МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Автоматизация схемотехнического проектирования»

Тема: **«ДЕРЕВЬЯ И ЛЕСА РЕШЕНИЙ»**

Студент гр. 2301 _____ Комиссаров П.Е.

Преподаватель: _____ Боброва Ю.О.

Санкт-Петербург
2026

Цель работы:

Реализация классификатора на основе дерева принятия решений и исследование его свойств

Основные теоретические положения

Дерево решений (распознающее дерево) — классификатор, в котором для объекта выполняется конечная последовательность сравнений признаков с пороговыми значениями.

Распознавание задаётся вложенными операторами вида

«if x[j] ?? d[k] then … else …»

и завершается листьями с ответом класса (return res[h]).

Структурно дерево решений состоит из объектов двух типов — узлов (node) и листьев (leaf).

В узлах расположены решающие правила и подмножества наблюдений, которые им удовлетворяют.

В листьях содержатся классифицированные деревом наблюдения

По сути дерево «разрезает» признаковое пространство гиперплоскостями по порогам признаков и является **линейным классификатором**. Применимо и для регрессии.

Обучение дерева — определение структуры, выбора признаков и порогов в узлах и ответов в листьях по размеченной выборке (обучение с учителем).

Жадный алгоритм: на каждом шаге выбирается локально оптимальное разбиение (по критерию, например энтропия или Gini); откат и перевыбор атрибута не выполняется, поэтому итоговое дерево не обязательно глобально оптимально.

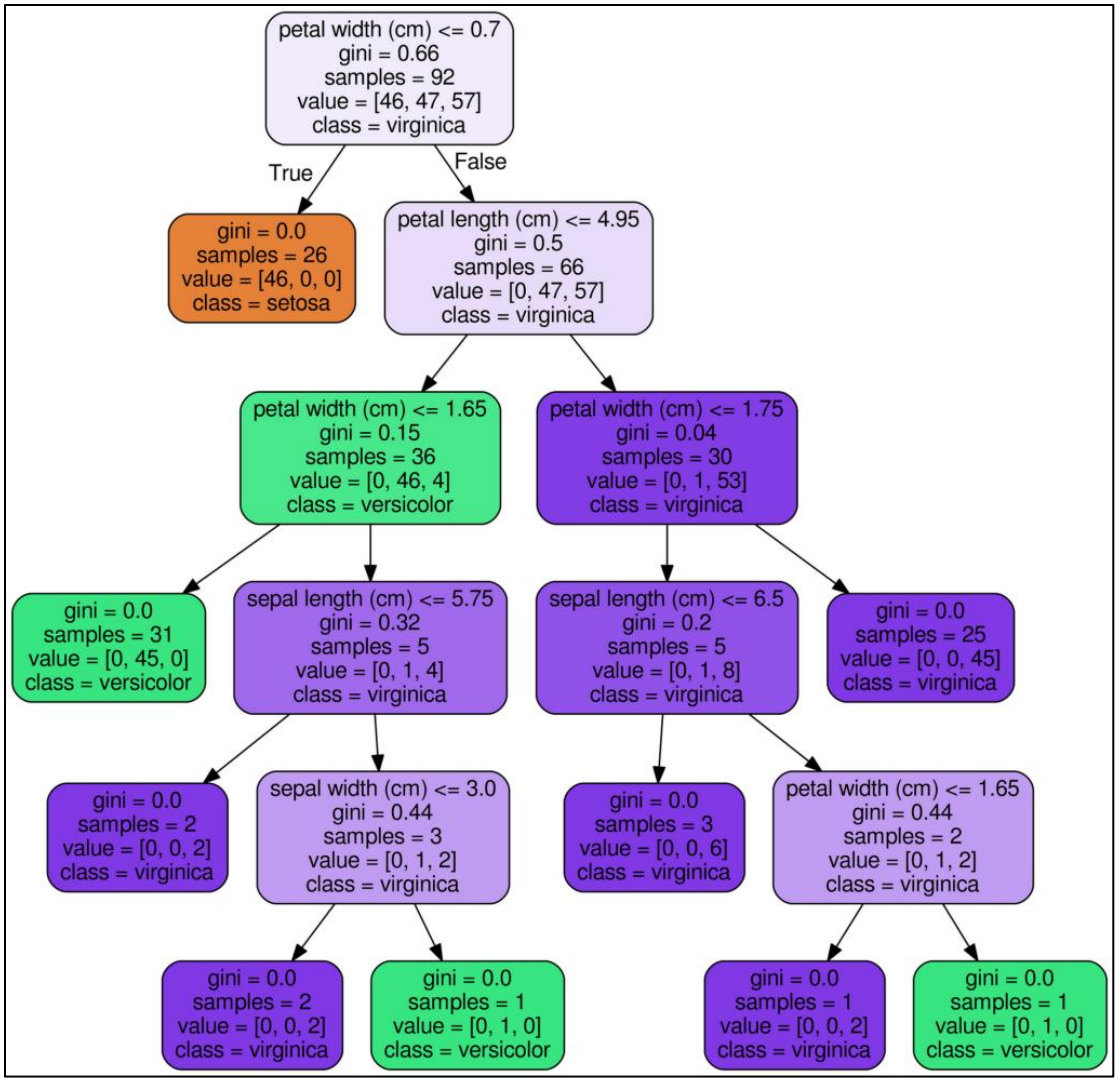

Рисунок 1 - пример дерева решений

Ансамбль — совокупность алгоритмов, объединённых в одно целое; итоговое решение получают голосованием (классификация) или усреднением (регрессия), что часто снижает ошибку по сравнению с одной моделью.

Бэггинг (Bootstrap aggregation): базовые модели обучаются на разных бутстрэп-подвыборках исходной выборки; это снижает дисперсию и уменьшает переобучение, ошибки моделей частично компенсируются при голосовании.

Случайный лес (random forest) — бэггинг над решающими деревьями; в каждом узле при разбиении признаки берутся из случайного подмножества. Для классификации итог — по

большинству голосов деревьев, для регрессии — по среднему. Склонность к переобучению и «рваным» границам остаётся; качество зависит от числа деревьев и других гиперпараметров.

ROC-кривая — зависимость доли истинно положительных (TPR) от доли ложноположительных (FPR) при изменении порога классификации. **AUC (площадь под ROC-кривой)** — интегральная мера качества; чем ближе к 1, тем лучше разделение классов.

Чувствительность = TP / (TP + FN); **специфичность** = TN / (TN + FP). Класс 0 — отсутствие признака, класс 1 — наличие. **Точность** = (TP + TN) / (всего объектов).

Ход выполнения работы

1. Повторены пункты 1–3 из лабораторной №2 с заменой логистической регрессии на дерево решений: импортированы numpy, matplotlib, pathlib, DecisionTreeClassifier из sklearn.tree; подключён lab1.DataGenerator.

Созданы два массива данных с **средней степенью пересечения** (нормальное распределение, выборка Б), выборка разбита на обучающую и тестовую 70 % / 30 %.

2. Модель дерева обучена на Xtrain, Ytrain методом fit(). Задан random_state=0 для воспроизводимости:

tree = DecisionTreeClassifier(random_state=0).fit(Xtrain, Ytrain).

3. На обучающей и тестовой выборках оценены точность (score), чувствительность и специфичность (по формулам, функция sensitivity_specificity).

На обучающих данных точность дерева равна или близка к 1; на тестовых — заметно ниже.

Это объясняется **переобучением**: дерево подстраивается под

обучающую выборку (вплоть до запоминания объектов), тогда как на новых данных обобщение хуже.

4. Аналогично обучен RandomForestClassifier на тех же данных. Результаты для дерева и леса выведены в таблицы и сравнены. Лес обычно даёт более стабильную точность на тесте и меньший разрыв между train и test за счёт усреднения многих деревьев.

5. Построены ROC-кривые для дерева и леса по вероятностям на тестовом наборе (predict_proba[:, 1]).

Площадь под кривой рассчитана через sklearn.metrics.roc_auc_score.

6. Построены гистограммы распределения вероятности принадлежности классу 1 для **случайного леса** на обучающей и тестовой выборках (по трём наборам данных — А, Б, В).

Самостоятельное задание.

Чувствительность и специфичность рассчитаны вручную по формулам.

Эффективность дерева и леса оценена на нелинейно разделимых классах.

Подобраны гиперпараметры: для дерева — глубина (max_depth) для снижения переобучения; для леса — число деревьев (n_estimators от 1 до 300 с шагом 10) по максимальному AUC на тестовой выборке.

Результаты подбора выведены в консоль и занесены в отчёт.

Гистограммы и ROC-кривые

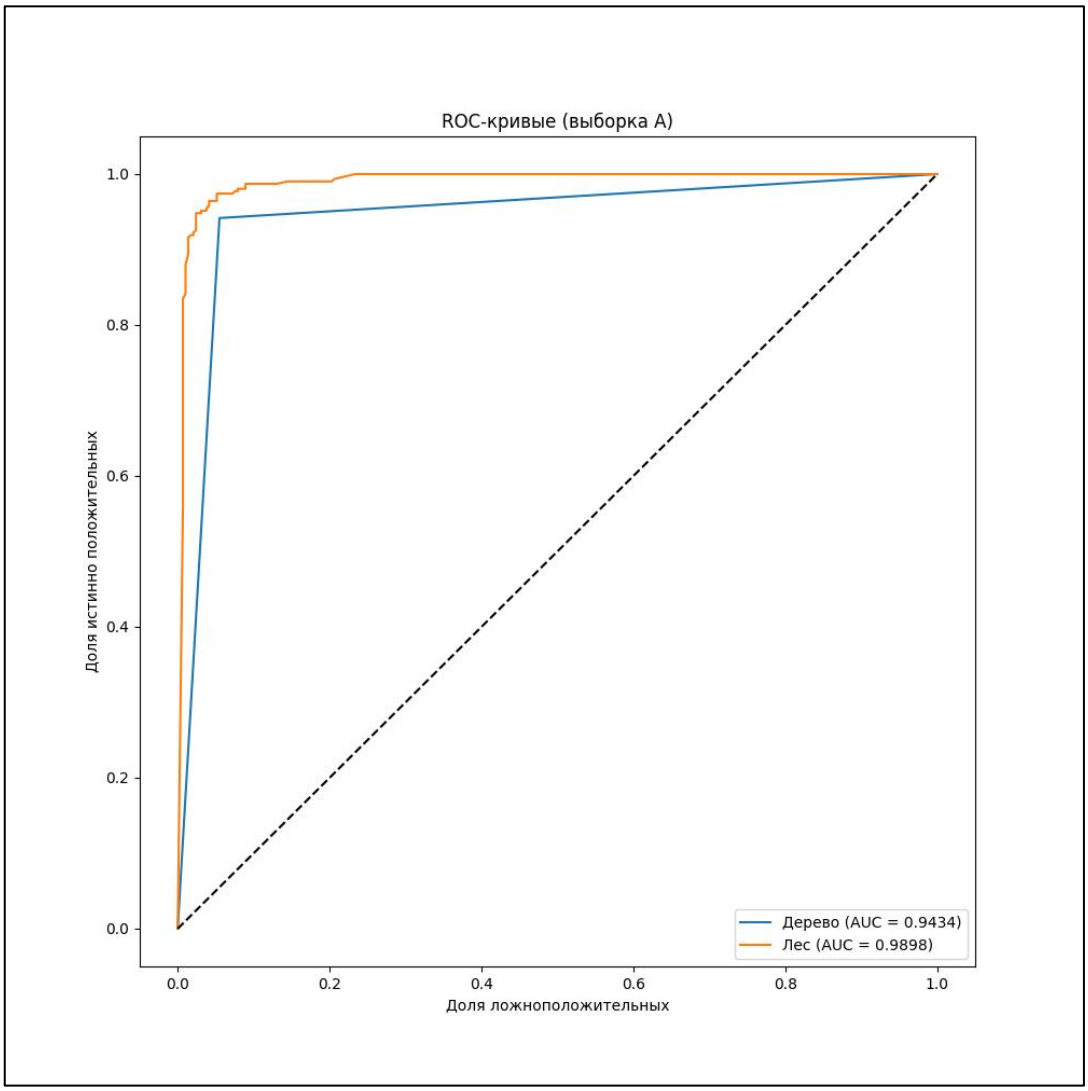

Рисунок 2 — ROC-кривые для дерева и леса, выборка A.

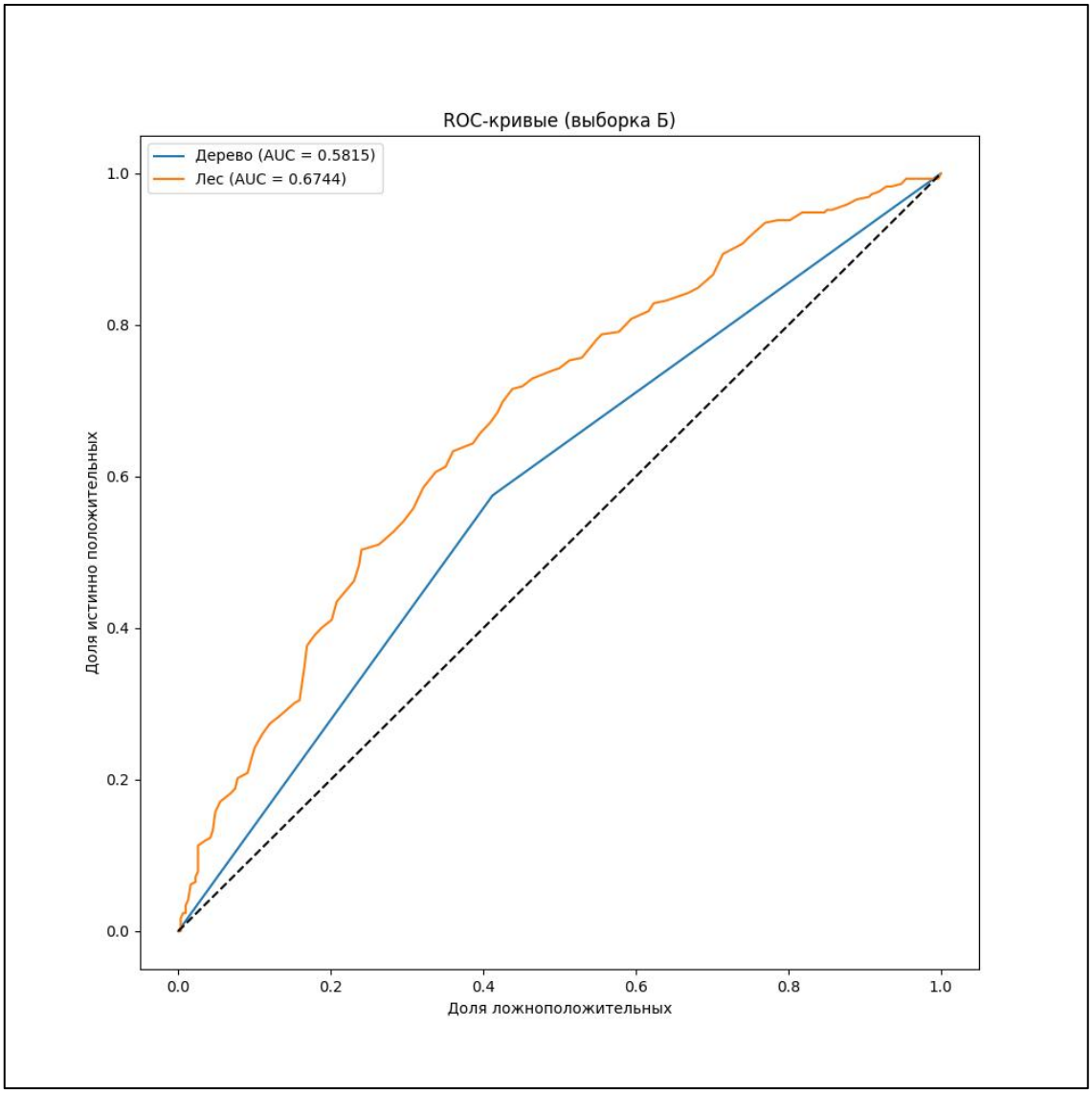

Рисунок 3 — ROC-кривые, выборка Б (плохо разделимые).

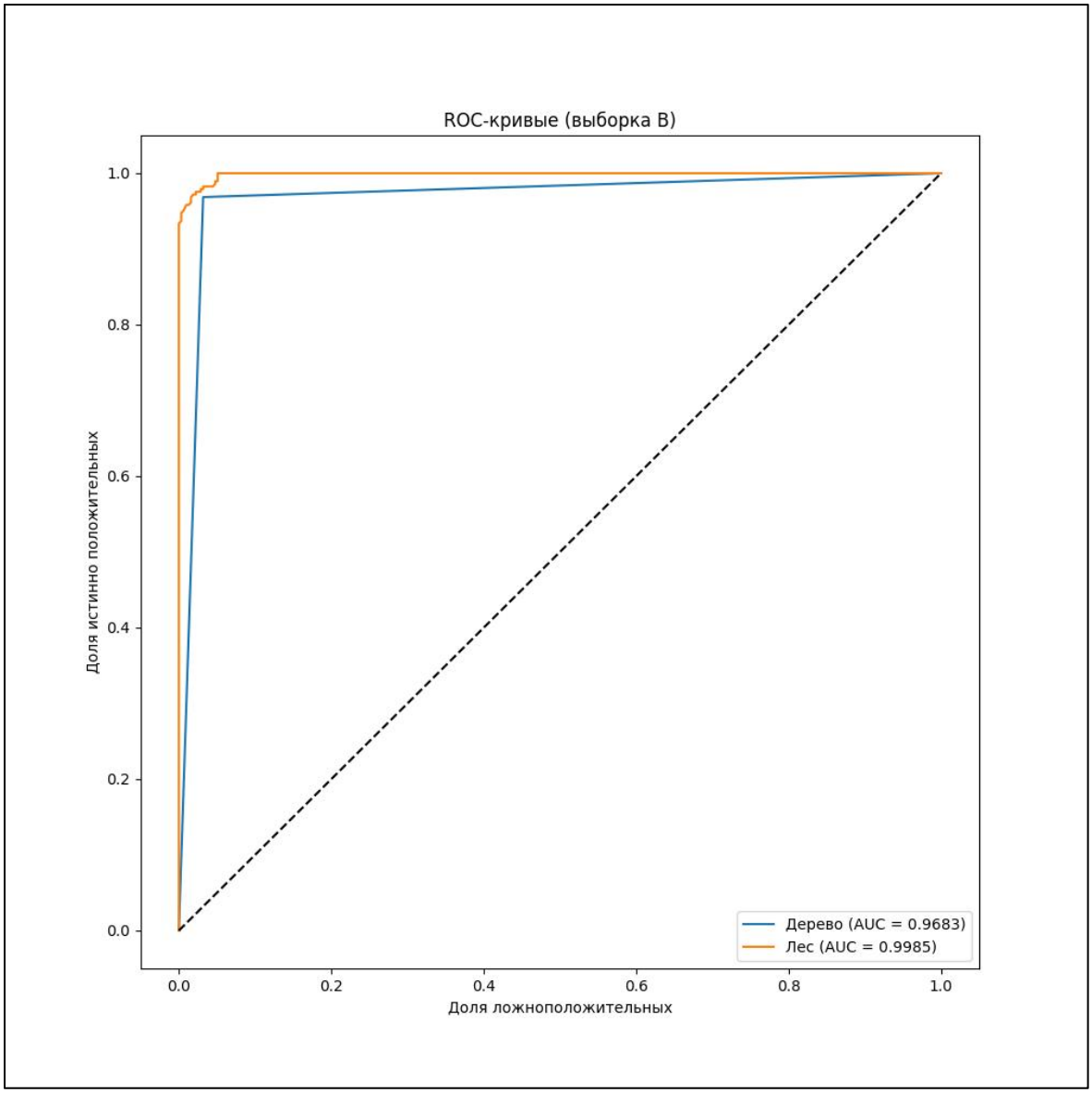

Рисунок 4 — ROC-кривые, выборка B (нелинейно разделимые).

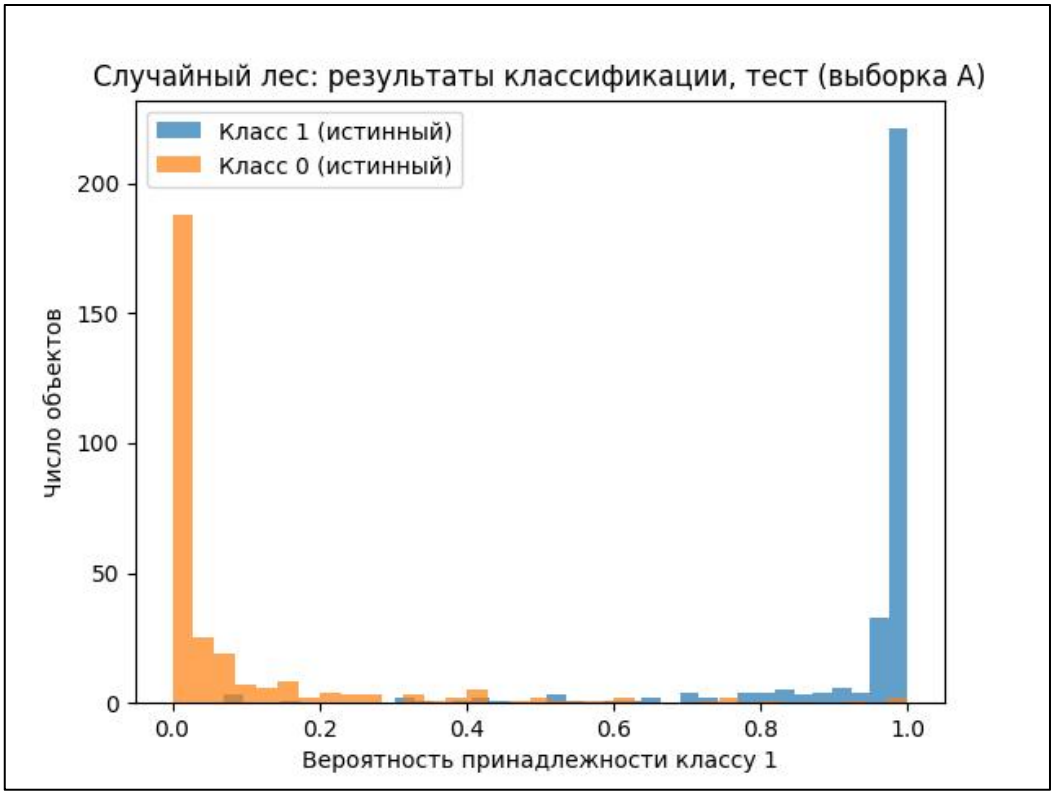

Рисунок 5 — Гистограмма распределения вероятности принадлежности классу 1 (лес), тест, выборка А.

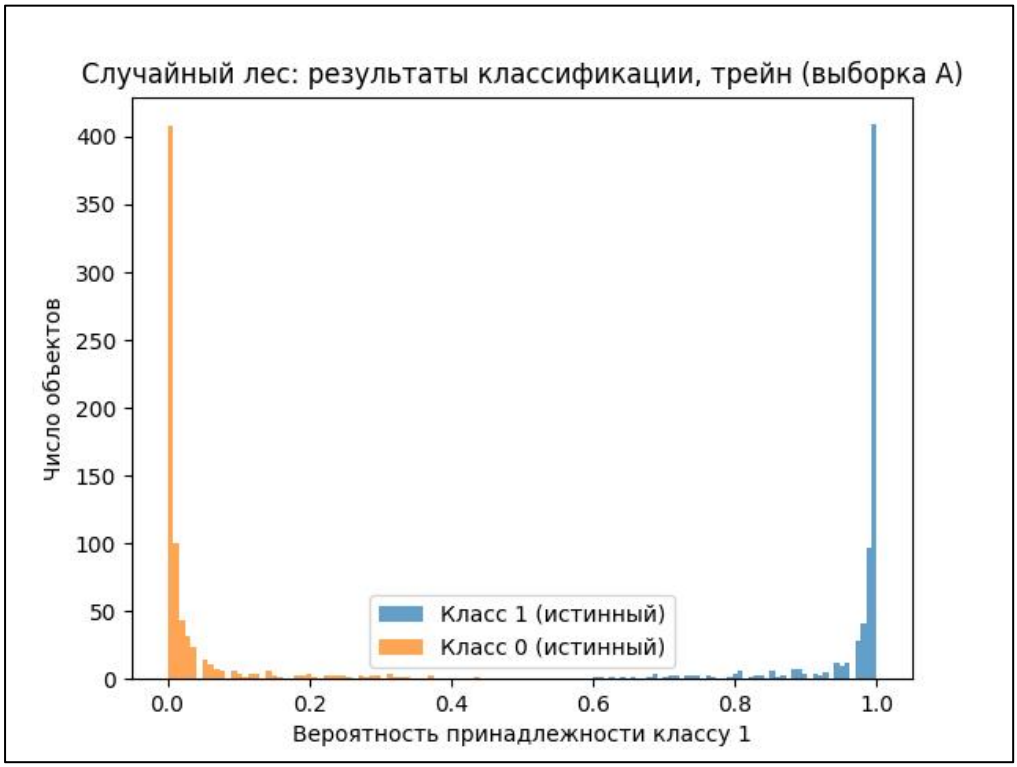

Рисунок 6 — Гистограмма (лес), трейн, выборка А.

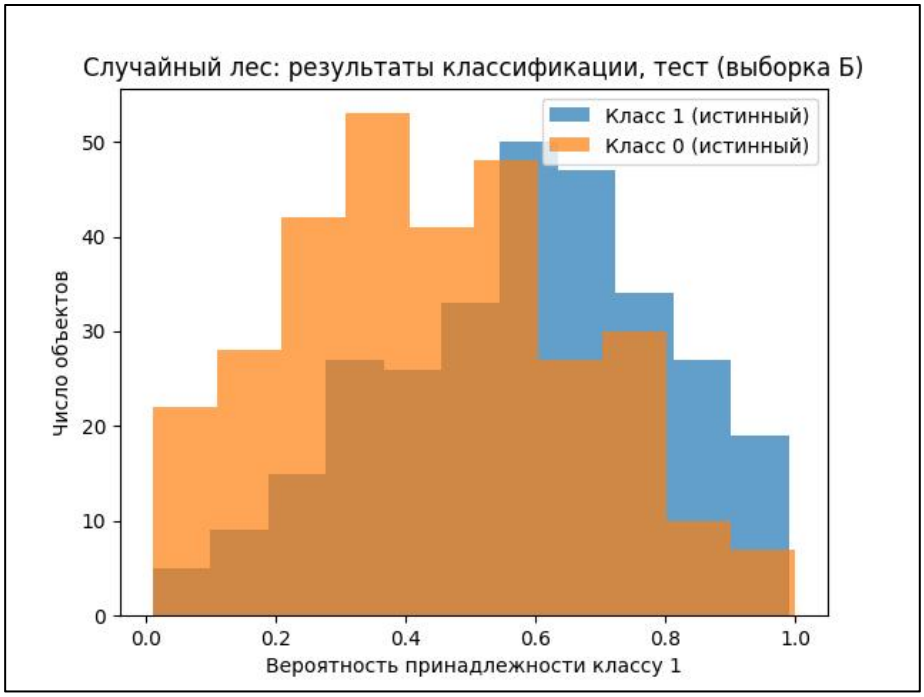

Рисунок 7 — Гистограмма (лес), тест, выборка Б.

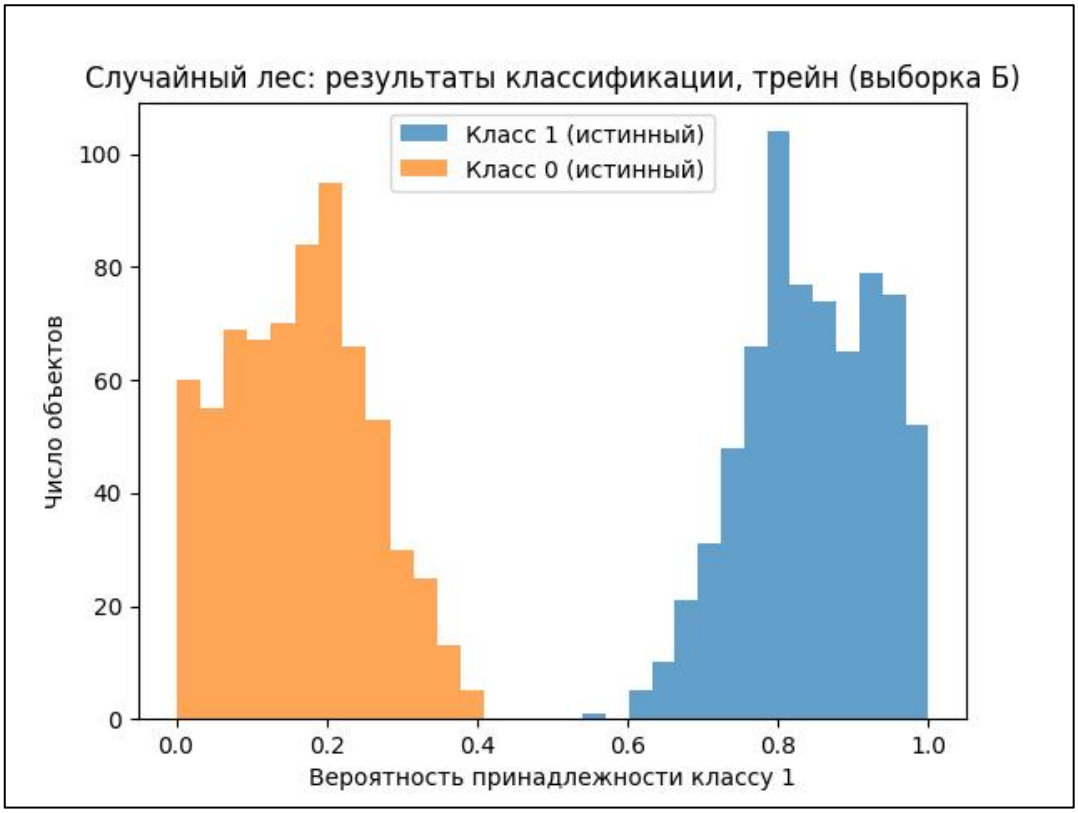

Рисунок 8 — Гистограмма (лес), трейн, выборка Б.

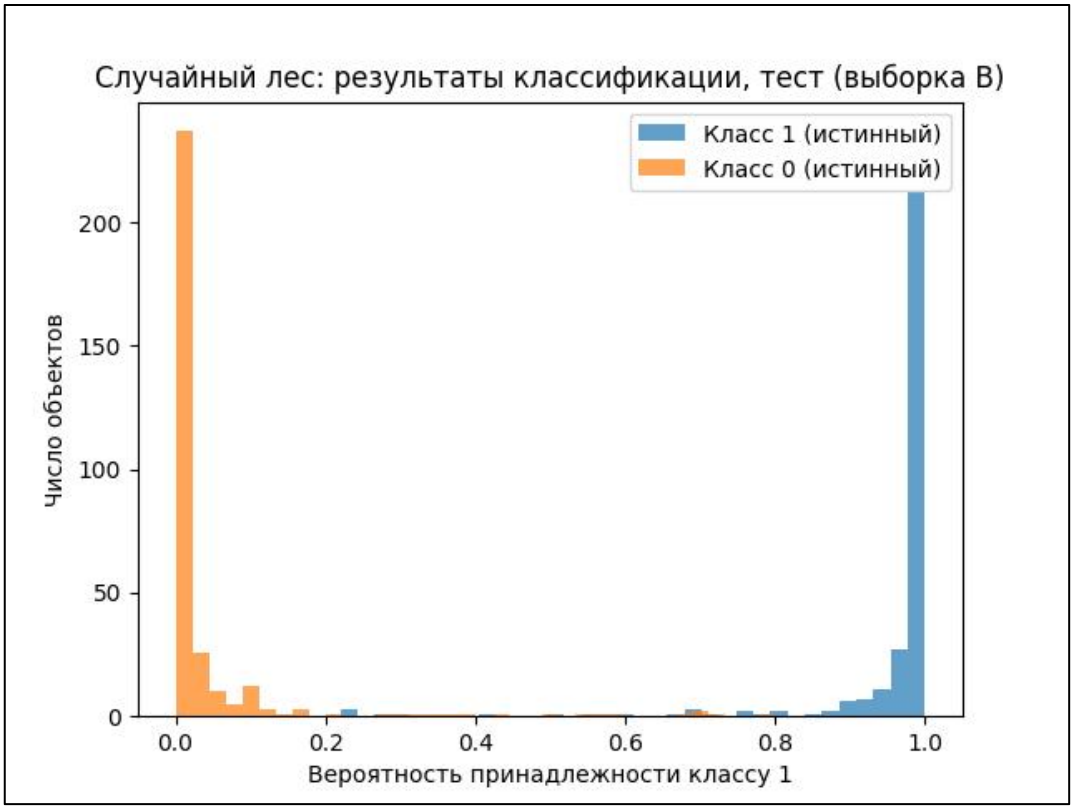

Рисунок 9 — Гистограмма (лес), тест, выборка B.

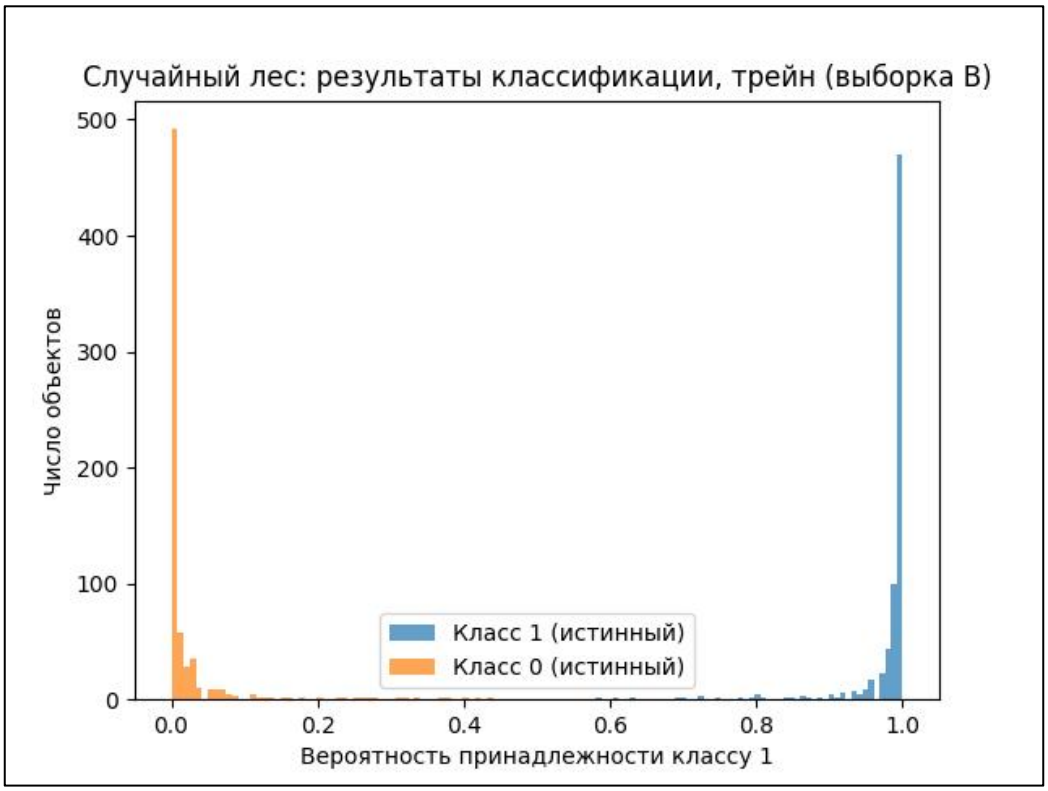

Рисунок 10 — Гистограмма (лес), трейн, выборка B.

Оценка по ROC-кривым и гистограммам

ROC-кривые

Выборка А (хорошо разделимые): Кривые дерева и леса сильно выходят над диагональю случайного классификатора. У дерева AUC ≈ 0,94, у леса ≈ 0,99 — лес даёт более высокий TPR при том же FPR и лучше разделяет классы. Обе модели уверенно работают на хорошо разделимых данных.

Выборка Б (плохо разделимые): AUC дерева около 0,58, леса — около 0,67.

Оба лишь немного лучше случайного угадывания (0,5); лес стабильно выше дерева.

Кривые ближе к диагонали — при сильном перекрытии классов ни дерево, ни лес не достигают высокой разделимости. Это согласуется с характером выборки Б (сближенные средние, большие СКО).

Выборка В (нелинейно разделимые): AUC дерева ≈ 0,97, леса ≈ 0,998. Кривые прижаты к верхнему левому углу — модели хорошо разделяют классы; лес снова лучше дерева. На нелинейной структуре (овал внутри U) кусочно-линейные границы дерева и леса оказываются эффективнее одной гиперплоскости (логистическая регрессия).

Итог по ROC: Случайный лес по AUC превосходит дерево на всех выборках. Наилучшие значения AUC — на выборках А и В; на выборке Б качество ограничено самими данными.

Гистограммы (случайный лес)

Выборка А — трейн и тест: Распределения вероятности класса 1 для истинных классов 0 и 1 сильно разделены: класс 0 концентрируется у 0, класс 1 — у 1, перекрытие в средней зоне

минимально. На тесте пики сохраняются (~180 объектов класса 0 у 0, ~220 класса 1 у 1) — модель уверенно обобщает.

Выборка Б —

трейн: На обучении класс 0 даёт пик около 0,2, класс 1 — около 0,8; разделение на трейне хорошее.

Тест: Перекрытие в зоне 0,4–0,7 заметное: много объектов обоих классов получают вероятности около 0,5. Пики класса 0 в районе 0,3–0,4 и класса 1 в 0,6–0,7 показывают, что на новых данных модель часто неуверена — это соответствует низкому AUC на выборке Б.

Выборка В — трейн и тест: На трейне и тесте класс 0 концентрируется у 0, класс 1 — у 1, перекрытие небольшое. Модель уверенно разделяет нелинейно разделимые классы; гистограммы согласуются с высоким AUC (≈0,998) на выборке В.

Итог по гистограммам: Чем лучше разделимость данных (А, В), тем сильнее пики у 0 и 1 и меньше перекрытие. На выборке Б перекрытие в середине шкалы вероятностей отражает природу данных и ограниченное качество при пороге 0,5.

Сравнение трейн/тест показывает, что на А и В лес не переобучается (картина на тесте похожа на трейн); на Б обобщение ограничено пересечением классов.

Таблицы с результатами оценки качества классификации

Выборка А (хорошо разделимые)

Модель		N объектов	Точность, %	Чувствительность , %	Специфичность, %
Дерево	**Train**	1400	100	100	100
	Test	600	94.33	94.17	94.50
Лес	**Train**	1400	100	100	100
	Test	600	96.17	96.44	95.88

AUC дерево: 0.9434, AUC лес: 0.9898

Выборка Б (плохо разделимые)

Модель		N объектов	Точность, %	Чувствительность , %	Специфичность, %
Дерево	**Train**	1400	100	100	100
	Test	600	58.17	57.53	58.77
Лес	**Train**	1400	100	100	100
	Test	600	63.00	65.75	60.39

AUC дерево: 0.5815, AUC лес: 0.6744

Выборка В (нелинейно разделимые)

Модель		N объектов	Точность, %	Чувствительность , %	Специфичность, %
Дерево	**Train**	1400	100	100	100
	Test	600	96.83	96.85	96.82
Лес	**Train**	1400	100	100	100
	Test	600	97.33	97.55	97.13

AUC дерево: 0.9683, AUC лес: 0.9985

Вывод по таблицам

На всех выборках и у дерева, и у леса точность, чувствительность и специфичность на обучающей выборке равны 100 %.

Это ожидаемо: дерево и лес способны идеально подстроиться под обучающие данные; оценку обобщающей способности дают метрики на тесте

Выборка А: на тесте дерево даёт точность 94,33 %, лес — 96,17 %; чувствительность и специфичность выше 94 %. AUC дерева 0,9434, леса 0,9898. Обе модели хорошо обобщают на хорошо разделимых данных; лес немного превосходит дерево.

Выборка Б: на тесте точность дерева 58,17 %, леса 63,00 %; чувствительность и специфичность в районе 57–66 %. AUC дерева 0,5815, леса 0,6744 — оба близки к случайному угадыванию (0,5).

Выборка В: на тесте дерево — 96,83 % точности, лес — 97,33 %; чувствительность и специфичность выше 96 %. AUC дерева 0,9683, леса 0,9985.

Итог: Случайный лес по всем метрикам на тесте превосходит одно дерево. Качество напрямую зависит от разделимости данных: наивысшее на А и В, низкое на Б. Разрыв между train (100 %) и test особенно велик на выборке Б — типичное проявление переобучения при сложной (пересекающейся) структуре классов.

Результаты подбора наилучших гиперпараметров моделей

Дерево решений (снижение переобучения).

Перебирались значения max_depth (например, 3, 5, 10, 15, None) на выборке Б.

Для каждого зафиксированы точность на обучающей и тестовой выборках.

Наилучшее значение max_depth по тестовой точности: 5 (test accuracy = 63.17%).

Ограничение глубины уменьшает переобучение: точность на train снижается, на test может вырасти.

```
--- Подбор max_depth дерева (выборка Б) ---
  max_depth=3: train=62.71%, test=61.17%
  max_depth=5: train=66.64%, test=63.17%
  max_depth=10: train=80.57%, test=59.17%
  max_depth=15: train=94.29%, test=58.67%
  max_depth=None: train=100.00%, test=58.17%
  Лучший max_depth по тестовой точности: 5 (test accuracy = 63.17%)
```

Случайный лес (число деревьев).

Перебор n_estimators от 1 до 300 с шагом 10 на выборке B. Наилучшее n_estimators по AUC на тестовой выборке: n_estimators: 201, AUC на тесте: 0.6887

С ростом числа деревьев AUC обычно сначала растёт, затем стабилизируется.

```
--- Подбор n_estimators леса (от 1 до 300, шаг 10), выборка В ---
  n_estimators=1: AUC на тесте=0.0000
  n_estimators=11: AUC на тесте=0.5727
  n_estimators=21: AUC на тесте=0.6855
  n_estimators=31: AUC на тесте=0.6946
  n_estimators=41: AUC на тесте=0.7031
  n_estimators=51: AUC на тесте=0.7031
  n_estimators=61: AUC на тесте=0.7039
  n_estimators=71: AUC на тесте=0.7039
  n_estimators=81: AUC на тесте=0.7039
```

```
n_estimators=91: AUC на тесте=0.7039
n_estimators=101: AUC на тесте=0.7039
n_estimators=111: AUC на тесте=0.7048
n_estimators=121: AUC на тесте=0.7048
n_estimators=131: AUC на тесте=0.7048
n_estimators=141: AUC на тесте=0.7048.
 …
Наилучшее n_estimators: 101, AUC на тесте: 0.7048
```

Вывод

В работе реализованы классификаторы на основе дерева решений и случайного леса, проведены эксперименты на трёх типах данных: хорошо разделимые нормальные (А), плохо разделимые нормальные (Б), нелинейно разделимые (В).

Дерево решений на обучающей выборке даёт точность, равную или близкую к 1, а на тестовой — существенно ниже; это типичное переобучение. Ограничение глубины (max_depth) позволяет улучшить обобщение. Случайный лес в среднем даёт более устойчивые результаты на тесте и меньший разрыв между train и test; AUC леса обычно не ниже, а часто выше, чем у одного дерева.

На нелинейно разделимых данных (выборка В) дерево и лес справляются лучше линейной логистической регрессии за счёт кусочно-линейных границ. ROC-кривые и AUC наглядно показывают качество разделения классов; гистограммы вероятностей леса отражают степень уверенности модели и перекрытие классов.

Подбор гиперпараметров (глубина дерева, число деревьев в лесе) улучшает компромисс между точностью на обучении и обобщающей способностью.

Код программы:

```
"""
Лабораторная работа 3
"""
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, roc_auc_score

FIGURES_DIR = Path(__file__).resolve().parent / "figures"
FIGURES_DIR.mkdir(exist_ok=True)

try:
    import lab1.DataGenerator as dg
except ImportError:
    import sys
    from pathlib import Path as _Path
    _root = _Path(__file__).resolve().parent.parent
    if str(_root) not in sys.path:
        sys.path.insert(0, str(_root))
    import lab1.DataGenerator as dg

try:
    import scikitplot as skplt
except ImportError:
    skplt = None

N = 1000


def sensitivity_specificity(Y_true, Y_pred):
    Y_true = np.asarray(Y_true).ravel()
    Y_pred = np.asarray(Y_pred).ravel()
    Y_true = (Y_true != 0).astype(int)
    Y_pred = (Y_pred != 0).astype(int)
    TP = np.sum((Y_true == 1) & (Y_pred == 1))
    TN = np.sum((Y_true == 0) & (Y_pred == 0))
    FP = np.sum((Y_true == 0) & (Y_pred == 1))
    FN = np.sum((Y_true == 1) & (Y_pred == 0))
    sens = TP / (TP + FN) if (TP + FN) > 0 else 0.0
    spec = TN / (TN + FP) if (TN + FP) > 0 else 0.0
    return sens, spec


def split_70_30(X, Y):
    n = len(Y)
    train_count = round(0.7 * n)
    return (
        X[:train_count], Y[:train_count],
        X[train_count:], Y[train_count:]
    )


def run_dataset(Xtrain, Ytrain, Xtest, Ytest, suffix, title_suffix):
    """
    Обучает дерево и лес на выборке, считает метрики, строит ROC и
```

```
гистограммы для леса.
    """

    # Дерево решений
    tree = DecisionTreeClassifier(random_state=0).fit(Xtrain, Ytrain)
    tree_pred_train = tree.predict(Xtrain)
    tree_pred_test = tree.predict(Xtest)
    tree_proba_train = tree.predict_proba(Xtrain)
    tree_proba_test = tree.predict_proba(Xtest)

    acc_tree_train = tree.score(Xtrain, Ytrain)
    acc_tree_test = tree.score(Xtest, Ytest)
    sens_tree_train, spec_tree_train = sensitivity_specificity(Ytrain,
tree_pred_train)
    sens_tree_test, spec_tree_test = sensitivity_specificity(Ytest,
tree_pred_test)

    # Случайный лес
    forest = RandomForestClassifier(random_state=0).fit(Xtrain, Ytrain)
    forest_pred_train = forest.predict(Xtrain)
    forest_pred_test = forest.predict(Xtest)
    forest_proba_train = forest.predict_proba(Xtrain)
    forest_proba_test = forest.predict_proba(Xtest)

    acc_forest_train = forest.score(Xtrain, Ytrain)
    acc_forest_test = forest.score(Xtest, Ytest)
    sens_forest_train, spec_forest_train = sensitivity_specificity(Ytrain,
forest_pred_train)
    sens_forest_test, spec_forest_test = sensitivity_specificity(Ytest,
forest_pred_test)

    # ROC-кривые для дерева и леса на одном графике
    Ytest_int = (np.asarray(Ytest) != 0).astype(int) if Ytest.dtype == bool
else np.asarray(Ytest)

    fpr_tree, tpr_tree, _ = roc_curve(Ytest_int, tree_proba_test[:, 1])
    fpr_forest, tpr_forest, _ = roc_curve(Ytest_int, forest_proba_test[:,
1])
    auc_tree = roc_auc_score(Ytest_int, tree_proba_test[:, 1])
    auc_forest = roc_auc_score(Ytest_int, forest_proba_test[:, 1])

    plt.figure(figsize=(10, 10))
    plt.plot(fpr_tree, tpr_tree, label=f"Дерево (AUC = {auc_tree:.4f})")
    plt.plot(fpr_forest, tpr_forest, label=f"Лес (AUC = {auc_forest:.4f})")
    plt.plot([0, 1], [0, 1], "k--")
    plt.xlabel("Доля ложноположительных")
    plt.ylabel("Доля истинно положительных")
    plt.title(f"ROC-кривые ({title_suffix})")
    plt.legend()
    plt.savefig(FIGURES_DIR / f"lab3_roc_{suffix}.png")
    plt.close()

    if skplt is not None:
        skplt.metrics.plot_roc(Ytest_int, forest_proba_test, figsize=(10,
10))
        plt.title(f"ROC — случайный лес ({title_suffix})")
        plt.savefig(FIGURES_DIR / f"lab3_roc_forest_only_{suffix}.png")
        plt.close()
```

```
    # Гистограммы распределения вероятностей для леса
    plt.figure()
    plt.hist(forest_proba_test[Ytest, 1], bins="auto", alpha=0.7,
label="Класс 1 (истинный)", color="C0")
    plt.hist(forest_proba_test[~Ytest, 1], bins="auto", alpha=0.7,
label="Класс 0 (истинный)", color="C1")
    plt.xlabel("Вероятность принадлежности классу 1")
    plt.ylabel("Число объектов")
    plt.title(f"Случайный лес: результаты классификации, тест
({title_suffix})")
    plt.legend()
    plt.savefig(FIGURES_DIR / f"lab3_hist_forest_test_{suffix}.png")
    plt.close()

    plt.figure()
    plt.hist(forest_proba_train[Ytrain, 1], bins="auto", alpha=0.7,
label="Класс 1 (истинный)", color="C0")
    plt.hist(forest_proba_train[~Ytrain, 1], bins="auto", alpha=0.7,
label="Класс 0 (истинный)", color="C1")
    plt.xlabel("Вероятность принадлежности классу 1")
    plt.ylabel("Число объектов")
    plt.title(f"Случайный лес: результаты классификации, трейн
({title_suffix})")
    plt.legend()
    plt.savefig(FIGURES_DIR / f"lab3_hist_forest_train_{suffix}.png")
    plt.close()

    print(f"[{title_suffix}] AUC дерево: {auc_tree:.4f}, AUC лес:
{auc_forest:.4f}")

    return {
        "tree": {
            "train": (len(Ytrain), acc_tree_train, sens_tree_train,
spec_tree_train),
            "test": (len(Ytest), acc_tree_test, sens_tree_test,
spec_tree_test),
            "auc": auc_tree,
        },
        "forest": {
            "train": (len(Ytrain), acc_forest_train, sens_forest_train,
spec_forest_train),
            "test": (len(Ytest), acc_forest_test, sens_forest_test,
spec_forest_test),
            "auc": auc_forest,
        },
    }


def print_table(results, model_name, dataset_name):
    """Печать таблицы метрик для одной модели (дерево или лес) по одной
выборке."""
    print(f"\n--- {dataset_name} — {model_name} ---")
    print(f"{'':8} {'Число объектов':>14} {'Точность, %':>12}
{'Чувствительность, %':>20} {'Специфичность, %':>18}")
    print("-" * 78)
    for part in ("train", "test"):
        label = "Train" if part == "train" else "Test"
        n, acc, sens, spec = results[model_name][part]
        print(f"{label:8} {n:>14} {acc*100:>11.2f}% {sens*100:>19.2f}%
```

```
{spec*100:>17.2f}%")


# ---------- Выборка A: хорошо разделимые нормальные данные ----------
mu0_A = [0, 2, 3]
mu1_A = [3, 5, 1]
sigma0_A = [2, 1, 2]
sigma1_A = [1, 2, 1]
X_A, Y_A, _, _ = dg.norm_dataset([mu0_A, mu1_A], [sigma0_A, sigma1_A], N)
Xtrain_A, Ytrain_A, Xtest_A, Ytest_A = split_70_30(X_A, Y_A)
results_A = run_dataset(Xtrain_A, Ytrain_A, Xtest_A, Ytest_A, "A", "выборка
A")
print_table(results_A, "tree", "Выборка A")
print_table(results_A, "forest", "Выборка A")

# ---------- Выборка Б: плохо разделимые нормальные данные (средняя степень
пересечения) ----------
mu0_B = [1, 3, 2]
mu1_B = [2, 4, 3]
sigma0_B = [2.5, 2, 2.5]
sigma1_B = [2, 2.5, 2]
X_B, Y_B, _, _ = dg.norm_dataset([mu0_B, mu1_B], [sigma0_B, sigma1_B], N)
Xtrain_B, Ytrain_B, Xtest_B, Ytest_B = split_70_30(X_B, Y_B)
results_B = run_dataset(Xtrain_B, Ytrain_B, Xtest_B, Ytest_B, "B", "выборка
Б")
print_table(results_B, "tree", "Выборка Б")
print_table(results_B, "forest", "Выборка Б")

# ---------- Выборка B: нелинейно разделимые данные ----------
X_C, Y_C, _, _ = dg.nonlinear_dataset_5(N)
Xtrain_C, Ytrain_C, Xtest_C, Ytest_C = split_70_30(X_C, Y_C)
results_C = run_dataset(Xtrain_C, Ytrain_C, Xtest_C, Ytest_C, "C", "выборка
B")
print_table(results_C, "tree", "Выборка B")
print_table(results_C, "forest", "Выборка B")

# ---------- подбор гиперпараметров дерева (глубина) для снижения
переобучения ----------
print("\n--- Подбор max_depth дерева (выборка Б) ---")
best_tree = None
best_test_acc = -1
best_depth = None
for max_d in [3, 5, 10, 15, None]:
    tree_tune = DecisionTreeClassifier(max_depth=max_d,
random_state=0).fit(Xtrain_B, Ytrain_B)
    acc_t = tree_tune.score(Xtrain_B, Ytrain_B)
    acc_v = tree_tune.score(Xtest_B, Ytest_B)
    depth_str = str(max_d) if max_d is not None else "None"
    print(f"  max_depth={depth_str}: train={acc_t*100:.2f}%,
test={acc_v*100:.2f}%")
    if acc_v > best_test_acc:
        best_test_acc = acc_v
        best_depth = max_d
print(f"  Лучший max_depth по тестовой точности: {best_depth} (test
accuracy = {best_test_acc*100:.2f}%)")

# ---------- подбор n_estimators для леса (максимум AUC на тесте) ---------
print("\n--- Подбор n_estimators леса (от 1 до 300, шаг 10), выборка A ---")
best_n = 1
```

```
best_auc = 0.0
for n_est in range(1, 301, 10):
    rf = RandomForestClassifier(n_estimators=n_est,
random_state=0).fit(Xtrain_B, Ytrain_B)
    proba = rf.predict_proba(Xtest_B)[:, 1]
    Ytest_B_int = (np.asarray(Ytest_B) != 0).astype(int)
    auc = roc_auc_score(Ytest_B_int, proba)
    print(f"  n_estimators={n_est}: AUC на тесте={best_auc:.4f}")
    if auc > best_auc:
        best_auc = auc
        best_n = n_est
print(f"  Наилучшее n_estimators: {best_n}, AUC на тесте: {best_auc:.4f}")

print(f"\nГрафики сохранены в папке: {FIGURES_DIR}")
```