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»***

Домашняя работа

Разработка нейронных сетей с помощью языка R

ДИСЦИПЛИНА: «Методы машинного обучения»

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

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

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга 2025

Вариант 14

Разработать нейросетевой классификатор для распознавания видов растений по отношению к механическому составу песчаных почв (пелитофиты, псаммофиты). Разработать набор признаков характеризующих каждый из двух заданных классов растений. Использовать функции NeuralNet и MLP (параметр learnFunc (алгоритм обучения) выбрать Rprop). Оптимизировать параметры нейронных сетей с помощью пакета caret и сравнит полученные результаты. Полный список с исходными данными взять из приложения к домашнему заданию.

task.R

```
# Вариант 14: классификатор пелитофитов vs псаммофитов
# Зависимости: установите при необходимости install.packages(...)
#install.packages("neuralnet", "RSNNS", "caret", "ROC", "dplyr")
library(neuralnet)   # для примера neuralnet
library(RSNNS)       # mlp, поддерживает learnFunc (включая "Rprop")
library(caret)       # тюнинг и оценка
library(pROC)        # ROC/auc
library(dplyr)

set.seed(2025)

# ---------------------------
# 1) Загрузка данных
# ---------------------------
# Вариант A: если у вас есть CSV с признаками и колонкой "class"
# CSV должен содержать: columns of numeric predictors and a column 'class'
with values 'pelit' or 'psamm'
# data <- read.csv("your_data.csv", stringsAsFactors = FALSE)

# Вариант B: демонстрационный синтетический датасет (для тестирования
скрипта)
# (Удалите или замените на загрузку реального файла)
n <- 300
# Признаки: grain_size, clay_content, silt_content, porosity, bulk_density,
organic_matter
pelit <- data.frame(
  grain_size = rnorm(n/2, mean = 0.12, sd = 0.03),     # более мелкие частицы
  clay_content = rnorm(n/2, mean = 0.35, sd = 0.07),
  silt_content = rnorm(n/2, mean = 0.30, sd = 0.05),
  porosity = rnorm(n/2, mean = 0.50, sd = 0.05),
  bulk_density = rnorm(n/2, mean = 1.1, sd = 0.07),
  organic_matter = rnorm(n/2, mean = 0.04, sd = 0.02),
  class = "pelit"
)

psamm <- data.frame(
  grain_size = rnorm(n/2, mean = 0.60, sd = 0.12),     # крупные песчинки
  clay_content = rnorm(n/2, mean = 0.03, sd = 0.02),
  silt_content = rnorm(n/2, mean = 0.10, sd = 0.05),
  porosity = rnorm(n/2, mean = 0.42, sd = 0.06),
```

```
  bulk_density = rnorm(n/2, mean = 1.45, sd = 0.08),
  organic_matter = rnorm(n/2, mean = 0.01, sd = 0.005),
  class = "psamm"
)

data <- bind_rows(pelit, psamm)
data$class <- factor(data$class, levels = c("pelit", "psamm"))

# ---------------------------
# 2) Простая разведка
# ---------------------------
cat("Rows:", nrow(data), " Classes:", levels(data$class), "\n")
summary(data)

# ---------------------------
# 3) Разделение на train/test
# ---------------------------
trainIndex <- createDataPartition(data$class, p = 0.75, list = FALSE)
train <- data[trainIndex, ]
test  <- data[-trainIndex, ]

# ---------------------------
# 4) Нормализация (min-max)
# ---------------------------
preproc <- preProcess(train %>% select(-class), method = c("range"))
train_norm <- predict(preproc, train %>% select(-class))
test_norm  <- predict(preproc, test %>% select(-class))

train_norm$class <- train$class
test_norm$class  <- test$class

# ---------------------------
# 5) Кодирование целевой переменной для нейросетей
# ---------------------------
# Для neuralnet: бинарные колонки
train_nn <- train_norm %>% mutate(pelit = ifelse(class == "pelit", 1, 0))
test_nn  <- test_norm  %>% mutate(pelit = ifelse(class == "pelit", 1, 0))

# ---------------------------
# 6) Модель 1: neuralnet (для сравнения)
# ---------------------------
# формула: pelit ~ predictors
predictors <- names(train_norm)[names(train_norm) != "class"]
fmla_nn <- as.formula(paste("pelit ~", paste(predictors, collapse = " + ")))

cat("Training neuralnet (benchmark)...\n")
nn_net <- neuralnet(fmla_nn, data = train_nn, hidden = c(5), linear.output =
FALSE,
                    stepmax = 1e6)
# Предсказания (вероятности)
nn_pred_raw <- neuralnet::compute(nn_net, test_nn[, predictors])
$net.result[,1]
nn_pred_class <- ifelse(nn_pred_raw >= 0.5, "pelit", "psamm")

conf_nn <- confusionMatrix(factor(nn_pred_class, levels =
levels(data$class)),
                           test_nn$class)
cat("neuralnet results:\n")
print(conf_nn)
```

```
# ---------------------------
# 7) Модель 2: RSNNS::mlp с learnFunc = "Rprop"
# ---------------------------
# RSNNS::mlp принимает матрицы: inputs (n x p) и targets (n x k)
# Для бинарной классификации можно использовать один выход (0/1) или два
выхода (one-hot).
# Используем один выход (pelit = 1, psamm = 0), актив. функция sigmoidal.
train_inputs <- as.matrix(train_nn[, predictors])
train_targets <- as.matrix(train_nn$pelit)  # 1/0
test_inputs  <- as.matrix(test_nn[, predictors])
test_targets <- as.matrix(test_nn$pelit)

cat("Training RSNNS::mlp with learnFunc = 'Rprop' ...\n")
# Параметры Rprop можно настроить через learnFuncParams, но RSNNS
документация и
# интерфейс позволяют явно задать learnFunc = "Rprop"
rsnns_model <- mlp(train_inputs, train_targets,
                   size = c(8),               # один скрытый слой с 8
нейронами
                   maxit = 200,
                   learnFunc = "Rprop",       # требование варианта
                   initFunc = "Randomize_Weights",
                   linOut = FALSE)

# Предсказания и порог 0.5
rsnns_pred_raw <- predict(rsnns_model, test_inputs)
rsnns_pred_class <- ifelse(rsnns_pred_raw >= 0.5, "pelit", "psamm")

conf_rsnns <- confusionMatrix(factor(rsnns_pred_class, levels =
levels(data$class)),
                              test_nn$class)
cat("RSNNS::mlp (Rprop) results:\n")
print(conf_rsnns)

# ---------------------------
# 8) Тюнинг через caret (метод = "mlp" использует RSNNS::mlp; тюнинг
параметра size)
# ---------------------------
# Важно: caret::train не даёт прямой удобной смены learnFunc в wrapper, но
для базового сравнения
# можно тюнить size (число скрытых нейронов). Если нужно обязательно менять
learnFunc внутри caret,
# можно использовать train(..., method = "mlp", ...) и затем вручную обучать
RSNNS с нужными learnFunc.
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3,
classProbs = TRUE,
                     summaryFunction = twoClassSummary, savePredictions =
"final")

# caret ожидает факторные метки с уровнем положительного класса в первой
позиции?
# twoClassSummary по умолчанию использует "event" = first level. Укажем
levels явно.
train_for_caret <- train_norm
train_for_caret$class <- factor(train_for_caret$class, levels = c("pelit",
"psamm"))

tuneGrid <- expand.grid(size = c(3,5,8,12))  # разные размеры
```

```
set.seed(123)
caret_mlp <- caret::train(
  class ~ .,
  data = train_for_caret,
  method = "mlp",
  metric = "ROC",
  trControl = ctrl,
  tuneGrid = tuneGrid
)


cat("caret mlp tuning results:\n")
print(caret_mlp)
best_size <- caret_mlp$bestTune$size
cat("Best size:", best_size, "\n")

# Оценка caret модели на тесте
caret_pred_prob <- predict(caret_mlp, newdata = test_norm, type = "prob")[,
"pelit"]
caret_pred_class <- ifelse(caret_pred_prob >= 0.5, "pelit", "psamm")
conf_caret <- confusionMatrix(factor(caret_pred_class, levels =
levels(data$class)), test_norm$class)
cat("caret::mlp results on test set:\n")
print(conf_caret)

# ----------------------------
# 9) ROC / AUC (для моделей, возвращающих вероятности)
# ----------------------------
roc_nn   <- roc(response = ifelse(test_nn$class == "pelit", 1, 0), predictor
= nn_pred_raw)
roc_rsnns<- roc(response = ifelse(test_nn$class == "pelit", 1, 0), predictor
= as.numeric(rsnns_pred_raw))
roc_caret<- roc(response = ifelse(test_norm$class == "pelit", 1, 0),
predictor = caret_pred_prob)

cat(sprintf("AUC (neuralnet) = %.3f\n", auc(roc_nn)))
cat(sprintf("AUC (RSNNS Rprop) = %.3f\n", auc(roc_rsnns)))
cat(sprintf("AUC (caret mlp) = %.3f\n", auc(roc_caret)))
# ----------------------------------------------------
# 10) ROC-кривые для трех моделей на одном графике
# ----------------------------------------------------
plot(roc_nn, col = "blue", lwd = 2,
     main = "ROC-кривые трёх моделей нейросетей")
plot(roc_rsnns, col = "darkgreen", lwd = 2, add = TRUE)
plot(roc_caret, col = "red", lwd = 2, add = TRUE)

legend("bottomright",
       legend = c(
         paste("neuralnet  AUC =", round(auc(roc_nn),3)),
         paste("RSNNS Rprop  AUC =", round(auc(roc_rsnns),3)),
         paste("caret mlp  AUC =", round(auc(roc_caret),3))
       ),
       col = c("blue", "darkgreen", "red"),
       lwd = 2,
       bty = "n")

# ----------------------------------------------------
# 11) Scatter-график правильных и ошибочных классификаций
#     (используем RSNNS Rprop как основную модель варианта)
```

```
# ----------------------------------------------------
rsnns_pred_class <- ifelse(rsnns_pred_raw >= 0.5, "pelit", "psamm")

plot(
  test_norm$grain_size,
  test_norm$clay_content,
  col = ifelse(test_norm$class == "pelit", "darkgreen", "red"),
  pch = ifelse(rsnns_pred_class == test_norm$class, 16, 4),
  main = "Классификация по двум признакам (RSNNS Rprop)",
  xlab = "grain_size (норм.)",
  ylab = "clay_content (норм.)"
)

legend("topright",
       legend = c("Пелит", "Псамм", "Ошибка"),
       col = c("darkgreen", "red", "black"),
       pch = c(16, 16, 4))
```