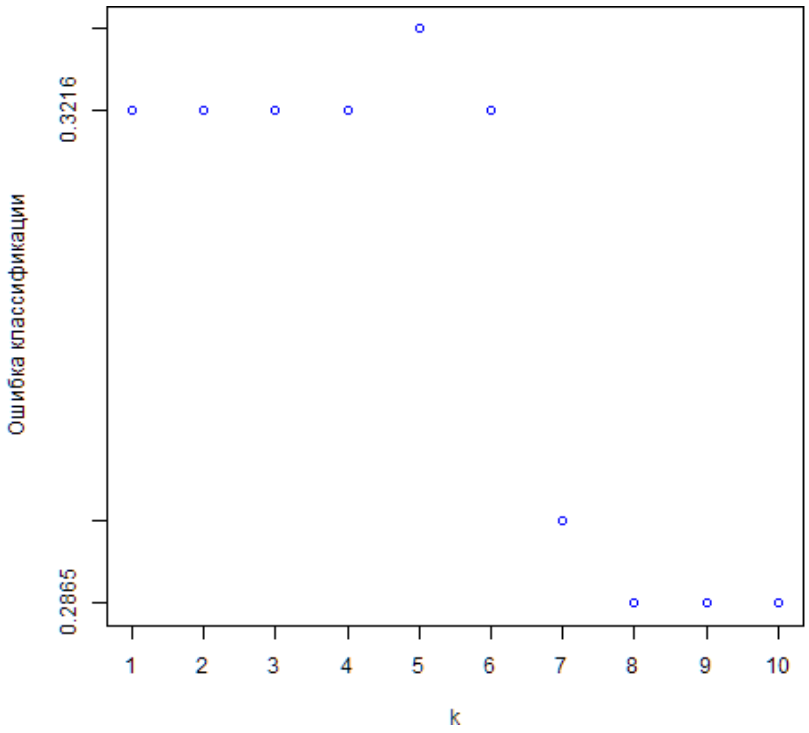

Рисунок 14. Зависимость ошибки классификации от числа ближайших соседей для ядра Optimal

На рисунке 14 изображена зависимость ошибки классификации от числа ближайших соседей k для ядра Optimal. Для данного ядра оптимальным числом ближайших соседей является k=8, 9, 10, где ошибка составляет 28,65%.

В результате проведенных экспериментов можем заметить, что наилучшими ядрами для классификатора rank с числом соседей k=6 (с точностью ≈73%) и optimal с числом соседей k=8, 9, 10 (с точностью ≈71%).

2. Исследовать, как параметр distance влияет на точность классификации.

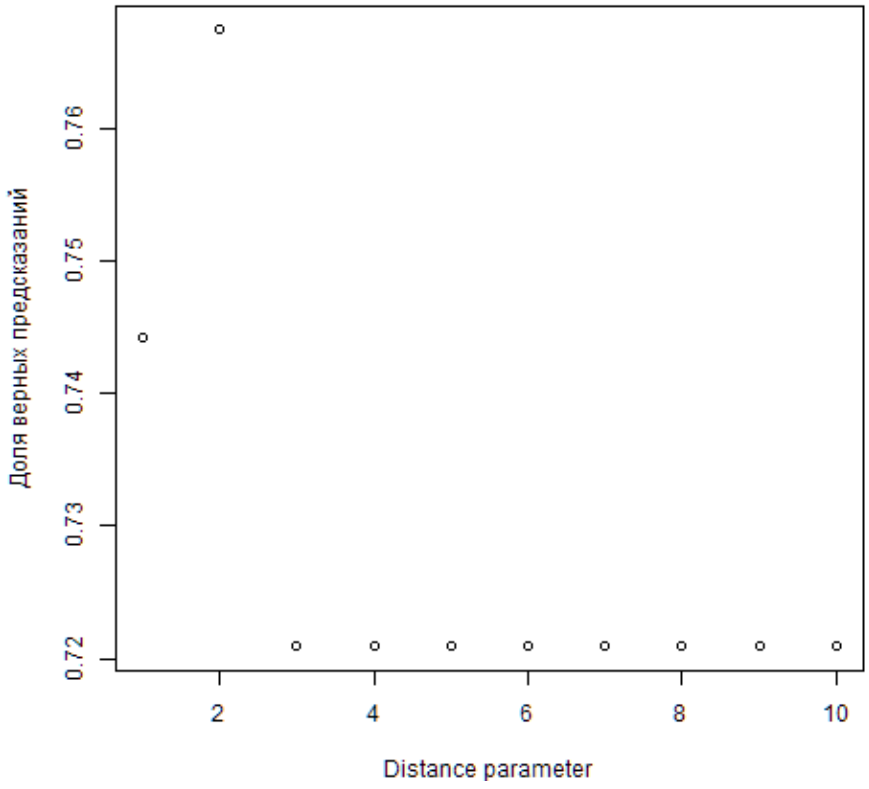

Рисунок 15. Зависимость точности предсказаний от параметра distance

Наиболее оптимальным расстоянием является 2, его точность сильно выше, чем у других значений параметра - ≈77%. При увеличении значения параметра точность классификации падает

3. 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

Класс	1	2	3	**5**	6	7
Вероятность	0.005933314	0	0.03685387	**0.9572128**	0	0

Представленный экземпляр относится к классу 5 с вероятностью ≈96%.

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

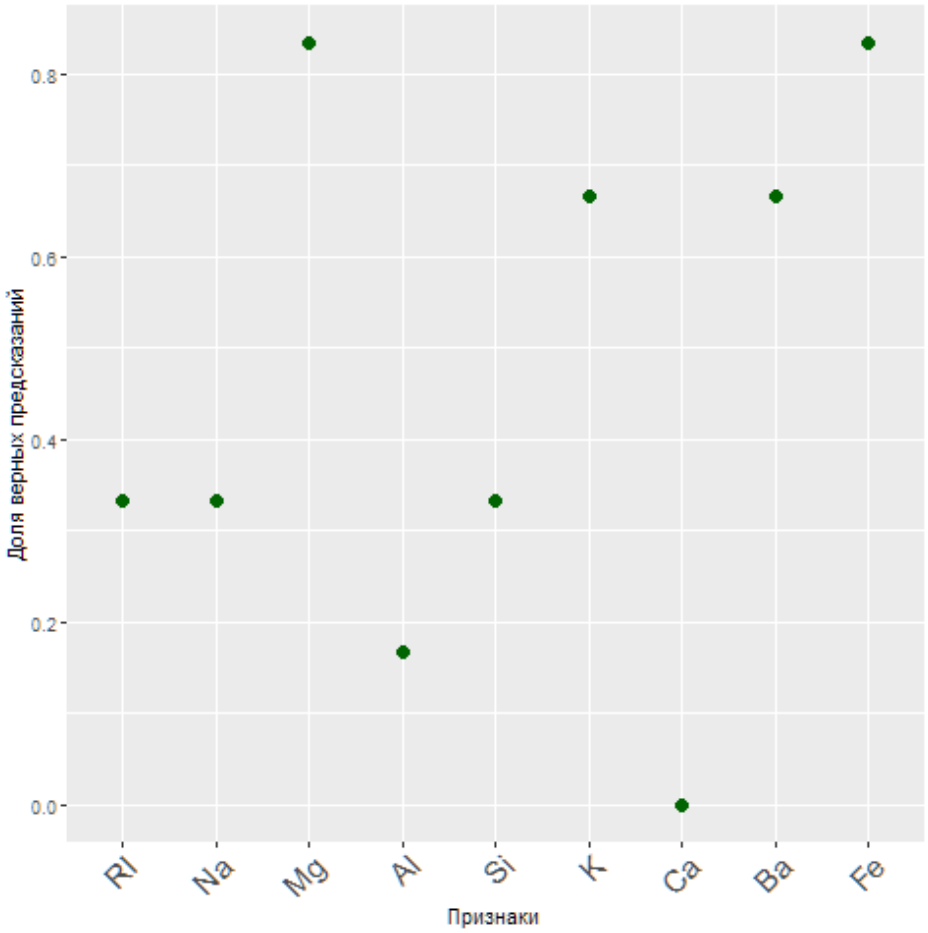

Рисунок 13. Вклад признаков в правильность предсказаний

На рисунке 16 можно видеть, какие из параметров влияют на точность классификации. Наиболее значимым является Ca, без него классификация невозможна. Далее довольно большой вклад вносят AI, а также Na, RI и Si.

Задание №3

Визуализируем обучающую выборку.

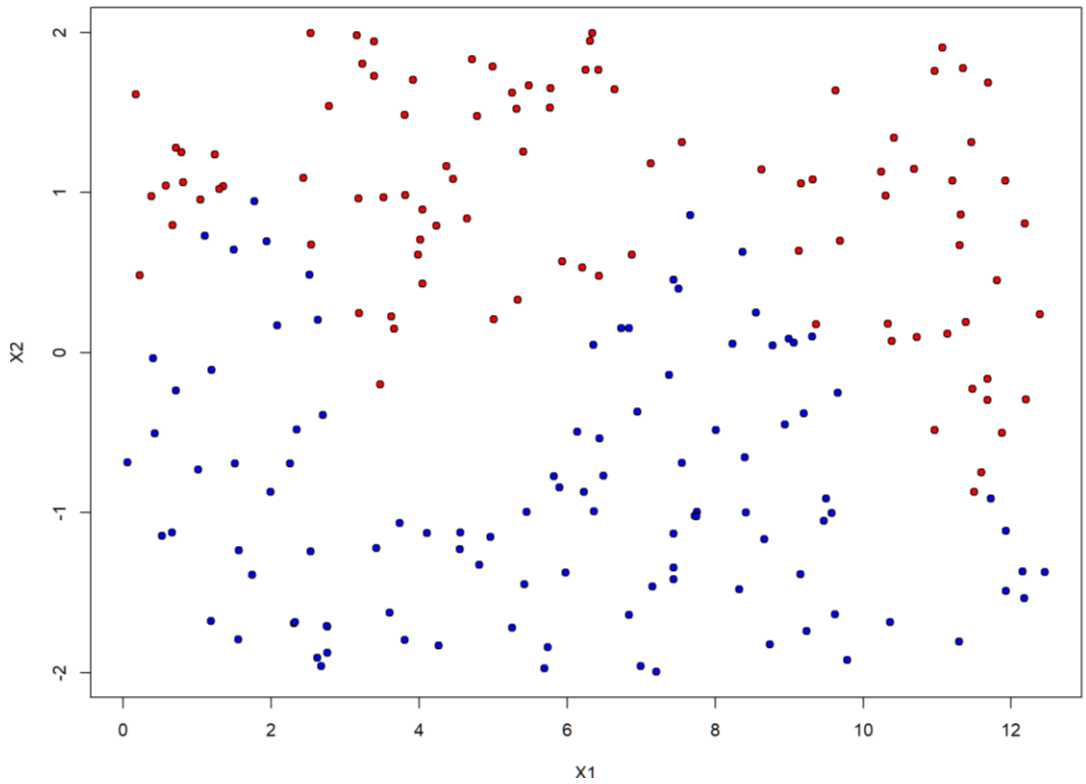

Рисунок 14. Визуализация обучающей выборки

Для нахождения оптимального k, которое обеспечивает наименьшую ошибку классификации воспользуемся кросс-валидацией и зафиксируем значения параметров distance (2) и kernel (optimal).

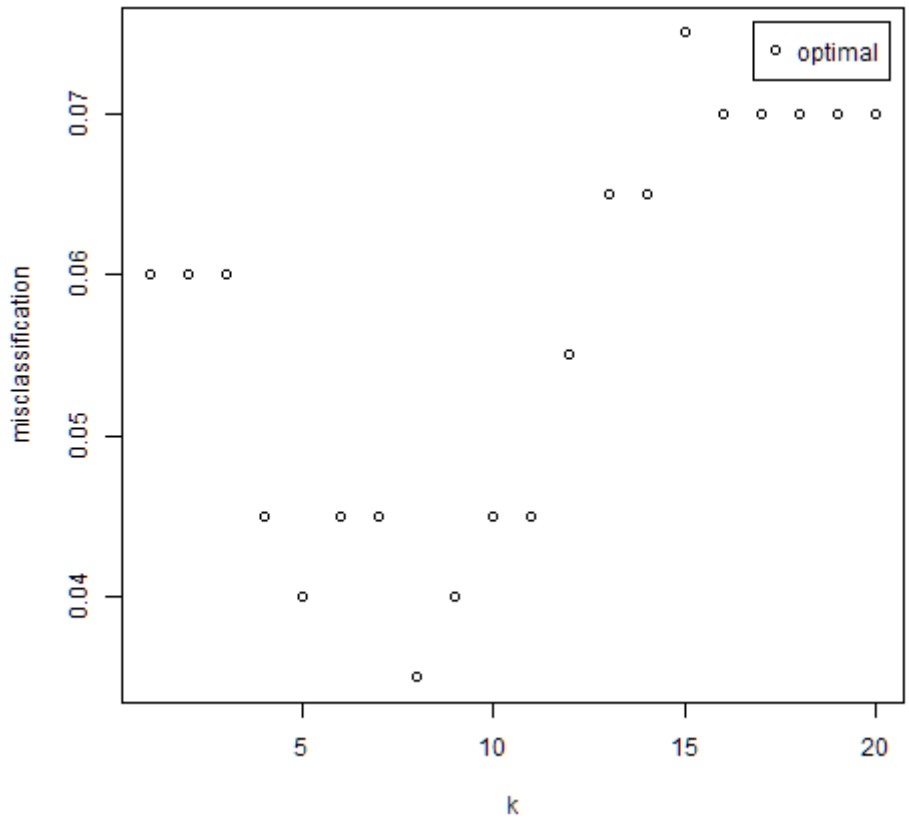

Рисунок 18. Зависимость ошибки от числа соседей

По рисунку 17 видно, что можно отследить довольно явную границу между классами. Это значит, что вероятность ошибки должна быть маленькой. Действительно, по рисунку 18 можно заметить, что процент ошибки довольно мал (до 6%). Наименьшее значение ошибки достигается при 8 соседях – 3.5%.

Задание №4

Исходная подготовка данных была взята из лабораторной №1.

В результате при помощи кросс-валидации была построена модель с ядром «optimal» и параметром расстояния 1. Наиболее близким к реальному результату оказался вариант модели с числом соседей 25.

```
Type of response variable: continuous
minimal mean absolute error: 0.2137243
Minimal mean squared error: 0.1301933
Best kernel: optimal
Best k: 25
```

Рисунок 19. Результаты кросс-валидации для датасета Titanic

Реальное соотношение данных в выборке – 61.6%/38.4%. Соотношения, полученные классификатором – 63.2%/36.8%. Погрешность составила 1.6%.

4. Вывод

В результате выполнения данной работы был изучен на практике метод ближайших соседей. Были исследованы различный зависимости гиперпараметров друг от друга. Так же исследовано влияние объема выборки на точность классификации. Несмотря на свою относительную алгоритмическую простоту, метод ближайших соседей показывает хорошие результаты.

Достоинства:

- Простота реализации.
- Классификацию, проведенную алгоритмом, легко интерпретировать путем предъявления пользователю нескольких ближайших объектов.

Недостатки:

- Необходимость хранения обучающей выборки целиком.
- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки.
- Главным его недостатком является высокая вычислительная трудоемкость, которая увеличивается квадратично с ростом числа обучающих примеров.

Приложение 1

```
# Задание 1 ##################################################
path <- "C:\\Users\\Софья\\Desktop\\1_курс_1_семестр\\Машинное
обучение\\Лабораторная 2. KNN\\"
install.packages("kknn")
library(kknn)
# получили датафрейм
A_raw <- read.table(paste(path, "Tic_tac_toe.txt", sep = ""), sep = ",", stringsAsFactors = TRUE)
# количество строк
n <- nrow(A_raw)

# задаем параметры обучения
learning <- seq(0.1, 0.95, 0.01)
res_vec <- vector()
for (learn_ratio in learning) {
  nt <- as.integer(n * learn_ratio)
  exp_res <- vector()
  set.seed(1121)
  for (j in 1:10) {
    A_rand <- A_raw[order(runif(n)), ]
    A_train <- A_rand[1:nt, ]
    A_test <- A_rand[(nt + 1):n, ]

    A_classifier <- kknn(V10 ~ .,
                         A_train,
                         A_test,
                         k=round(sqrt(nt), digits = 0),
                         distance = 1,
                         kernel = "triangular")

    res <- table(A_classifier$fitted.values, A_test$V10)

    exp_res <- append(exp_res, (res[1, 1] + res[2, 2]) / sum(res))
  }
  res_vec <- append(res_vec, mean(exp_res))
}

x <- learning
y <- res_vec
max(y)
mean(y)
spline = smooth.spline(x, y, spar = 0.6)
png(paste(path, "Крестики-нолики1.png"))
plot (x, y,
  xlab = "Доля обучающей выборки" ,
  ylab = "Доля совпадений в предсказаниях",
  col = "green",
  pch = 19,
  main = "Крестики-нолики, KNN"
```

```
)
lines(spline, col = "red")
dev.off()
#----------------------------------------------------------------
library(kernlab)
data(spam)

learning <- seq(10, 1000, 10)
res_vec <- vector()
for (learn_ratio in learning) {
  exp_res <- vector()
  for (j in 1:10)
  {
    idx <- sample(1:nrow(spam), learn_ratio)
    spamtrain <- spam[-idx, ]
    spamtest <- spam[idx, ]

    model <- kknn(type ~ .,
                  train = spamtrain,
                  test = spamtest,
                  distance = 1,
                  k=round(sqrt(learn_ratio), digits = 0),
                  kernel = "triangular")

    res <- table(model$fitted.values, spamtest$type)

    exp_res <- c(exp_res, (res[1, 1] + res[2, 2]) / sum(res))
  }
  res_vec <- c(res_vec, mean(exp_res))
}

x <- learning
y <- res_vec
spline = smooth.spline(x, y, spar = 0.6)
png(paste(path, "Спам, KNN (1).png"))
plot (x, y,
  xlab = "Размер обучающей выборки",
  ylab = "Доля совпадений в предсказаниях",
  col = "green",
  pch = 19,
  main = "Спам, KNN"
)
lines(spline, col = "red")
dev.off()

max(y)
mean(y)
```

Приложение 2

```
# Задание 2 ##########################################################
#-------------------------------------------------------------
library(kknn)
library(kernlab)
data("glass")

# Id number
glass <- glass[, -1]
glass$Type

learn_ratio = 0.8
n <- nrow(glass)
nt <- as.integer(n * learn_ratio)
glass_rand <- glass[order(runif(n)), ]
glass_train <- glass_rand[1:nt, ]
glass_test <- glass_rand[(nt + 1):n, ]

kernels <- c(
  "rectangular",
  "triangular",
  "epanechnikov",
  "biweight",
  "triweight",
  "cos",
  "inv",
  "gaussian",
  "rank",
  "optimal")

for (k in kernels) {
  best.rect <- train.kknn(Type ~ .,
                          glass_train,
                          kmax = 10,
                          distance = 2,
                          kernel = k)
  mis.rect <- best.rect$MISCLASS
  y<-mis.rect[!duplicated(mis.rect)]
  i <- order(mis.rect)

  png(paste(path, paste(k, "kernels.png")))
  plot(seq(1, 10, 1),
      mis.rect,
      xlab="k",
      ylab = "Ошибка классификации",
      col="blue",
      axes = FALSE)
  axis(side = 1,
      at = seq(1, 10, 1),
```

```
      labels = seq(1, 10, 1))
  axis(side = 2,
      at = y[i],
      labels = round(y[i], digits = 4),
      las = 0)
  box()
  dev.off()
}

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

distances = 1:10
res_vec <- vector()
for (d in distances) {
  model <- kknn(
    Type ~ .,
    train = glass_train,
    test = glass_test,
    distance = d,
    k=6,
    kernel = "rank"
  )
  res <- table(model$fitted.values, glass_test$Type)
  res_vec <- append(res_vec, (sum(diag(res)) / sum(res)))
}

x <- distances
y <- res_vec
png(paste(path, "Distance parameter.png"))
plot(x, y, xlab = "Distance parameter", ylab = "Доля верных предсказаний")
dev.off()

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

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
)

res <- kknn(
  Type ~ .,
  train = glass_train,
```

```
  example,
  distance = 2,
  k = 6,
  kernel = "biweight")

print(res$fitted.values)
print(res$prob)

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

res_vec <- vector()
for (i in 1:length(example)) {
  test_ex <- example
  test_ex[1, i] = 0
  res <- kknn(
    Type ~ .,
    train = glass_train,
    test_ex,
    k=6,
    distance = 2,
    kernel = "rank"
  )
  res_vec <- append(res_vec, res$prob[4])
}

x <- 1:length(colnames(example))
y <- res_vec
df <- data.frame(x, y)
png(paste(path, "Признаки.png"))
ggplot(df) +
  geom_point(aes(x = x, y = y), color = "darkgreen", lwd = 3) +
  scale_x_discrete(limit = x, labels = colnames(example)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 15)) +
  labs(x = "Признаки", y = "Доля верных предсказаний")
dev.off()
```

Приложение 3

```
# Задание 3 ##########################################################
install.packages("ggplot2")
library(ggplot2)
SVM_Train <- read.table(paste(path, "svmdata4.txt", sep = ""),
                        sep = "\t",
                        stringsAsFactors = TRUE)
SVM_Test <- read.table(paste(path, "svmdata4test.txt", sep = ""),
                       sep = "\t",
                       stringsAsFactors = TRUE)

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

n <- nrow(SVM_Train)

png(paste(path, "Task3.png"))
ggplot(SVM_Train, aes(x = X1, y = X2, color = Colors)) +
  geom_point(size = 2) +
  theme_minimal()
dev.off()

model <- train.kknn(
  Colors ~ .,
  data = SVM_Train,
  #ks = 1:14,
  kmax=20,
  kernel = "optimal",
  distance = 2
)
png(paste(path, "Optimal.png"))
plot(model)
dev.off()
```

Приложение 4

```
# Задание 4 ##################################################
T_train<-read.csv(paste(path, "Titanic_train.csv", sep = ""),
                  stringsAsFactors = TRUE)
T_test<-read.csv(paste(path, "Titanic_test.csv", sep = ""),
                 stringsAsFactors = TRUE)
T_train[0,] #метки признаков

passengers <- T_test$PassengerId
#удаляем признаки PassengerId, Name,Ticket, Cabin, Embarked
T_train <- T_train[,-c(1,4,5,10,12,13)]
T_test <- T_test[,-c(1,3,4,9,11,12)]

#меняем пол на численные метки 0-м, 1-ж
T_train$Sex <- sapply(as.character(T_train$Sex), switch, 'male' = 0, 'female' = 1)
T_test$Sex <- sapply(as.character(T_test$Sex), switch, 'male' = 0, 'female' = 1)

#записываем вместо значения NaS среднее значение Age
train_age<-na.omit(T_train$Age)
T_train$Age[is.na(T_train$Age)] <- mean(train_age)

test_age<-na.omit(T_test$Age)
T_test$Age[is.na(T_test$Age)] <- mean(test_age)

#записываем вместо значения NaS среднее значение Fare
test_fare<-na.omit(T_test$Fare)
T_test$Fare[is.na(T_test$Fare)] <- mean(test_fare)

train_fare<-na.omit(T_train$Fare)
T_train$Fare[is.na(T_train$Fare)] <- mean(train_fare)

quantity_of_data<-dim(T_train)[1]
model <- train.kknn(Survived ~ .,
                    data = T_train,
                    kmax=floor(sqrt(quantity_of_data)),
                    distance = 1)
model

model_titanik <- kknn(Survived ~ .,
                      T_train,
                      T_test,
                      k = 37,
                      kernel = "optimal",
                      distance = 1)

fit <- round(fitted(model_titanik))

prop.table(table(T_train$Survived))
prop.table(table(fit))
```