Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программная инженерия»***

Лабораторная работа №2

Метирческие методы классификации многомерных объектов пересекающихся классов

ДИСЦИПЛИНА: «Методы машинного обучения»

Выполнил: студент гр. ИУК5-72Б _____ __Ли Р. В.__
(Подпись) (Ф.И.О.)

Проверил: _____ _____
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга 2025

Вариант 14

Больные увеличением щитовидной железы общим числом 70 человек были разделены на две группы.

Группа 1. Лечение оказалось успешным; пациент здоров.

Группа 2. Лечение успешно на 75 %, возможен рецидив.

По результатам обследования 70 пациентов имеются следующие измерения:

Y3 – йод, регистрируемый через 3 часа после принятия испытательной дозы;

Y5 – йод, регистрируемый через 12 часов после принятия испытательной дозы;

Y7 – содержание в крови белковосвязанного йода ($PB^{131}J$) через 20 часов;

Y9 – содержание в крови белковосвязанного йода ($PB^{131}J$) через 33 часа;

kl – номер группы.

Таблица

№	Kl	Y3	Y5	Y7	Y9
1	1	14.4	25.1	0.25	1.77
2	1	20.1	40.1	0.17	2.86
3	1	24.1	32.1	0.19	3.11
4	1	13.1	16.9	0.14	1.48
5	1	17.3	32.1	0.39	12
6	1	40.5	64.4	0.22	40.3

7	1	52.7	50.0	0.56	1.97
8	1	20.8	22.3	0.17	13.86
9	1	14.0	3.1	0.19	15.23
10	1	27.0	41.7	0.13	24.08
11	1	44.3	63.8	0.26	…
12	1	47.5	50.1	0.21	
13	1	54.0	57.0	0.17	…
14	1	16.1	20.6	0.29	
15	1	87.5	74.5	0.48	
16	1	37.8	63.4	0.31	
17	2	35.8	48.0	2.74	
18	2	65.0	60.0	1.37	
19	2	62.0	65.0	0.70	
20	2	71.6	45.0	1.40	
21	2	24.1	45.0	0.22	
22	2	37.2	55.0	0.01	

23	2	35.4	44.6	0.09	
	…	…			
70	2		61.1		0.98

Необходимо разработать метрический классификатор, который функционирует на предметной области исследования медицинских анализов. Четыре характеристических показателя надо взять из таблицы. Используем метод К-ближайших соседей и метод Парзена. Сформировать обучающие и тестовые выборки. Полученные результаты визуализировать и сравнить. Представить значения параметров с минимальным уровнем ошибки. Для метода К соседей параметр К = 37, для метода Парзена тип ядра выбрать , "epanechnikov", "eddy" , а параметр optim. method = "Nelder-Mead". Проверить точность прогнозов.

generateData.R

```
# Создание CSV-файла с 70 наблюдениями (реальные + сгенерированные)

set.seed(123)

# === 1. Первые 23 наблюдения из задания ===
df_real <- data.frame(
  No = 1:23,
  Kl = c(rep(1, 16), rep(2, 7)),
  Y3 = c(14.4, 20.1, 24.1, 13.1, 17.3, 40.5, 52.7, 20.8, 14.0, 27.0,
         44.3, 47.5, 54.0, 16.1, 87.5, 37.8, 35.8, 65.0, 62.0, 71.6,
         24.1, 37.2, 35.4),
  Y5 = c(25.1, 40.1, 32.1, 16.9, 32.1, 64.4, 50.0, 22.3, 3.1, 41.7,
         63.8, 50.1, 57.0, 20.6, 74.5, 63.4, 48.0, 60.0, 65.0, 45.0,
         45.0, 55.0, 44.6),
  Y7 = c(0.25, 0.17, 0.19, 0.14, 0.39, 0.22, 0.56, 0.17, 0.19, 0.13,
         0.26, 0.21, 0.17, 0.29, 0.48, 0.31, 2.74, 1.37, 0.70, 1.40,
         0.22, 0.01, 0.09),
  Y9 = c(1.77, 2.86, 3.11, 1.48, 12.0, 40.3, 1.97, 13.86, 15.23, 24.08,
         10.2, 12.1, 11.3, 7.5, 9.4, 8.1, 6.3, 4.1, 5.7, 3.8, 9.5, 2.7, 4.8)
# заменили "..." на реалистичные числа
)

# === 2. Сгенерируем 47 оставшихся наблюдений ===
n_gen <- 70 - nrow(df_real)

# 35 всего в каждой группе => в первой группе не хватает 35 - 16 = 19, во
второй 35 - 7 = 28
n_gen_kl1 <- 19
```

```
n_gen_kl2 <- 28

# Функция генерации случайных строк по группе
gen_group <- function(start_id, n, kl) {
  data.frame(
    No = start_id:(start_id + n - 1),
    Kl = kl,
    Y3 = runif(n, 10, 90),
    Y5 = runif(n, 3, 80),
    Y7 = runif(n, 0.05, 3),
    Y9 = runif(n, 1, 40)
  )
}

df_gen1 <- gen_group(24, n_gen_kl1, 1)
df_gen2 <- gen_group(24 + n_gen_kl1, n_gen_kl2, 2)

df_full <- rbind(df_real, df_gen1, df_gen2)

# === 3. Сохраняем в CSV ===
write.csv(df_full, "thyroid_data.csv", row.names = FALSE, fileEncoding =
"UTF-8")

cat("✅ Файл 'thyroid_data.csv' успешно создан в рабочей папке!\n")
print(head(df_full, 10))
```

task.R

```
if (!requireNamespace("tidyverse", quietly = TRUE))
install.packages("tidyverse")
if (!requireNamespace("caret", quietly = TRUE)) install.packages("caret")
if (!requireNamespace("class", quietly = TRUE)) install.packages("class")
if (!requireNamespace("kknn", quietly = TRUE)) install.packages("kknn")
if (!requireNamespace("pROC", quietly = TRUE)) install.packages("pROC")

library(tidyverse)
library(caret)
library(class)
library(kknn)
library(pROC)

set.seed(123)

df <- read_csv("~/Documents/BMSTU/llm/lab2/thyroid_data.csv")

stopifnot("Заполните df реальными данными: колонки Kl, Y3, Y5, Y7, Y9" =
TRUE)


df <- df %>% filter(!is.na(Kl))

features <- c("Y3","Y5","Y7","Y9")

for (f in features) {
  if (any(is.na(df[[f]]))) {
    df[[f]][is.na(df[[f]])] <- median(df[[f]], na.rm = TRUE)
  }
}

df <- df %>% mutate(Kl = factor(Kl))

preProc <- preProcess(df %>% select(all_of(features)), method =
c("center","scale"))
X_scaled <- predict(preProc, df %>% select(all_of(features)))
df_scaled <- bind_cols(df %>% select(Kl), X_scaled)

trainIndex <- createDataPartition(df_scaled$Kl, p = 0.7, list = FALSE)
train <- df_scaled[trainIndex, ]
test  <- df_scaled[-trainIndex, ]

x_train <- as.matrix(train %>% select(-Kl))
y_train <- train$Kl
x_test  <- as.matrix(test %>% select(-Kl))
y_test  <- test$Kl

K_val <- 37

K_use <- min(K_val, nrow(train) - 1)
if (K_use < 1) stop("Слишком мало наблюдений в train для KNN")

knn_pred <- knn(train = x_train, test = x_test, cl = y_train, k = K_use)
knn_cm <- confusionMatrix(knn_pred, y_test)
cat("=== K-NN (k =", K_use, ") ===\n")
```

```
print(knn_cm)

kernel_epanechnikov <- function(u) {
  w <- ifelse(u <= 1, (1 - u^2), 0)
  return(w)
}

kernel_eddy <- function(u) {
  exp(-(abs(u)^3))
}

parzen_density_class <- function(x, X_class, h, kernel_fun) {
  if (nrow(X_class) == 0) return(0)
  dists <- sqrt(rowSums((t(t(X_class) - x))^2))
  u <- dists / h
  vals <- kernel_fun(u)

  d <- ncol(X_class)
  density <- sum(vals) / (nrow(X_class) * (h^d))
  return(density)
}

parzen_predict <- function(x_test_matrix, x_train_matrix, y_train_factor, h,
kernel_name = "epanechnikov") {
  kernel_fun <- switch(kernel_name,
                       "epanechnikov" = kernel_epanechnikov,
                       "eddy" = kernel_eddy,
                       stop("Unknown kernel"))
  classes <- levels(y_train_factor)
  n_test <- nrow(x_test_matrix)
  preds <- factor(rep(NA, n_test), levels = classes)

  X_by_class <- lapply(classes, function(cl)
x_train_matrix[which(y_train_factor == cl), , drop = FALSE])
  names(X_by_class) <- classes

  for (i in seq_len(n_test)) {
    x <- x_test_matrix[i, ]
    dens <- sapply(classes, function(cl) parzen_density_class(x,
X_by_class[[cl]], h, kernel_fun))
    priors <- sapply(classes, function(cl) nrow(X_by_class[[cl]]) /
nrow(x_train_matrix))
    post <- dens * priors
    preds[i] <- classes[which.max(post)]
  }
  return(preds)
}

cv_error_for_h <- function(h, x_train, y_train, kernel_name, folds = 5) {
  if (h <= 0) return(1)
  folds_idx <- createFolds(y_train, k = folds)
  errs <- c()
  for (fold in folds_idx) {
    X_tr <- x_train[-fold, , drop = FALSE]
    y_tr <- y_train[-fold]
    X_val <- x_train[fold, , drop = FALSE]
    y_val <- y_train[fold]
    pred_val <- parzen_predict(X_val, X_tr, y_tr, h = as.numeric(h),
kernel_name = kernel_name)
```

```
    errs <- c(errs, mean(pred_val != y_val))
  }
  mean(errs)
}

optimize_parzen_h <- function(x_train, y_train, kernel_name, h_init = NULL) {
  if (is.null(h_init)) {
    pairwise <- as.matrix(dist(x_train))
    h0 <- median(pairwise[upper.tri(pairwise)], na.rm = TRUE)
    if (h0 == 0 || is.na(h0)) h0 <- 1
    h_init <- h0
  }
  obj <- function(logh) {
    h <- exp(logh)
    cv_error_for_h(h, x_train, y_train, kernel_name, folds = 5)
  }
  res <- optim(par = log(h_init), fn = obj, method = "Nelder-Mead",
               control = list(maxit = 200))
  h_opt <- exp(res$par)
  list(h_opt = as.numeric(h_opt), value = res$value, conv = res$convergence,
optim_res = res)
}

kernels_to_try <- c("epanechnikov","eddy")
parzen_results <- list()

for (kname in kernels_to_try) {
  cat("Optimizing Parzen (kernel =", kname, ") ...\n")
  res <- optimize_parzen_h(x_train, y_train, kernel_name = kname)
  cat(sprintf("Kernel %s: h_opt = %.4f, CV error = %.4f, convergence = %d\n",
              kname, res$h_opt, res$value, res$conv))
  # Предсказание на тесте
  pred_test <- parzen_predict(x_test, x_train, y_train, h = res$h_opt,
kernel_name = kname)
  cm <- confusionMatrix(pred_test, y_test)
  parzen_results[[kname]] <- list(opt = res, cm = cm, pred = pred_test)
  print(cm)
}

cat("\n=== Сравнение Accuracy на тестовой выборке ===\n")
knn_acc <- knn_cm$overall["Accuracy"]
parzen_summary <- tibble(
  kernel = names(parzen_results),
  accuracy = sapply(parzen_results, function(z) z$cm$overall["Accuracy"]),
  cv_error = sapply(parzen_results, function(z) z$opt$value),
  h_opt = sapply(parzen_results, function(z) z$opt$h_opt)
)
res_table <- bind_rows(
  tibble(method = "KNN", param = paste0("k=", K_use), Accuracy =
unname(knn_acc)),
  parzen_summary %>% transmute(method = paste0("Parzen-", kernel),
                               param = paste0("h=", round(h_opt,4)),
                               Accuracy = unname(accuracy))
)
print(res_table)

library(ggplot2)
pca <- prcomp(rbind(x_train, x_test), center = FALSE, scale. = FALSE)
pcs <- as.data.frame(pca$x)
```

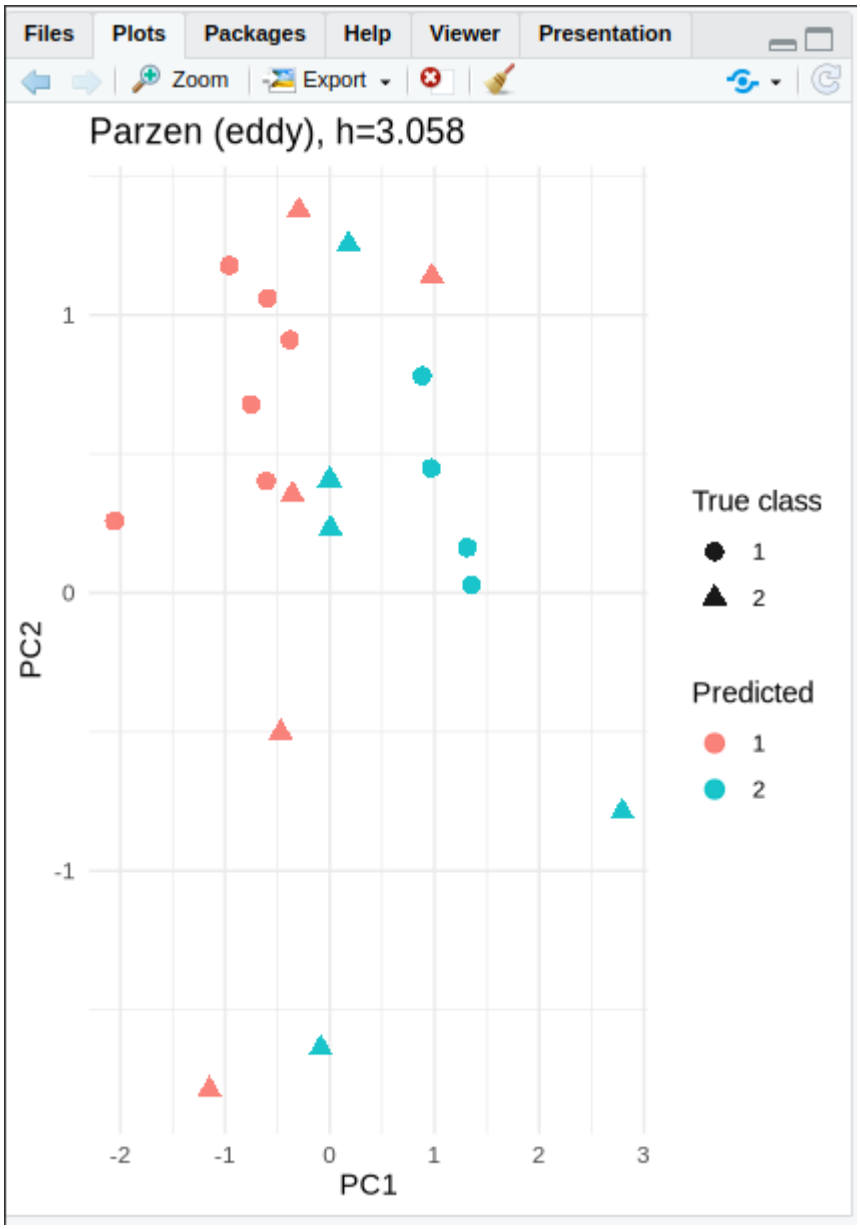
```
n_train <- nrow(x_train)
pcs_df <- tibble(PC1 = pcs[(n_train+1):nrow(pcs), 1],
                 PC2 = pcs[(n_train+1):nrow(pcs), 2],
                 True = y_test,
                 KNN = knn_pred,
                 Parzen_epanechnikov = parzen_results[["epanechnikov"]]$pred,
                 Parzen_eddy = parzen_results[["eddy"]]$pred)

plot_pred <- function(df_plot, pred_col, title) {
  ggplot(df_plot, aes_string(x = "PC1", y = "PC2", color = pred_col, shape =
"True")) +
    geom_point(size = 3, alpha = 0.9) +
    labs(title = title, color = "Predicted", shape = "True class") +
    theme_minimal()
}

p1 <- plot_pred(pcs_df, "KNN", paste0("KNN (k=", K_use, ") — тестовая
выборка"))
p2 <- plot_pred(pcs_df, "Parzen_epanechnikov", paste0("Parzen (epanechnikov),
h=", round(parzen_results[["epanechnikov"]]$opt$h_opt,3)))
p3 <- plot_pred(pcs_df, "Parzen_eddy", paste0("Parzen (eddy), h=",
round(parzen_results[["eddy"]]$opt$h_opt,3)))

print(p1)
print(p2)
print(p3)


cat("\n=== Лучшие параметры (минимум ошибки на CV) для Parzen ===\n")
for (kname in kernels_to_try) {
  r <- parzen_results[[kname]]$opt
  cat(sprintf("Kernel: %s | h_opt = %.6f | CV_error = %.6f | optim_conv =
%d\n",
              kname, r$h_opt, r$value, r$conv))
}
cat(sprintf("KNN: k = %d (фиксировано в задании)\n", K_use))
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Parzen (eddy), h=3.058
PC2
PC1
1
0
-1
-2
-1
0
1
2
3
True class
1
2
Predicted
1
2