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

ЛАБОРАТОРНАЯ РАБОТА №1

«Наивный Байесовский классификатор»

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

1. Цель работы

В данной работе необходимо ознакомиться с наивным Байесовским классификатором, реализованном в пакете e1071 языка программирования R, исследовать результаты его работы на разных данных.

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. Сгенерируйте 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что первые 50 точек (class -1) имеют параметры: мат. ожидание X1 равно 10, мат. ожидание X2 равно 14, среднеквадратические отклонения для обеих переменных равны 4. Вторые 50 точек (class +1) имеют параметры: мат. ожидание X1 равно 20, мат. ожидание X2 равно 18, среднеквадратические отклонения для обеих переменных равны 3. Построить соответствующие диаграммы, иллюстрирующие данные. Построить байесовский классификатор и оценить качество классификации.

3. Разработать байесовский классификатор 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

Использовать функцию read.csv для чтения данных из csv-файлов.

Классы:

survival Выжил (0 = No; 1 = Yes)

Признаки:

pclass Класс каюты (1 = 1st; 2 = 2nd; 3 = 3rd)

name Имя

sex Пол

age Возраст

sibsp Число братьев-сестер/муж-жена на борту

parch Число родителей/детей на борту

ticket Номер билета

fare Стоимость билета

cabin Каюта

embarked Порт посадки (C = Cherbourg; Q = Queenstown; S = Southampton)

Специальные отметки:

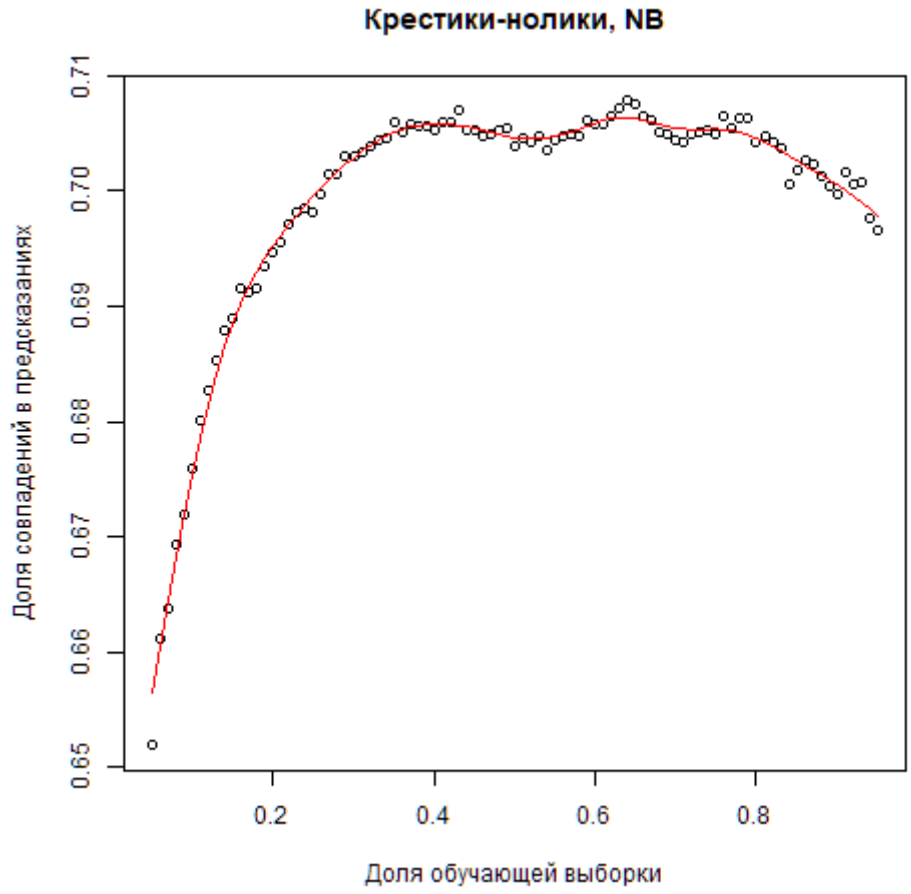
Pclass: 1st ~ Верхний; 2nd ~ Средний; 3rd ~ Нижний

Age – в годах; дробный, если возраст меньше одного года

3. Ход работы

Задание №1

Для примера Крестики-нолики было взято от 5% 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.



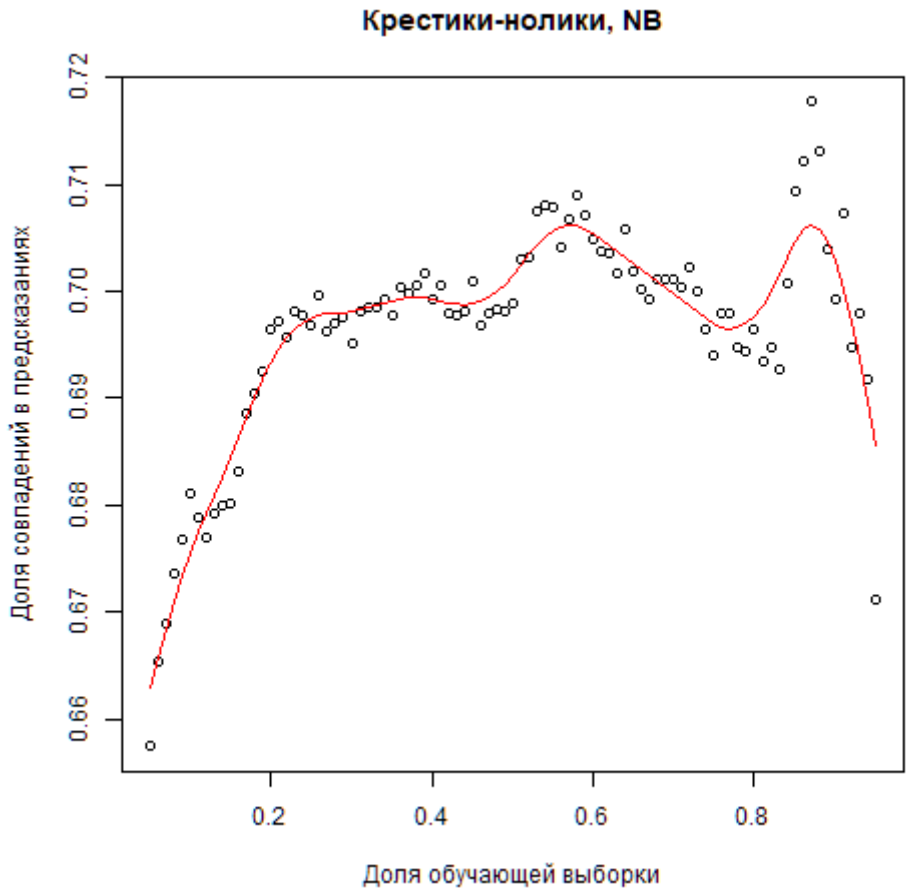


Рисунок 1. Зависимость точности классификации от объема обучающих/тестовых данных в примере «Крестики-нолики»

По рисунку 1 можно выделить зоны недообучения в 10-20% и переобучения – 80-95%. Процент правильных предсказаний составил 70%.

Можно сделать вывод, что для датасета «Крестики-нолики» увеличение числа примеров в обучающей выборке не приводит к повышению качества работы классификатора.

Для примера «спам» бралось от 10 до 4601 экземпляров обучающих данных с шагом 100. Данные также усреднялись после 10 испытаний для каждого количества экземпляров. По рисунку 2 можем заметить, что на данном датасете заметно недообучение: результаты заметно разбросаны, особенно на участке 10-1000. Средний показатель предположений классификатора составил 70%.

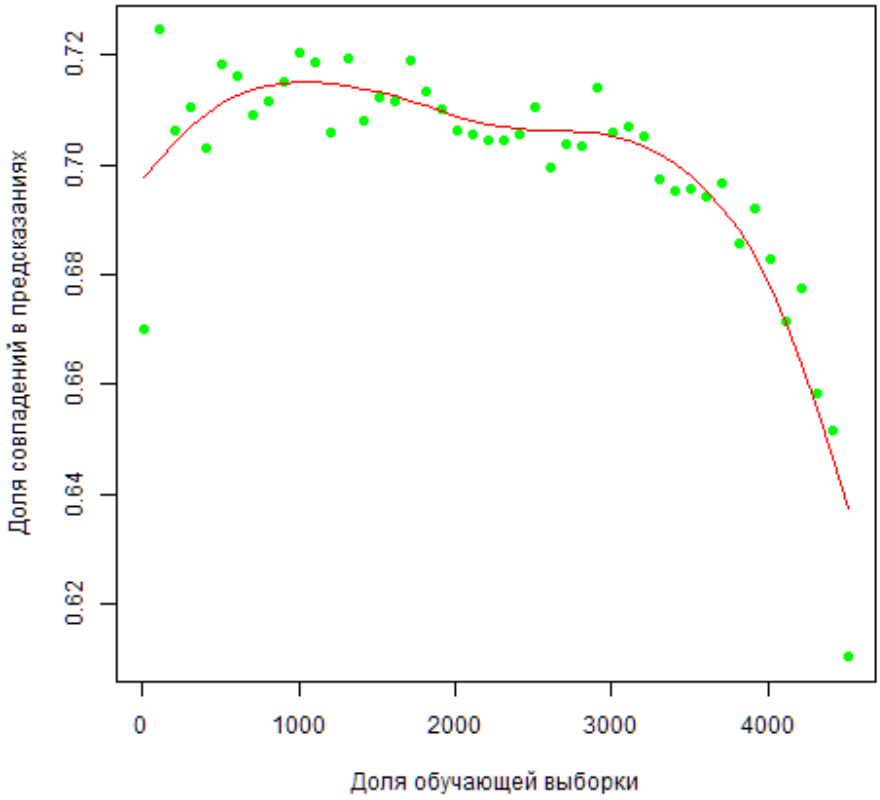


Рисунок 2. Зависимость точности классификации от объема обучающих/тестовых данных в примере Спам

После 1000 объектов происходит заметное переобучение, поэтому рассмотрим участок от 10 до 1000 объектов с шагом 10.

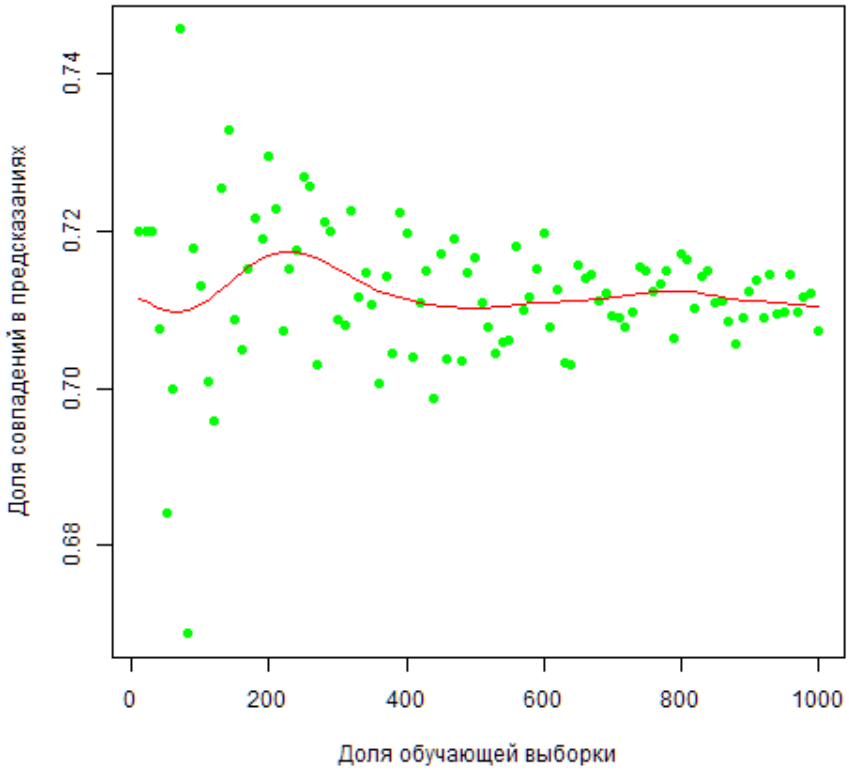


Рисунок 3. Зависимость точности классификации от объема обучающих/тестовых данных в примере Спам

Средний показатель предположений классификатора на участке 10-1000 составил 71%.

Задание №2

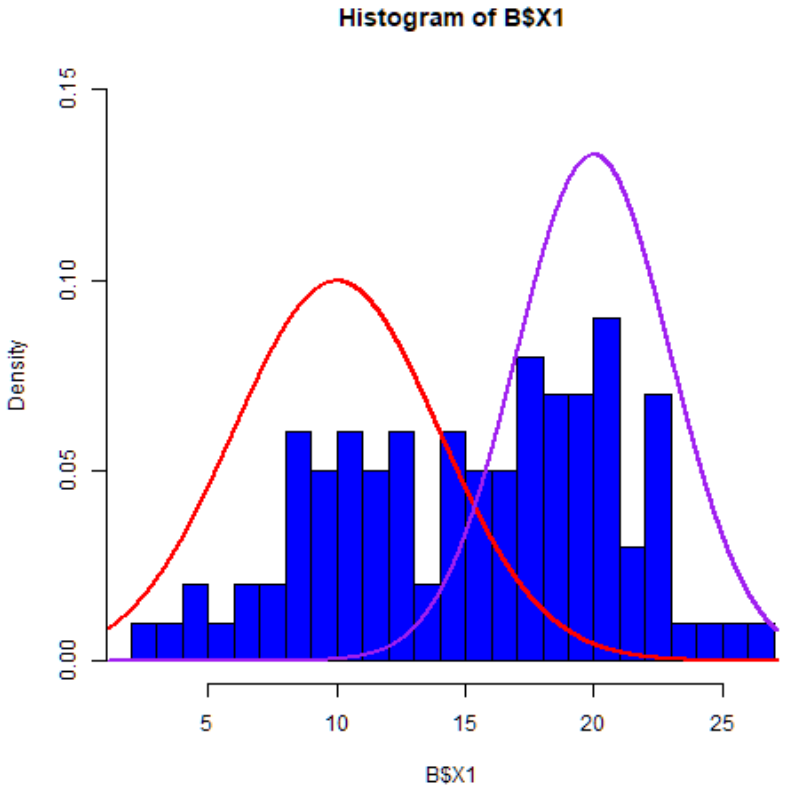


Рисунок 2. Распределение признака X1 и графики нормального распределения с заданными параметрами

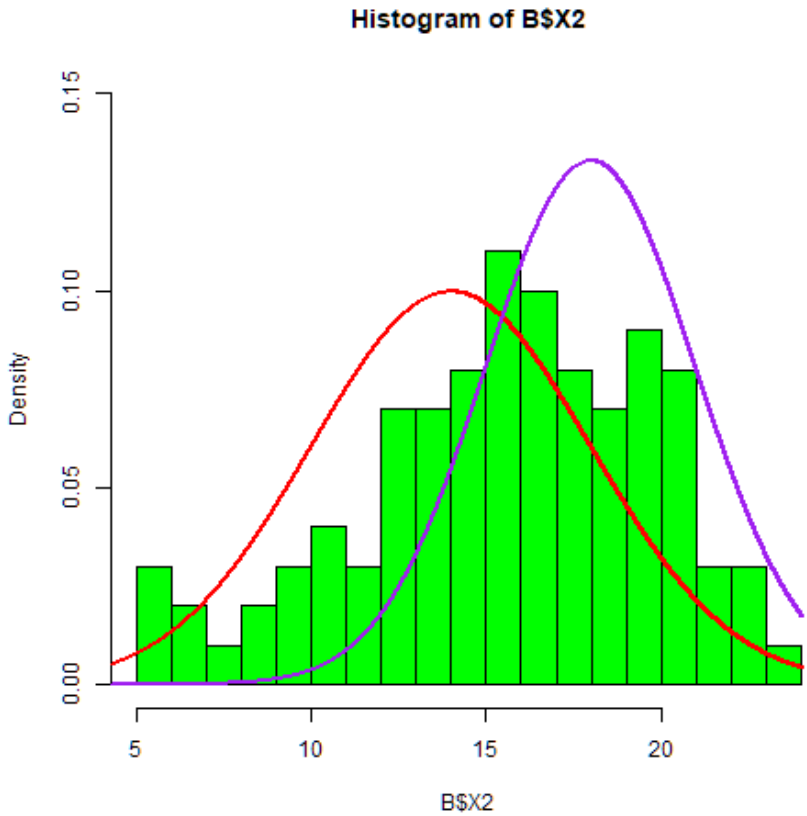


Рисунок 5. Распределение признака X2 и графики нормального распределения с заданными параметрами

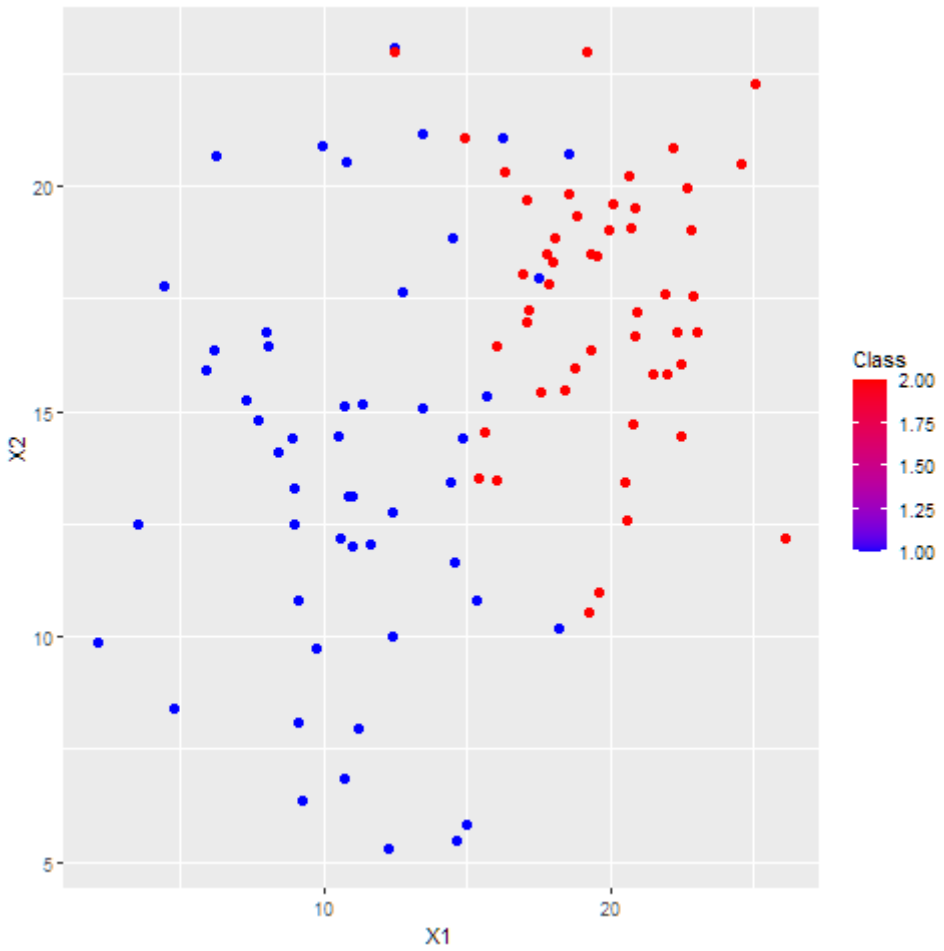


Рисунок 6. Результат генерации данных в координатах (X1, X2)

Для построенного байесовского классификатора с соотношением обучающей/тестовой выборок 70%/30% относительно изначальных данных точность классификации составила 90%.

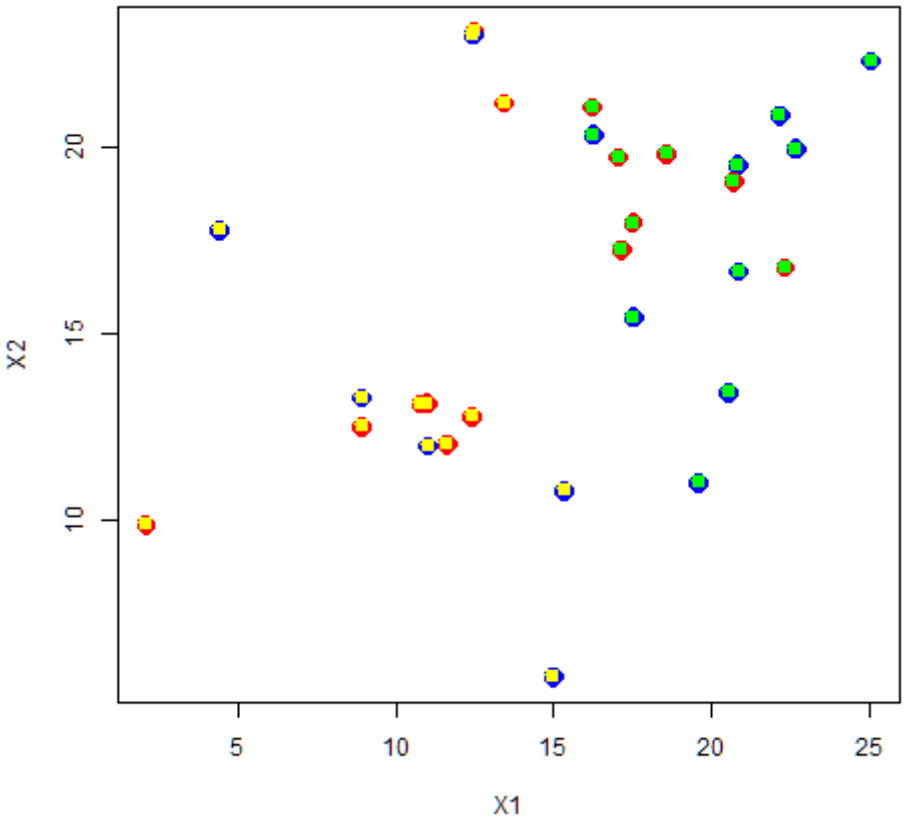


Рисунок 7. Результаты классификации для тестовых данных наглядно

Задание №3

Тестовые данные не обладают меткой Survived, которую мы пытаемся определить. Метки обучающей выборки: PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked.

Можно удалить признаки, не влияющие на классификацию данных – PassengerId, Name,Ticket, Cabin, Embarked. Такой выбор удаляемых признаков был сделан исключительно из собственных соображений: скорее всего номер кабины и порт посадки не повлияют на расположение человека в той или иной части корабля, их можно удалить, оставив метку Pclass, определяющую класс кабины. Id и имя пассажира – уникальные данные, которые совершенно точно никак не повлияют на точность классификации.

Далее было замечено, что в некоторых строках пропущены метки возраста и стоимость билета. Вместо пропущенных данных были подставлены средние значения по соответствующим признакам. Также в исходных данных были заменены признаки пола на числовые значения 0 – мужчина, 1 - женщина.

Используя функцию predict, получили метки для тестовых данных. Таблицу данных для каждого пассажира можно посмотреть при помощи функции data.frame(PassengerId = passengers,Survived = A_predicted).

В итоге для пассажиров тестовой выборки 169 – выжили и 249 – мертвы.

	Выжил	Погиб
Реальное соотношение данных в выборке	62%	38%
Результаты, данные построенным классификатором (с преобразованием датасета)	60%	40%
Результаты, данные построенным классификатором (без преобразования датасета)	72%	28%

Итак, получаем, что погрешность в определении верного результата составила 2%.

4. Вывод

В результате выполнения данной работы было изучено влияние объема данных на точность классификации наивного Байесовского классификатора. Предположение, что увеличение объема данных увеличивает разнообразие примеров, уменьшает ошибку обобщения, тем самым повышая точность классификации, не подтвердилось. Эксперименты показали, что при достижении некоторого количества тренировочных данных, дальнейшее увеличение примеров не повышает точность классификатора, а наоборот, может ее ухудшить.

Также классификатор хорошо себя показал при наивных предположениях: случай, когда нормально распределенные данные имеют разные параметры распределения.

Приложение 1

```
#Задание-1 ############################################################
install.packages("e1071")
library(e1071)

### Naive Bayesian (данные категориальные) ########################
# 1 ############################################################
# импортируем данные в R
# установить параметр stringsAsFactors = TRUE,
# так как все данные - категориальные
path <- "C:\\Users\\Софья\\Desktop\\1_курс_1_семестр\\Машинное обучение\\Лабораторная 1. NaiveBayesian\\"
A_raw <- read.table(paste(path, "Tic_tac_toe.txt", sep=""), sep = ",", stringsAsFactors = TRUE)
# число строк в базе
n <- dim(A_raw)[1]
# Создан фрейм, который можно просмотреть, используя str(A_raw).
# Имеется 9 столбцов признаков V1-V9 и V10 (класс) и
# все имеют один и тот же тип Factor.

# 2 ############################################################
# Создание обучающей и тестирующей выборки
# Скажем, имеем n примеров в исходной выборке,
# используем 80% для обучения и оставшиеся - для тестирования.

# Устанавливаем базу генерации случайных чисел и рандомизируем выборку
set.seed(12345)
A_rand <- A_raw[ order(runif(n)), ]
# разделим данные на обучающие и тестирующие
nt <- as.integer(n*0.8)
A_train <- A_rand[1:nt, ]
A_test <- A_rand[(nt+1):n, ]
# Можно убедиться, какой имеется процент каждого
# класса V2 в обучающей и тестирующей выборке
prop.table(table(A_train$V10))
prop.table(table(A_test$V10))

# 3 ############################################################
# Используем Наивный Байесовский классификатор из пакета e1071
#       A_classifier <- naiveBayes(A_train[,-10], A_train$V10)
# Другой вариант классификатора
A_classifier <- naiveBayes(V10 ~ ., data = A_train)

# 4 ############################################################
# Теперь оценим полученную модель:
A_predicted <- predict(A_classifier, A_test)
# Используем table для сравнения прогнозируемых значений с тем, что есть
table(A_predicted, A_test$V10)
```

```
# задаем параметры обучения
learning<-seq(0.05, 0.95, 0.01)
res_vec<-vector()
quantity_of_experiments<-10

for (learn_ratio in learning) {
 nt <- as.integer(n*learn_ratio)
 exp_res <- vector()
 #set.seed(12345)
 set.seed(1121)
 for (j in 1:quantity_of_experiments)
 {
  A_rand <- A_raw[order(runif(n)),]
  A_train <- A_rand[1:nt,]
  A_test <- A_rand[(nt + 1):n,]
  A_classifier <- naiveBayes(A_train[,  -10], A_train$V10)
  #print(A_classifier)
  #print(A_test)
  A_predicted <- predict(A_classifier, A_test)

  res <- table(A_predicted, A_test$V10)
  # доля правильно предсказанных результатов
  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
spline = smooth.spline(x, y, spar = 0.6)
png(paste(path, "Крестики-нолики.png"))
plot (x, y, xlab = "Доля обучающей выборки" , ylab = "Доля совпадений в предсказаниях", main = "Крестики-нолики, NB")
lines(spline, col = "red")
dev.off()

# spam-email ##############################################
install.packages("kernlab")
library(kernlab)
data(spam)

dim(spam)[1]
#learning <- seq(10, 4601, 100)
learning <- seq(10, 1000, 10)
res_vec <- vector()
for (learn_ratio in learning)
{
 exp_res <- vector()
```

```
  for (j in 1:quantity_of_experiments)
  {
    idx <- sample(1:nrow(spam), learn_ratio)
    spamtrain <- spam[-idx,]
    spamtest <- spam[idx,]

    model <- naiveBayes(type ~ ., data = spamtrain)

    res <- table(predict(model, spamtest), spamtest$type)

    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
 spline = smooth.spline(x, y, spar = 0.7)
 png(paste(path, "Спам.png"))
 plot (x, y, xlab = "Доля обучающей выборки" , ylab = "Доля совпадений в предсказаниях", col = "green", pch = 19, main = "Спам, NB")
 lines(spline, col = "red")
 dev.off()
```

Приложение 2

```
#Задание-2 ############################################################
install.packages("vctrs")
library(ggplot2)

n1 <- rnorm(50, 10, 4)
n2 <- rnorm(50, 20, 3)
n3 <- rnorm(50, 14, 4)
n4 <- rnorm(50, 18, 3)

X1 <- c(n1,n2)
X2 <- c(n3,n4)
C <- rep(c("-1" , "1") , each = 50)
B <- data.frame(X1, X2, C, stringsAsFactors = TRUE)

#визуализация

png(paste(path, "Hist1.png"))
hist(B$X1, col = "blue", breaks = as.integer(1 + 3.322*log(n)), probability = TRUE, ylim =
c(0, 0.15))
xx<-seq(min(n1, n2)-1, max(n1, n2)+1, 0.0001)
lines(xx, dnorm(xx, 10, 4), col = "red", lwd = 2)
lines(xx, dnorm(xx, 20, 3), col = "purple", lwd = 2)
dev.off()

png(paste(path, "Hist2.png"))
hist(B$X2, col = "green", breaks = as.integer(1 + 3.322*log(n)), probability = TRUE, ylim
= c(0, 0.15))
xx<-seq(min(n3, n4)-1, max(n3, n4)+1, 0.0001)
lines(xx, dnorm(xx, 14, 4), col = "red", lwd = 2)
lines(xx, dnorm(xx, 18, 3), col = "purple", lwd = 2)
dev.off()

png(paste(path, "Points.png"))
ggplot(B, aes(x=X1, y=X2, color = as.numeric(C))) +
  geom_point(size = 2) +
  scale_color_gradient(low="blue", high="red", name = "Class")
dev.off()

n = 100
learn_ratio = 0.7

B_rand <- B[order(runif(n)),]

nt <- as.integer(n*learn_ratio)
B_train <- B_rand[1:nt,]
B_test <- B_rand[(nt + 1):n,]

B_classifier <- naiveBayes(C ~ ., data = B_train)
```

```
B_predicted <- predict(B_classifier, B_test)

png(paste(path, "Points_2.png"))

plot(
  B_test[1:2],
  pch = 23,
  lwd = 4,
  bg = c("red", "blue"),
  col = c("red", "blue")
)
points(
  B_test[1:2],
  pch = 20,
  lwd = 4,
  col = c("yellow" , "green")[unclass(B_predicted)]
)
dev.off()
res <- table(B_predicted, B_test$C)
res

(res [1 , 1] + res [2 , 2]) / sum(res)
```

Приложение 3

```
#Задание-3 ############################################################
#загружаем данные
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)


A_classifier <- naiveBayes(T_train[,-1], as.factor(T_train$Survived))
A_predicted <- predict(A_classifier, T_test)
response <- data.frame(PassengerId = passengers,Survived = A_predicted)
#таблица предсказаний для тестовых данных
print(summary(A_predicted))

T_predicted <- predict(A_classifier, T_test)
prop.table(table(T_train$Survived))
prop.table(table(T_predicted))

############################################################

T_train <- read.csv(paste(path, "Titanic_train.csv", sep = ""), stringsAsFactors = T)
T_test <- read.csv(paste(path, "Titanic_test.csv", sep = ""), stringsAsFactors = T)

T_train$Name <- paste(T_train$Name, sep = ",", T_train$X)
T_train <- T_train[, -5]
```

```
T_test$Name <- paste(T_test$Name, sep = ",", T_test$X)
T_test <- T_test[, -4]

nt <- nrow(T_train)
n <- nt + nrow(T_test)

T_train$Survived <- as.factor(T_train$Survived)

T_classifier <- naiveBayes(Survived ~ . , data = T_train)

T_predicted <- predict(T_classifier, T_test)

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